Attentive Cognitive Agents for Real-time Virtual Environments

DISSERTATION
zur Erlangung des Grades eines Doktors
der Ingenieurwissenschaften

vorgelegt von M. Sc. Sven Seele

eingereicht bei der Naturwissenschaftlich-Technischen Fakultät
der Universität Siegen
Siegen 2021

Betreuer und erster Gutachter
Prof. Dr. Andreas Kolb
Universität Siegen

Co-Betreuer und zweiter Gutachter
Prof. Dr.-Ing. Rainer Herpers
Hochschule Bonn-Rhein-Sieg

Dritter Gutachter
Prof. Dr.-Ing. Marcin Grzegorzek
Universität zu Lübeck

Tag der mündlichen Prüfung
04.05.2022

Acknowledgements

The long journey to the completion of my dissertation was an exciting ride! Leaving Hochschule Bonn-Rhein-Sieg (H-BRS) and major changes in my private life, before completing the project, created some interesting detours, which were not always easy and made for many late nights. Therefore, I would like to take this opportunity and express my gratitude to those that have helped and guided me throughout this journey. First, I would like to thank my supervisors Prof. Dr. Andreas Kolb, Prof. Dr. Marcin Grzegorzek, and Prof. Dr. Rainer Herpers. I am extremely grateful for your invaluable support, feedback, insights, and patience. I also like to thank Prof. Dr. Christian Bauckhage for the fantastic support in the beginning of this thesis project.

I would also like to give a big thank you to all my coworkers at the Institute of Visual Computing (IVC) for many interesting discussions and ideas, but also for making work at the institute such a pleasure. Thank you, Christoph, David, Ernst, Jens, Katharina, Martin, Nadine, Olli, Rita, Sandra, Thorsten, Timur as well as Prof. Dr. Heiden and Prof. Dr. Hinkenjann. A special thank you goes to Jonas, who has also been my coworker at the Fraunhofer IAIS, my mentor and supervisor, and friend for many years. Your advice, inspiration, and support, which played a significant role in completing this achievement.

I would like to acknowledge all the great students and interns I had the privileged of supervising, for your contributions, effort, and many fruitful discussions. Thank you Björn, Fabian, Francis, Helmut, Luisa, Marco, Norbert, Steffen, Suzannah, Tim G., Tim M., Timo, and Xinyi. I would like to especially mention Thomas and Tobias, who have been my students, co-workers, and friends. Thank you for your continuing support.

Completing this dissertation would not have been possible without the financial support I received via the AVeSi project (FKZ 17028X11) funded by FHprofUnt program of the German Federal Ministry of Education and Research (BMBF), the EPICSAVE project (FKZ 01PD15004A) funded as part of the program Digitale Medien in der Beruflichen Bildung 2 by the BMBF and by the European Social Fonds (ESF), the FIVIS project (FKZ FP307) with additional funding by the Deutsche Gesetzliche Unfallversicherung (DGUV), the IVC PhD scholarship, and the scholarship provided by the Graduate Institute (GI) of H-BRS. Support in form and content was also provided by the GI of H-BRS and the DFG Research Training Group 1564 of the University of Siegen.

Thank you also to everybody who supported me in any way, shape, or form, particularly Sebastian, Thorsten, and Tobias for their insightful feedback and suggestions. I would also like to thank the SICAT GmbH & Co. KG, the HICAT GmbH, and the ELISE GmbH for

giving me the opportunity and flexibility of continuing my research while being a full-time employee.

Last, but most certainly not least, I would like to thank my family and friends, especially my parents and my sister for always believing in me and for the love and support I received from them throughout my life. Finally, I would like to thank the two most important people in my live, Kathleen and my son Noah. I could not have completed this journey without your endless encouragement, understanding, patience, and love. For this and countless other things I will be forever grateful.

Sven Seele
Sankt Augustin, October 2022

Abstract

Intelligent virtual agents provide a framework for simulating more life-like behavior and increasing plausibility in virtual training environments. They can improve the learning process if they portray believable behavior that can also be controlled to support the training objectives. In the context of this thesis, cognitive agents are considered a subset of intelligent virtual agents (IVA) with the focus on emulating cognitive processes to achieve believable behavior. The complexity of employed algorithms, however, is often limited since multiple agents need to be simulated in real-time. Available solutions focus on a subset of the indicated aspects: plausibility, controllability, or real-time capability (scalability). Within this thesis project, an agent architecture for attentive cognitive agents is developed that considers all three aspects at once. The result is a *lightweight cognitive agent architecture* that is customizable to application-specific requirements. A generic trait-based personality model influences all cognitive processes, facilitating the generation of consistent and individual behavior. An additional mapping process provides a formalized mechanism to transfer results of psychological studies to the architecture. Personality profiles are combined with an emotion model to achieve situational behavior adaptation. Which action an agent selects in a situation also influences plausibility. An integral element of this selection process is an agent's knowledge about its world. Therefore, synthetic perception is modeled and integrated into the architecture to provide a credible knowledge base. The developed perception module includes a unified sensor interface, a memory hierarchy, and an attention process. With the presented realization of the architecture (CA^{2}RVE), it is possible for the first time to simulate cognitive agents, whose behaviors are simultaneously computable in real-time and controllable. The architecture's applicability is demonstrated by integrating an agent-based traffic simulation built with CA^{2}RVE into a bicycle simulator for road-safety education. The developed ideas and their realization are evaluated within this work using different strategies and scenarios. For example, it is shown how CA^{2}RVE agents utilize personality profiles and emotions to plausibly resolve deadlocks in traffic simulations. Controllability and adaptability are demonstrated in additional scenarios. Using the realization, 200 agents can be simulated in real-time (50 FPS), illustrating scalability. The achieved results verify that the developed architecture can generate plausible and controllable agent behavior in real-time. The presented concepts and realizations provide sound fundamentals to everyone interested in simulating IVA in real-time environments.

Zusammenfassung

Virtuelle Umgebungen sind ein effizientes Trainingswerkzeug, besonders wenn Trainingsszenarien durch die Simulation von intelligenten virtuellen Agenten (IVA) unterstützt werden. Dafür muss das Agentenverhalten plausibel und steuerbar sein, um die Immersion nicht zu mindern und das Trainingsziel zu unterstützen. Methoden, mit denen diese Anforderungen erfüllt werden, können jedoch nicht beliebig komplex sein, da oft mehrere Agenten in Echtzeit simuliert werden müssen. Im Rahmen dieser Arbeit stellt sich somit die Aufgabe eine Lösung zu entwickeln, welche die Anforderungen an Plausibilität, Kontrollierbarkeit und Skalierbarkeit zusammen adressiert. Die Plausibilität wird dabei durch das Simulieren kognitiver Prozesse erreicht. Ein Kernelement der entwickelten *leichtgewichtigen kognitiven Agentenarchitektur* ist ein Persönlichkeitsprofil, das sich auf alle anderen kognitiven Prozesse auswirkt. Somit kann konsistentes, individualisiertes Verhalten erzeugt werden, welches zusätzlich mit Hilfe eines entwickelten, formalen Abbildungsprozesses aus psychologischen Persönlichkeitsstudien abgeleitet werden kann. Durch die Kopplung des Profils mit Emotionen kann das Verhalten dynamisch an die Gegebenheiten eines Agenten angepasst werden. Welche Aktion ein Agent in einer Situation auswählt, beeinflusst ebenfalls die Glaubwürdigkeit. Ein wichtiger Bestandteil dieses Auswahlprozesses ist das Wissen, das ein Agent über seine Umgebung besitzt. Um eine plausible Wissensbasis bereit zu stellen, wurde ein Perzeptionsmodul konzipiert und integriert, das eine einheitliche Sensorschnittstelle definiert und Informationen in einem hierarchischen Gedächtnis durch einen Aufmerksamkeitsprozess verwaltet. Die realisierte Architektur erlaubt erstmalig die Simulation kognitiver Agenten, die gleichzeitig kontrollierbar und in Echtzeit berechenbar sind. Demonstriert wird dies u. a. durch die Umsetzung als Software-Architektur (CA^{2}RVE) und eine damit entwickelte agentenbasierte Verkehrssimulation. Die entwickelten Ideen und deren Realisierung wurden im Rahmen der Arbeit anhand verschiedener Strategien evaluiert. Es wird gezeigt wie CA^{2}RVE-Agenten, anhand ihrer Persönlichkeiten und Emotionen, verschiedene Verkehrssituationen glaubwürdig auflösen. Die Kontrollierbarkeit und Anpassungsfähigkeit wird ebenfalls in Evaluationsszenarien demonstriert. Die Skalierbarkeit wird durch die Simulation von 200 Agenten in Echtzeit (50 FPS) nachgewiesen. Die Ergebnisse zeigen, dass eine Architektur für das Generieren von plausiblem, kontrollierbarem und echtzeitfähigem Agentenverhalten erfolgreich realisiert wurde. Damit stellt diese Arbeit fundamentale Grundlagen für diejenigen bereit, die kognitive IVA in Echtzeitanwendungen einsetzen wollen.

Contents

List of Figures

List of Tables

1

Introduction

"If a machine is expected to be infallible, it cannot also be intelligent."

- Alan Turing

Virtual environments (VE) are a promising tool for teaching real-world skills to learners. They allow training within potentially threatening environments without exposing trainees to physical harm. By providing immersive and believable experiences, learning effects can be improved even further (see, e.g., [Naj98, RJ00]). At the same time, advances in and availability of technology enable the use of immersive training application in more and more areas. Improving visual fidelity is often the prime aspect to increasing immersion and presence in interactive environments. Continuous improvements of computer hardware result in steady improvements of rendering quality, animation, and physics simulations. However, visual fidelity is not the only aspect of immersive training experiences. In many cases, the simulated environment must also include entities that support or hinder users in fulfilling their training tasks. These entities are typically referred to as *agents*. Sometimes, appropriately interacting with these agents is the skill to be learned. In other scenarios, agents are required to create the illusion of a living world. In all cases, agents must act according to the expectations set by the virtual world to achieve and maintain immersion, plausibility, and educational or entertainment value. If agents exhibit implausible behavior, a user's sense of immersion can break even under high fidelity visual conditions. Consequently, improving the plausibility of generated agent behavior has gained recognition in academia and the games industry (see, e.g., [KS16]).

In training scenarios, it is also important for trainees to be able to recognize a plausible connection between an agent's choice of action and the situation it resulted from. Behavior portrayed by agents can thus never be completely non-deterministic. Instead, agents should perceive, understand, and reason about their current situation before selecting an available

action. In contrast, this causally determined action selection does not mean that an agent's behavior is always predictable, but it must be comprehensible to an observer in retrospect. The means by which this correlation between situation and reaction is achieved varies widely depending on various factors, such as technological, temporal, or financial constraints, area of application, target audience, pursued purpose, or others. Intelligent virtual agents (IVAs) provide a framework for achieving the objective of simulating more human-like and more plausible behavior. In the context of this thesis, *cognitive agents* are considered a subset of IVA with a specific focus on emulating cognitive processes for behavior simulation.

1.1 Motivation for Attentive Cognitive Agents

There are many reasons for employing IVAs or cognitive agents in simulations and games. The demand for immersive worlds with interactive inhabitants is continuously increasing, yet creating believable characters is one of the most challenging endeavors in designing these virtual worlds (cf. [LB19]). In game development, the main constraints for virtual characters are typically production cost and the ability to author behavior. Consequently, behavior of non-player characters (NPCs) is mostly being pre-defined at design time (i.e., *scripted*), especially in production environments. It is important to point out that scripted behavior does not necessarily mean that it is simple. Modern NPCs exhibit a multitude of complex behaviors. However, this also means that authoring their behavior is an overly complex task that is prone to errors. The more complex the NPC behavior and the virtual worlds they inhabit become, the more beneficial are agents that autonomously choose their actions based on their current surroundings. Another major challenge in combining autonomous agents with games is that they must work in unison with game design to keep players engaged. For example, NPC opponents need to provide a challenge to the player while acting within plausible bounds to avoid breaking the player's suspension of disbelief. Challenge and plausibility need to be in accordance with the game design and the world created therein [YT18]. Therefore, generated behavior must also be controllable. Similar arguments can be made for IVA applications, although they are often more experimental in nature. However, even in research environments, financial resources must be considered and agent behavior serves a specific purpose, which means there must be a way of authoring it accordingly.

Cognitive agents that emulate human behavior are a promising solution to overcome this challenge. The idea is not new, and various approaches already exist that model certain cognitive capabilities or processes of humans to generate more human-like agent behavior (e.g., [ALS09, BGG19, BL06, BKA+05a, CDB+02, LB19, RJSL10, RJ00, SSSS16]). Applications can be as simple as chat bots or as complex as trying to achieve general intelligence in

cognitive architecture research. However, as will be formulated throughout this chapter and Chapter 2, existing methodologies do not provide enough flexibility, are not suited for real-time, multi-agent applications, or do not offer authoring capabilities. For example, the objective of achieving human-level intelligence resulted in cognitive architectures like Soar [Lai12] or ACT-R [RTO19], which have become increasingly complex over time. Capabilities are added to improve the system and are not meant to be switched on or off during runtime or to be customized towards a specific application domain. As a result, it is difficult to apply cognitive architectures to real-time environments and even more difficult to scale solutions to multi-agent settings. Furthermore, individualizing behaviors of different agents or controlling outcomes towards application-dependent requirements are not of concern to cognitive architecture research. While these aspects are considered in IVA research, the focus is on communication and social interaction, often with a human user, and less on observable actions. Middleware solutions, e.g., CIGA [vO14] or POGAMUT [GBK+10], try to bridge the gap between agent-specific topics and game engine technology, but as a result require application developers to be knowledgeable in both fields. Additionally, middleware or cognitive architectures are difficult to optimize for real-time environments due to their logical and physical separation from the system used to realize the virtual environment. Thus, there is a need for a new concept that combines cognitive components with flexible composition and controllability. A generalized concept should also allow customization towards different intended uses, from populating virtual training environments to simulating embodied conversational agents to investigating human perception.

Innovative approaches, especially in game development, typically focus on the learning and decision-making aspects of agents to improve believability (e.g., [HGH+18, JBV+18]). Further aspects, such as personality and perception including attention mechanisms, are rarely mentioned as part of virtual agent or NPC research. However, within this thesis, it is argued that these are substantial components required for generating plausible behavior.

1.2 Research Context

The work presented here is related to multiple research domains. It is closely correlated with Intelligent Virtual Agents (IVA) and Embodied Conversational Agents (ECA) research. In both research areas, agents are typically represented as individuals including a model of personality, emotional responses, and mood states. Agents also need to be able to perceive, reason, and act in real-time within a dynamic simulated environment. However, the focus is often on communication with a human user, i.e., language processing, dialog generation, gestures, and facial expressions [vO14], which are not considered in this thesis project.

Cognitive architecture research is concerned with developing (software) agents with human-level intelligence and in the process gaining insights into the functionality of the human brain. Resulting architectures are typically extraordinarily complex and computationally expensive. While this research has only little practical relevance for real-time multi-agent systems, existing architectures provide important inspirations for the overall design presented in this thesis project.

One aspect of this work is scalability to enable the use of a cognitive agent architecture in real-time applications. Consequently, besides agent design, aspects of virtual reality (VR) and virtual environment (VE) research, like rendering, simulation, display technologies, etc., also need to be considered during the development process. Another research area where real-time performance and optimization play critical roles is game development. The task of generating plausible behavior for non-player characters (NPC) in digital games is often referred to as game AI or just AI. Without further context, these terms are ambiguous, especially since there are academic research fields denoted by the same term. Therefore, the term game AI is used throughout this thesis to refer to techniques used in the gaming industry related to NPCs. While this area of games technology includes many aspects, e.g., animation, physics simulations, and path finding, elements related to action selection are mostly considered in this thesis. However, techniques for game AI do not only need to scale well, but they must also allow control over generated behavior as it must support the game design. Without control, it is impossible to have any influence on the fun and entertainment of an application. Players want to win, but be challenged, and whenever they lose, they do not want to feel cheated by a game's AI [Lid04]. At the same time, game AI developers strive towards making NPCs more autonomous to keep the authoring task manageable because game worlds and NPC behavior continuously increase in complexity. Due to these reasons, game AI and games technology is an important source of inspiration for the work presented here. The connection to this work will be most apparent in the description of the realization of the proposed architecture design in Chapter 6.

The objective of this thesis project is to explore and close the current gap regarding cognitive agent architectures for real-time virtual environments. To achieve the objective, four research questions are formulated and investigated. To provide answers to the RQs, relevant approaches and results from all the mentioned research domains are utilized to create a new cognitive agent architecture that is controllable, scalable to real-time, multi-agent applications, and capable of generating plausible agent behavior. As a result, it will be possible to simulate agents, which make comprehensible decisions based on their perception of their virtual environment. Agents will show individualized behavior that is not based

on randomization and adapts to environmental stimuli. Finally, it will be possible to adjust behavior generation to application-specific requirements.

1.3 Research Questions

Using a combination of the considered research domains, the following research questions are investigated. From each research question specific subtasks are derived and presented in Chapters 3 through 6. By fulfilling these subtasks, prospective answers to the research questions are developed and evaluated.

RQ1: What is an appropriate design of an attentive cognitive agent architecture for virtual environments?

Existing architectures and approaches (e.g., [ALS09, BGG19, GBK+10, Lai12, LB19, RTO19, vO14]) are not able to simultaneously address (1) plausibility of behavior, (2) controllability of behavior, and (3) scalability. Therefore, one of the main objectives of this thesis project is to design an agent architecture that adheres to these three main principles. The resulting design should provide a flexible foundation for adaptive agent behavior configurable to specific application scenarios. The intention is to achieve plausibility by emulating human-like behavior, which is to be realized by simulating cognitive components, e.g., personality, perception, attention, memory, and decision-making. Therefore, inspiration is drawn from ideas and concepts of cognitive architectures. The fields multi-agent systems (MAS) and agent-based systems (ABS) are also considered since multiple co-operating agents are to be simulated in a common environment. As a result, not only is each agent's individual behavior of interest but also the global behavior emerging from agent interactions in a complex system. In game AI and VR/VE, the goal is to provide an interactive experience to a user with a specific objective, e.g., education or entertainment. Simulated agents must support these objectives and must be embedded into a real-time system. Based on this constraint, both fields serve as source for tried and tested algorithms and technologies. IVA and ECA are the domains that correspond most with the research described here, as the objectives are closely aligned, but often focus on human-computer interaction; specifically on speech, text output, facial and gaze expressions (cf. [tSKT+20]). Other areas of interest are, e.g., detecting user emotion and guessing user intention as well as trust, acceptance, and credibility of agents [CHLC18, RGA+09, VSB+20].

RQ2: Can the agent architecture systematically generate individual, dynamic agent behavior?

Focusing on the plausibility constraint, one requirement resulting from the research gap is the generation of individualized behavior as multiple agents should be observable during one simulation. For agents to be distinguishable by their actions, an agent's behavior should be consistent with its past actions [Ort03]. One common way of achieving this individualization in IVA research is by assigning a personality to each agent. While the type and characteristics of theses personalities can be arbitrarily defined to suit the application, another objective of this thesis project is to utilize this mechanism to link synthetic behavior to real human behavior. Therefore, the task is to transfer knowledge about human behavior to the architecture and map it onto individual agent behavior during simulation. Furthermore, to avoid repetitive and dull behavior, the proposed solution should also be capable of dynamically altering generated behavior. The approach to this objective should be aligned with the individuality constraint by connecting it to an agent's personality.

RQ3: What is the role of perception and attention in behavior generation for cognitive agents?

According to Peters et al. [PCR+11], virtual agents require perceptual attention, and thereby virtual perception, for aesthetic and functional reasons. The former to increase naturalness of behavior and the latter to prevent performance issues by reducing the amount of information to process. In accordance with these requirements, how can perception be integrated into the proposed cognitive architecture to contribute to plausible behavior generation and at the same time fulfill real-time constraints? Investigating this RQ requires considering the plausibility and scalability requirements. However, perception-based agent behavior must not only be plausible and computable in real-time, but it must also be controllable to support the application's intention. Therefore, a virtual perception component must also facilitate authoring agent-centric, goal-driven behavior without compromising the other requirements.

RQ4: How well do the proposed concepts scale in terms of real-time, multi-agent applications?

Human-level cognitive capabilities are difficult to realize on conventional computer hardware, even without focusing on efficiency [DOP08]. But is it possible to realize a cognitive architecture approach applicable to interactive applications without compromising plausibility, reducing scale, or neglecting domain-specific requirements? As the proposed concepts are meant for real-time, multi-agent scenarios, this research

question is concerned with scalability. To achieve this objective, a realization of the developed concepts should not be limited by the number of simultaneously simulated agents.

1.4 Research Approach and Scientific Contributions

To answer the proposed research questions, the research is divided into two main stages: a conceptual stage and an application stage. During the conceptual stage, the combination of related research domains is used to define the scientific gap regarding the thesis objectives. Based on the research and the gap, three designs for a cognitive agent architecture are derived, which differ in their level of detail. First, an overall architectural design for cognitive agents is developed. Second, specific details of the overall design are considered, namely personality, emotion, and perception. Although the contributions in this stage are of conceptual nature, they are implemented and evaluated on a practical level blending theory and application. Third, the application stage is used to demonstrate how the design developed during the conceptual stage is applied in practice. The main purpose is to verify that application-specific needs and real-time constraints were sufficiently considered during the first stage from a software engineering perspective.

As a result of the laid-out process and by answering the proposed research questions, the following contributions are made by this thesis: A **lightweight cognitive architecture** concept is put forth, which is intended for generalized usage in real-time applications. The design is focused on enhancing agents with cognitive capabilities and characteristics, with the specific intention of integrating these agents into real-time virtual environments. To this end, the necessary requirements towards an architecture, able to simulate agents in the desired way, are devised. Common elements of cognitive architecture and IVA/ECA research are identified, examined, and assessed for their usefulness towards the intended design. As a result, a blueprint for attentive cognitive agents is defined and elaborated in Chapter 3, providing a foundation for following research contributions. From the methodological architecture concepts, the Cognitive Attentive Agents for Real-time Virtual Environments (CA^2RVE) architecture is derived and applied to a realistic evaluation scenario.

By integrating a personality at the core of the architecture, agent behavior is individualized while being consistent with an agent's past actions. The intention is to avoid the impression of randomized behavior, making agents more credible. In Chapter 4, a formal model of a **generic trait-based representation of personality** is described. To realize the model within an applied agent architecture, a specific personality model must be selected. Here, the common Five Factor model [MC08] is used as an example. Furthermore, a mapping

is defined that allows **transferring results from psychology studies** onto the architecture independent of the personality inventory[1] used within the study. The concept of task specific parameters is introduced, which determine how an agent's profile is utilized to influence its behavior. In combination with the mapping algorithm, task parameters can be used to relate agent behavior to personality both intuitively and based on findings reported in psychology literature. The personality profile is combined with a dynamic emotional state for situational behavior generation. By **combining personality and emotion**, a given decision-making approach can be enhanced, allowing an agent-individual and consistent, but dynamic action selection process.

To verify the initial claim of providing cognitive agents for real-time virtual environments, the proposed concepts are applied to an interactive, virtual user experience. A software architecture is designed and implemented based on CA^{2}RVE to realize and evaluate all the developed concepts. Using this software architecture, agents are used as traffic participants populating scenarios within a bicycle simulator for road-safety education of school children. The modular and extensible design of the architecture allows adjusting simulations to application-related constraints. Furthermore, the prototype implementation demonstrates the realization of the theoretical concepts in a possible production environment. By realizing an interactive experience using hundreds of CA^{2}RVE agents, the prototype also shows the **scalability** of the proposed solution. In this process, adjustments are introduced to improve scalability in the context of the utilized game engine. The reported concepts and examples have also been published in twenty-one peer-reviewed contributions listed in the Publications section at the end of this thesis (pp. 179).

1.5 Structure

The structure of this thesis is derived from the research approach described in the previous section. Chapter 2 provides an overview of the fundamentals underlying the work described in this thesis, with a specific focus on agents and related technologies, such as cognitive architectures and intelligent virtual agents. The related work is described according to the research domains involved in this thesis. Additionally, application related topics are discussed, e.g., traffic simulation and game engine technology.

In Chapter 3 the underlying architecture design is introduced together with the considerations that lead to specific design choices.

Following the overall design, Chapter 4 includes a more detailed view of the person-

[1] The inventory refers to the way of measuring personality traits in a specific study (i.e., questionnaire types), not the model itself. The personality model used in considered studies must be identical.

ality aspect of the architecture. It is described how personality profiles are defined within the architecture and how they are used in conjunction with emotional states to provide individualized yet adaptive behavior patterns.

Concluding the conceptual part, a model for human-like perception that fits into the overall approach is introduced in Chapter 5. The integrated components and their purpose are explained, with a specific focus on modeling attention.

A realization of the architecture design is detailed in Chapter 6 to provide a proof of concept that the theoretical design is applicable to a practical problem. Additionally, several approaches are presented that address inherent limitations of the provided design including scalability and maintainability.

All concepts presented in Chapters 4 and 5 were implemented using the software architecture described in Chapter 6. The evaluation approaches related to these concepts are recapitulated in Chapter 7, where they are also related to the proposed research questions evaluating the legitimacy and quality of the answers given to them. Further thoughts on an overall evaluation strategy and work conducted towards this approach are discussed as well. Finally, Chapter 8 includes a summary of the work and contributions mentioned throughout this thesis and an outlook on potential future research.

2

Related Work

"Success depends upon previous preparation, and without such preparation there is sure to be failure."

- Confucius

Before explaining the details of the proposed approach to attentive cognitive agents for real-time virtual environments, the required background information and related work are discussed in this chapter. First, in Section 2.1, cognitive architectures are discussed as they are the structural basis for the developed agent architecture concept. After introducing selected aspects of virtual humans in Section 2.2 (regarding the fields IVA/ECA and game AI), specific focus is directed towards enhancing virtual humans with personality and emotion in Sections 2.2.1 and 2.2.2. This enhancement is the most relevant topic about virtual humans as these elements are the central aspect of the proposed architecture. Besides the underlying personality concept, synthetic agent perception is the cognitive module that is examined in most detail within this thesis project. Therefore, Section 2.2.3 provides information on this topic. Finally, one of the objectives of this thesis project is investigating the applicability of the proposed framework to real-time virtual environments. As described in Chapter 6, a traffic simulation is chosen as a proof of concept for such an application. Therefore, the last section of this chapter provides information about traffic simulation systems as well as considerations concerning traffic simulation systems in digital games.

2.1 Cognitive Architectures

The general goal of developing a cognitive architecture is to devise a framework that allows autonomous (software) agents to solve diverse and unknown problems. The main intention

is to achieve agents with human-level intelligence, but also to gain insights into the cognitive processes of humans. The need for cognitive architectures was put forth by Allen Newell during the 1970's. His proposal was based on the observation that research about cognitive processes concentrated on highly isolated theories of specific behaviors or processes (cf. [Byr07]). From his idea of "unified theories of cognition" systems emerged that attempt to provide a computational framework modeled after the human mind and capable of solving universal problems. Today, numerous cognitive architectures exist and are continually being improved and extended. Although they all share similar goals, they can be coarsely divided into three categories: symbolic, emergent, and hybrid architectures [DOP08].

Symbolic architectures are most commonly associated with cognitive architecture research; well-known examples are Soar, EPIC, and ICARUS. As the name suggests, they represent the external environment and reason about it using symbols. Thus, the assumption of such systems is that the internal representation is isomorphic to the external one. Perception is the process that abstracts from the external world to the symbolic representation based on sensory data. The symbol tokens are usually described by human designers or programmers with the advantage that humans can directly interpret these systems. At the same time, the human-made representations influence the system's mode of operation and may restrict its potential. Due to their analytical approach and the methods employed (machine learning, probabilistic modeling, etc.), symbolic architectures have a strong connection to the field of classic artificial intelligence. Different representations of memory are modeled as either rules or graphs, while knowledge is inferred by either analytical or inductive learning [Byr07, DOP08, LL07].

Emergent architectures take a vastly different approach with most of them trying to model actual brain structures. Instead of symbolizing the environment of an agent, their internals are based on a network of distributed processing elements. Self-organizing and associating properties are achieved through interactions between these elements. Examples of emergent architectures are IBCA, Cortronics, NuPIC, and NOMAD. While these systems used to be less mature (cf., [DOP08]), symbolic architectures are being replaced by or augmented with elements from emergent approaches due to their flexibility and simpler design [KT18]. Information is processed on a global scale in such a way that all elements in the network influence the output or, when *localist* memory organization is used, only a subset of elements determine the result based on the input. Learning is achieved through training, i.e., the network either learns how to map certain input representations to appropriate output representations in a supervised or reinforcement learning fashion, or it learns unsupervised in a competitive fashion [DOP08]. However, since knowledge is represented by the entire

network and not represented as symbolic entities, traditional logical inference in emergent systems is difficult and and they become hard to read and understand [KT18].

Hybrid architectures combine the characteristics of symbolic and emergent architectures to benefit from the respective advantages of both paradigms. A well-known architecture of this kind is ACT-R, which utilizes a symbolic production system that is guided by a sub-symbolic layer of parallel processes [RTO19]. Other examples of hybrid systems include CLArION, LIDA, DUAL, Polyscheme, 4CAPS, and Shruti [DOP08].

Although different cognitive architectures were developed with a common goal, all focus on different aspects. However, as some of them continue to grow and mature, they add functionalities formerly exclusive to other cognitive architectures [Byr07]. Thus, it seems apparent that certain representations better fit certain problems or tasks. For example, semantic learning and semantic memory are central aspects of ACT-R but were also included in Soar to store and retrieve declarative knowledge more efficiently [Lai08, Lai12].

In the context of this thesis project, the Soar cognitive architecture [Lai12] is the most interesting of the examples mentioned above as it has been used to implement human-like agents in real-time virtual environments [LJ98, Lai01]. However, these examples require extensive rule declarations and processing resources. These requirements make it difficult to impossible to utilize Soar in limited hardware environments or in multi-agent applications. The traditional Soar architecture has been extended by multiple concepts. The most interesting concepts are knowledge representation, reinforcement learning, emotion, clustering, visual imagery as well as semantic and episodic memory. Long-term knowledge in Soar is comprised of production rules which can propose, evaluate, or apply operators (i.e., actions) providing for a flexible representation of procedural knowledge. A recursive processing of rules even allows for meta-reasoning within the framework. Reinforcement learning is used to fine tune the production rules and the integration of emotion, mood, and feeling enables a speed-up of the learning process. The concept of semantic learning borrowed from the ACT-R architecture enables encoding declarative facts facilitating the use of general knowledge about the environment. Episodic memory was implemented by storing snapshots of working memory for later retrieval of past experiences, which could be used for capabilities like internal simulation, prediction, episodic learning, and retrospective reasoning. Visual imagery is used as an alternative representation of the current state more suitable for tasks requiring visual-spatial learning [LL07]. A very intriguing addition is clustering, which enables Soar to create new symbolic structures overcoming one of the major shortcomings of symbolic architectures, as the encoding of symbols by a human designer is no longer restricting the system. To achieve this feature, experiences are clustered to identify statistical regularities, which are then mapped to new symbols [Lai08].

ICARUS [LC06] is another interesting symbolic architecture, because it is specifically designed to be used in physical agents. The architecture shares several features with Soar, but adds hierarchical representations of long-term memory, plus additional memories to support diverse types of knowledge. Furthermore, it defines skills to achieve goals, which are stored in prioritized fashion in separate memory. Input abstraction is handled by creating beliefs, which are compared to the skill hierarchy to find and select associated actions that advances the agent towards its current goal. What makes this architecture interesting to this thesis is its successful application to video games and even an in-city driving scenario. The architecture also deviates from the goal of providing a solution for general problems and instead focuses on emulating human behavior as authentic as possible [CKN+07, LC06, LSC+09]. However, ICARUS only enabled agents to learn how to perform basic driving tasks, like aligning themselves in a lane, accelerating, or decelerating for turns.

As an emergent architecture, Cortronics [DOP08] emulates the neuronal structure of the brain. By organizing these neurons into groups, which in turn form lexicons, the Cortronics architecture achieves a process called confabulation that enables learning and information retrieval. While confabulation is not suited for reasoning with complex knowledge, it does play a role in anticipation, imagination, and creativity.

ACT-R is a hybrid architecture utilizing symbolic and sub-symbolic representations. It combines the opportunity of allowing a human designer to describe the system's behavior with the self-organizing capabilities of connectionist structures, allowing autonomous fine tuning of the system. Knowledge in ACT-R can be either declarative, describing what an object is, or procedural, which encodes how things are done. Just as Soar and ICARUS, ACT-R has been applied to video games and related simulations, for example, to model tactical fighter pilots [CDB+02] or agents in the "Lemonade game" [RJSL10]. For ACT-R, the same constraints apply as those mentioned in correlation with Soar, which means it is not suited for the intended use cases of this thesis project.

2.2 Intelligent Virtual Humans

As indicated in the previous section, attempts have been made to utilize cognitive architectures and related technologies to create more realistic, human-like virtual characters. However, these applications appear to be a side-effect of the focused research on general problem solving. Others have recognized the potential of emulating human cognitive processes in virtual agents to generate more human-like and more believable behavior. In 2000, Rickel and Johnson [RJ00] used Soar to integrate the virtual tutor Steve into a virtual environment for military training exercises. For Steve's cognition module, Soar had

to be extended with a layer to handle collaboration capabilities and a layer containing the domain-specific knowledge that is taught to students. In addition to these intricate extensions, two modules needed to be developed for perception and motor control, all of which are connected by a message queue system. Overall, Steve is a complex application slowed down by several abstraction layers. To build a Military Operations in Urban Terrain (MOUT) simulation, Best and Lebiere built cognitive agents based on ACT-R to integrate aspects of human performance [BL06]. Agents were realized as virtual soldiers in Unreal Tournament 2004 (UT2004)[1] and as a robot in a physical test environment to demonstrate the use of the same architecture in different domains. While they succeeded with both embodiments, their system is complex and distributed across multiple machines for the Unreal Tournament scenario (one for each agent and one for the environment). Furthermore, adapting the system to a scenario requires sophisticated domain knowledge and scenario-specific extensions. To reproduce well-defined processes (e.g., clearing an L-shaped hallway) an extensive set of production rules is required for each behavior, which is reminiscent of complex scripting. Arrabales et al. also acknowledged the need for a flexible cognitive architecture for agents in a virtual environment [ALS09, AML+12]. Separating their implementation into a generic bot control architecture and a cognitive control architecture, the idea was to provide a high-level behavior controller that can be reused in different virtual environments. In contrast to other approaches, Arrabales et al. based their CERA-CRANIUM cognitive architecture on machine consciousness (MC) research, specifically on Global Workspace Theory and the Multiple Draft Model. They demonstrated their approach in a case study by developing bots for UT2004 using CERA-CRANIUM. However, applying the cognitive control architecture to the UT2004 environment required two additional third-party frameworks (GameBot and Pogamut2) and scenario-specific adaptations to almost all CERA-CRANIUM components.

In 2016, Smart et al. [SSSS16] claimed to be the first to integrate ACT-R into the commonly used game engine Unity[2]. While being an admirable effort, they faced various technical difficulties and confirmed the performance issue of running both a game engine and a cognitive architecture on the same machine. The integration also required extending ACT-R with a custom module, developing an ACT-R API for network communication, and a third-party network interface, which also needed to be extended. Despite all this effort, the final experimental setup showed only inefficient agent behavior and could not utilize numerous ACT-R core modules as they are not meant for virtual characters. Recently, Llobera and Boulic developed an agent architecture for the Unity game engine [LB19]. Instead of integrating a complex cognitive architecture, they describe a user-centric design based on

[1] Epic Games, 2004

[2] `https://unity.com`, [online: May 2, 2023] Unity Technologies

embodied cognition theory. This approach is promising, especially since it emphasizes modularity, scalability, and stability while being tightly integrated into a game engine. However, the solution requires a manual and laborious process for each individual agent to link the agents' skills to their goals.

Another approach is providing a middleware that allows interfacing a virtual environment system with an agent architecture. Two prominent examples are POGAMUT and CIGA. The POGAMUT framework was introduced by Gemrot et al. [GKB+09, GBK+10] as a mediation layer between an agent's body, situated in a virtual environment, and its mind. The framework includes a simulator of the virtual world, a graphical IDE, a library of base classes for agent behavior, a service handling communication between the VE and the agent architecture, and tools for defining and running experiments. Gemrot et al. motivate their work with the difficulty and steep learning curves of defining autonomous behavior for virtual characters and the lack of commonly available software tools supporting this task. While POGAMUT 2 was tightly coupled to UT2004 as its VE, the major improvement of the third version of the framework was to achieve a looser coupling. However, from their description of the required adjustments to the core systems (perception, interaction, and high-level API), integrating a virtual environment other than UT2004 or a different agent framework remains a complex and time-consuming task. Interesting extensions to the framework were planned (e.g., ACT-R integration, emotion modeling, connection to other VEs), but according to the publication record, it seems they have never been completed. Furthermore, the framework is limited to simulating ten virtual characters simultaneously.

Van Oijen and his colleagues are also looking to make intelligent agent technology more accessible, but with the additional intent of making it available to the games industry [vOD11, vOVD12, vO14]. Like POGAMUT, van Oijen et al. distinguish specifically between an agent's embodiment (the game engine/virtual environment), its mind (the agent system or cognitive domain), and the interface between them (the middleware). While their CIGA platform demonstrates possible realizations of these three elements, the research objective was to provide a theoretical overview of a middleware's role, design issues, and guidelines when integrating intelligent agents into virtual worlds. Therefore, they also present a general-purpose interface to facilitate a connection between an agent system and a virtual environment independent of specific frameworks, technologies, or application domains.

Taking a middleware approach, like van Oijen et al. or Gemrot et al., seems like an obvious choice. If done carefully, the middleware is independent of specific systems for an agent's body and mind. This aspect is especially interesting since available game engines and agent frameworks are usually implemented using incompatible programming languages. In theory, this open-system approach also means any agent behavior framework

can be connected, allowing a wide spectrum of behavioral complexity. Unfortunately, the abstraction required to facilitate such an open system comes at the cost of reduced performance, as information must pass through several interfaces and layers typically involving multiple data conversions and often network connections. Especially in sectors like the games industry, where established programming techniques are ignored in favor of micro-optimized code bases (see, e.g., [Act14]), this fact plays a significant role. Loosely coupling embodiment and cognitive functions also means that the middleware must manage issues like asynchronous inter-process and network communication as well as synchronization of internal states, data, and life cycles. One of the most prominent objectives of middleware is to make creating human-like agents more accessible. However, it may be argued that the contrary is true since behavior designers must be proficient in both VE/games technology and agent technology. Both areas are complex, share little similarities, are often realized using different technologies, and provide distinctive design tools or UI. Furthermore, the advantage of a general-purpose approach often fades away as soon as it is realized and must efficiently interface with concrete systems on both ends. Judging from several reports (e.g., [ALS09, AML+12, BL06, GKB+09, GBK+10, SSSS16, vO14]), a significant effort is required to connect independent virtual environments and agent systems. A middleware may support this effort but cannot completely cover the gap between the systems [vO14], which further diminishes the advantages of a general-purpose approach. Finally, being able to interchangeably use different virtual environments and agent systems is a valuable characteristic in fields like research and education. However, this interchangeability is an unlikely requirement in a production environment where integrated systems are rarely replaced, especially in the games industry. Instead of striving for a middleware approach, the objective of this thesis project is to present a blueprint for cognitive agents, in case you want to realize your own architecture based on it, but also to provide a solution integrated into a game engine that is one of the games industry's standards.

2.2.1 Personality

All the above are potentially interesting techniques for developing more human-like virtual characters, which are an essential element of various applications, e.g., human-computer interaction, entertainment, education, and digital storytelling (e.g., [AKG+00, KHW+11, LTC+01, RP12, SFC+10]). Often users interact with these entities over a prolonged period, in some cases across multiple sessions. According to Ortony, one of the most important aspects to making agents believable to users is by conveying consistent behavior based on personality and emotions expressed through gestures, facial expressions, and decisions they make [Ort03]. Defining personality profiles requires mapping an individual and complex

aspect of human beings to a simplified and quantifiable model. It is important to note that this aspect is temporally stable and not a momentary mental state of a human. During the 19th century, multiple models for representing personality have been developed, validated, discussed, and sometimes disproved. One of the oldest models for dimensioning personality are Cattell's *16 Personality Factors* (16PF) [Cat46, Cat57]. His research was based on a list compiled by Gordon Allport, which included 4000 English words that can each be used to describe a person. Cattell analyzed and condensed the list to 171 terms, which he then reduced to the 16 factors using statistical means.

In contrast to this extensive model, the model introduced by Eysenck initially only included two dimensions: *Neuroticism* and *Extraversion/Introversion* [Eys47]. These two dimensions – sometimes referred to as the "Big Two" – are included in many models of personality to this day. Later, Eysenck added *Psychoticism* as an additional dimension to his model (cf. [Dig90, Eys70]).

From a review of Cattell's work by Fiske, the development of the *Five Factor Model* (FFM) began in 1949 [Fis49]. At this time not more than five personality dimensions could be confirmed. While Fiske's work seemed to have negligible impact on the scientific community at first, his results were verified in 1961 by a review of several conducted studies by Tupes and Christal [TC92]. Although the validity of the FFM was reinforced by other researchers (e.g., [Bor64, Nor63, Smi67]); it was not until the 1980s that the model gained popularity in the field of personality psychology through the publication of numerous reviews and studies (cf. [MJ92]). Another reason for the model's increased popularity is its consistency across different observations, self-reports, interviews, and languages [SGW12]. While there are different labels for each of the five factors, the labels from the Five-Factor-Inventory (FFI) questionnaire and its updated version, the NEO-FFI, are most used. The FFM's descriptive nature and the fact that only five personality traits are sufficient to define a personality are likely responsible for its popularity when it comes to modeling personality for virtual humans [AKG+00, KMT08]. Table 2.1 summarizes each label including an exemplary interpretation of high and low values for each dimension. Since different scales exist for measuring each trait, even within one model, e.g., the NEO-FFI [CM92] and the Big Five Questionnaire [CBBP93] for the FFM, the development of a generic model for computational use of personality profiles is motivated in Chapter 4. Furthermore, it is demonstrated, how findings from psychological studies can be utilized in the agent architecture using the FFM, which is chosen due to its mentioned advantages.

To understand why people take certain actions and make specific decisions, several studies were performed to find correlations between their personality and performance in specific tasks (e.g., [MDWS00], [Her09], [LL01]). Because of differences in the cognitive

Table 2.1: Dimensions of the Five Factor Model including exemplary interpretations of high and low scores (cf. [MDWS00], table from [IVC13]).

Label	Abbr.	High	Low
Openness	O	creative, artistic, multilayered	practical, simple, superficial
Conscientiousness	C	methodical, efficient, thorough	unorganized, lazy, incautious
Extraversion	E	social, optimistic, impulsive	introverted, quiet, reserved
Agreeableness	A	nice, cooperative, likable	indifferent, rough, dislikable
Neuroticism	N	nervous, tense, emotionally volatile	calm, satisfied, relaxed

processes, the way personality influences the performance of a person is highly dependent on the specific task [MDWS00]. However, by integrating a formalized model of personality profiles into the decision-making processes of an agent, they can be utilized to model more plausible and realistic behavior. For example, Rushforth et al. extended an existing virtual human architecture with a personality model to increase the believability of virtual interrogatees in a tactical questioning system [RGA+09]. Their evaluation included a personality model parameterized by ten personality traits, which are derived from the FFM. The results showed that personality difference could be perceived by test subjects influencing potential outcomes of the interrogation scenario. Bevacqua et al. investigated whether an agent's personality profile could not only generate perceivable differences in behavior, but also reflect the agent's "actual" personality [BdSP+10]. They extended Sensitive Artificial Listener (SAL) agents with Eysenck's three-factor model to determine the agent's preferred means of non-verbal communication. However, their agents only display backchannel behavior, i.e., the agents express their reaction to the user's input, showing if they are listening, agreeing, etc. Unfortunately, they did not deploy the system in an evaluation study. Castillo et al. addressed this shortcoming, who used a similar setup to confirm that human personality questionnaires can be used to measure the perception of agent personality [CHLC18]. They also found that the perception of the exact personality depends on the availability of the communication channel. According to their results, the perception and design of virtual personalities requires multiple channels that are consistent among themselves. Zhou et al. went even further and examined the role of agent personality on a user's trust in the agent [ZMLY19]. Using a chat-based agent, they conducted job interviews in high-stakes and low-stakes scenarios. The agent cannot only have its own personality, based on the FFM, it is also able to infer the interviewee's FFM personality based on chat messages. Based on their interview analysis, they found that the agent's personality affects how much interviewees trust that agent and how willing they share information with the agent based on that trust. Interestingly, they noted that by inferring a user's personality, an agent could adapt

its own personality to maximize trust according to their findings. Zhou et al., like many others used the FFM to model agent personality. However, Völkel et al. argue that human personality models, like the FFM, are not suitable to describe personalities of conversational agents [VSB+20]. Thus, they propose their own personality model based on a psycholexical approach, consisting of ten traits that are not consistent with the FFM. However, their argument directly contradicts statements by other researchers, e.g., Castillo et al. [CHLC18], and is specifically aimed at speech-based virtual assistants like Alexa, Siri, or Google Assistant. Therefore, this result should be considered with care. Knob et al. presented another application of agents with personalities, simulating heterogeneous crowds based on the FFM [KBM18]. Personality traits are used to control behavior regarding group formation during evacuation scenarios. Their evaluation showed that with personalities, the intended results could be observed.

2.2.2 Emotion

In contrast to the temporally stable personality, emotions are dynamic mental states, which can change over short periods of time. Colloquially, the term emotion is often used interchangeably with the term feelings. In the context of this thesis project, emotions are defined as the sensation and perception of feelings resulting in a short-lived change in mental state. Within this model, the change in state is always linked to a direct cause, which separates the nature of emotion from mood. Among other things, mood can often not be clearly linked to a cause. Instead, it is the result of complex cognitive processes, influenced by various factors and often not completely comprehensible. A further distinction between emotion and mood is their temporal effect. A mood can last for a prolonged period of time, while emotions are typically immediate and short-lived. However, despite the separation of both states, their linguistic concepts are not independent of each other (cf. [Tha89]).

Like personality, many approaches for modeling, categorizing, and dimensioning human emotion exist. A method for objectively and accurately measuring human emotion has yet to be discovered. Therefore, models of emotion are typically identified by the name of the method used to survey them. One of the core discussions is the question whether the dimensions identified as part of a model are interdependent [Tha89]. Some studies suggest a mutual dependency, others indicate positive correlations between dimensions of the model, and yet others state that dimensions are independent. Further studies show that weak emotions are independent or positively influence each other, while strong emotions negatively influence each other. The number of dimensions is also a subject of debate. For example, the Eight State Questionnaire (8SQ) samples eight dimensions [CC76], while for other models, like the Activation-Deactivation Adjective Check List (AD ACL) [Tha86], two

dimensions are sufficient. Thayer points to the different ways of data acquisition as the cause for these different results. Especially the combination of subjects' self-reporting, non-standardized questionnaires that are tailored toward an individual model, and changing analysis methods are problematic. Thayer mentions one example where an analysis of the AD ACL confirmed two dimensions, however, by adding further items to the questionnaire, a third dimension could be extracted from the reported answers [Tha89].

For modeling emotions of virtual characters, the OCC Model of Emotion [OCC88, Meh96] is popular among researchers in this field [KMT08, NFdSS10]. The model includes 22 types of emotions that can be either positive or negative. However, its scale makes the model complex for integrating them into embodied characters [Bar02]. Depending on the application, alternatives for representing emotions can be used. For example, Curran's 8SQ requires only eight dimensions [CC76] and the PANAS model reduces this number further by only considering positive and negative affect [WCT88]. The agent architecture proposed in this thesis specifies neither the type of emotion model nor the number of emotion dimensions. However, an integral aspect of this new architecture is the combination of the emotion model with an agent's personality. Therefore, for the application-driven architecture, a two-dimensional model based on AD ACL and PANAS is suggested to keep the combinatorics of dimension interactions maintainable.

Few examples in the literature consider both personality and emotion. One of them is Gebhard's layered model of affect (ALMA) that defines mood states in PAD (Pleasure, Arousal, Dominance) space [Geb05]. Each octant of PAD space is a discrete mood description, which is further categorized by its strength depending on the distance of the current mood state from the origin. An agent's default mood is defined by a FFM trait configuration. The emotion model generates emotions according to the OCC and each type of emotion is mapped to PAD space. The intensities of the emotions are controlled by an agent's FFM personality. From all active emotions, an emotion center in PAD space is calculated that is used to gradually change the agent's current mood. While the agent's action selection is scripted, different variations of the action can be provided for each discrete mood state. Consequently, the mood state determines an agent's displayed behavior, e.g., by affecting the selection of dialog options and strategies, idle gestures, or facial expressions.

Johansson and Dell'Acqua criticize that AI behavior in commercial games is predictable and invariant [JD12]. They also argue that random action selection does not solve the problem, because the resulting behavior is inexplicable to an observer. Therefore, they extend behavior trees, which are a common tool for authoring character AI in the games industry, with a new priority selector node. This node type considers a combination of a character's emotional state and a set of correlated factors (time, risk, planning) to decide which child

node to select. Due to the difficulty of measuring emotion, they define correlations based on intuition and advise users of their system to choose emotions that are consistent with the modeled character. Belle et al. extend the idea by Johansson and Dell'Acqua and combine it with Gebhard's ALMA to develop a lightweight cognitive architecture [BGG19]. Similar to the work presented here, their objective is to address the amount of computational resources required by conventional cognitive architectures while being able to simulate affect with psychological accuracy. Their *emotion adder* node can be integrated into behavior trees and induces the specified emotion in an NPC. The NPCs current emotional state affects its mood state (a position in PAD-space). An *e-selector* node uses the current mood state to choose between its child nodes. Although there are eight different mood states, an e-selector node can only have two children which represent either a positive or negative action. Furthermore, emotional responses can also be triggered by events stored in memory. Unfortunately, they applied their system only to dialog selection and there seems to be no direct connection between a character's personality, emotional state, or mood state, and the action selection. Another positive example is the game Watch Dogs 2[3], where the developers equipped agents with an emotional state and personality traits to diversify their behavior [BP17]. Decision-making is mostly reaction-based using fuzzy rules to choose between multiple behavior options based on emotion and personality. An additional mood state machine avoids erratic changes in emotion. While the deployed system is not complex and not based on psychological models, it emphasizes the role of personality and emotions in plausible behavior generation for virtual characters.

2.2.3 Synthetic Perception

Perception can be regarded as the input stage to an agent's action selection processes [PCR+11] and as such represents the foundation of plausible behavior (cf. [Blu97, POS03, Rey87]). Therefore, perception processes for agents have been addressed in various research fields. The most relevant fields for the work presented here are computer vision, intelligent virtual agents (IVA), embodied conversational agents (ECA), and game development. Within these fields, perception processes are often mentioned, but rarely are they the focus.

Modeling perception, specifically attention, in a biologically plausible way has been an important subject of computer vision research for decades. Suitable models can aid in detecting interesting image regions as well as in segmenting and tracking relevant objects. For example, Tsotsos et al. presented an approach for feature selection in images by activating specific units within an image processing pyramid [BT09, TCW+95]. Units are activated by a process called *selective tuning*, which requires several traversals of the pyramid. The result

[3] Ubisoft, 2016

is an attentional beam through the pyramid. Feature selection is simulated by including a winner-take-all (WTA) process on each level of the pyramid, as is the case in many computational models of biologically plausible cortical selection processes, like visual attention [KT18]. As an example, the work of Bruce and Tsotsos was strongly influenced by the models presented by Koch and Ullman [KU87] and Olshausen et al. [OAVE93]. Other examples are those reported by Bruce [BT09] and Pomplun [Pom06]. Some research has also been performed to apply these concepts to more practical applications (e.g., [Fri06, HKRS95]). The foundation of most computer vision approaches are intricate image processing algorithms, which are often not applicable to real-time virtual environments due to their complexity (cf. [BA13, RPA+15]). Additionally, theses algorithms require images to process, which need to be synthesized in such scenarios, which increases computational costs even further.

In virtual reality and agent research, perception and attention models have also gained significance [PCR+11]. In this field, plausibility takes precedence over accuracy in a neurobiological sense. Although, agents generally do not need to differentiate between the virtual environment and their model of the environment, perception capabilities of simulated entities are often limited, mimicking their real-life counterparts, to generate believable behavior [LA00]. While any input received by an agent could be defined as a *percept* [RN10], it makes sense to distinguish between stimuli and percepts. Stimuli are information collected by virtual sensors from the agent's environment. Percepts are stimuli that an agent has become aware of, i.e., the agent must have focused its attention on the stimulus (cf. [POS03]). This distinction will be revisited in Chapter 5.

Additionally, solutions within the area of virtual human research must fulfill real-time requirements to allow for interactive experiences, limiting the complexity of applied approaches (e.g., [KW15]). Most research in this area focuses on simulating eye gaze as it represents a suitable external indication of attention (cf. [APMG12, BA13, PAGM15]). For example, Kim models attention for virtual humans to decide when to change gaze direction alongside its attention to another object [Kim06]. A model by Andrist et al. considers physiological characteristics of gaze shifts to convey the presence of attention in a virtual character to improve users' affiliation with the character and learning. While simulating eye gaze is an interesting aspect requiring virtual perception, it remains only a specific subtopic.

Like human perception, virtual attention models can also be used to reduce or filter the complexity of a scene saving computational and memory resources within the perception process. At the same time, such an approach can create the impression of an overlooking or unobservant agent, creating a more realistic representation [BA13]. Typically, solutions generate *saliency maps* to identify regions or objects that attract an agent's attention [IDP03]. In virtual environment applications, these techniques require rendering a virtual camera

image, usually in reduced resolution, for every simulated agent (e.g., [BSFL+12, POS03]). Many approaches even require several feature maps of the rendered image. Itti et al., for example, generate 72 feature maps from pre-processed video frames in combination with a task-relevance map for top-down attention and a winner-take-all approach to guide a virtual agent's gaze [IDP03]. While abiding to neurobiological models of visual attention, such an approach does not scale for real-time multi-agent scenarios. Furthermore, the applied attention models are not considered on a cognitive level in most cases [CKH+15]. However, the amount, relevance, and type of available information is an integral part of an agent's action selection process [PCR+11]. Among others, agents can choose physical actions within their environment (e.g., moving to a position from which more appropriate information can be obtained) or "internal" actions like attention direction or primary sense orientation. On a technical level, several approaches address attention processes within larger agent frameworks, e.g., filters [BBT99] or subscriptions [vOD11]. In these examples, versatile middleware approaches are presented, but the actual implementation of specific attention algorithms or strategies are not addressed. Instead, they are left to a designer.

The idea of limiting sensor capabilities of agents to simulate perception is a more common approach than considering actual attention models. In game development, it is mostly concerned with detecting and reacting to players (e.g., [Leo03]). However, while game AI programmers try to create plausible behavior, the most important aspect is always to provide a fun game experience. Players must never feel cheated by their in-game opponents; thus, agent behavior must always be obvious to the players. The agent's perception processes need to be designed accordingly. As a result, behavior generation is typically solved with diverging approaches in game AI and virtual humans. One objective of this thesis project is to bring both areas closer together by providing a controllable yet versatile solution.

Virtual human researchers are not constrained by the goal of providing fun experiences and are able to explore more realistic perception models within their work. In general, IVAs are equipped with a set of virtual sensors often representing actual sensory organs of living beings. Due to the importance of the visual channel in human perception processes, simulating visual sensation is the most frequent approach to generate more authentic behavior for simulated entities. However, a few examples exist that deal with the perception of stimuli from multiple sensor modalities, e.g., [BA13, KW13, KvVH05]. The simulation of the visual sense is most commonly implemented in one of two approaches: geometric algorithms or synthetic vision [PCR+11]. Geometric approaches utilize techniques such as ray-casts or intersection tests to check objects for visibility. A common technique is to evaluate whether objects are placed within one or more view cones (e.g., [KW11, Leo03]). Synthetic vision approaches render a low-resolution image of the environment from each agent's point of view

to determine which object is visible to an agent. The advantage of rendering the scene is that object occlusion is provided without additional effort. However, the challenge is extracting relevant information from the rendered scene view. Noser et al. [NRTMT95] simplified the image analysis by rendering objects with a unique color while neglecting textures and lighting. As a result of this *false coloring*, an object is visible if at least one pixel of the rendered image contains the object's unique color. Later, other researchers extended this approach to encode more information or improve performance (e.g., [CT06, PO02, SL05, OPOD10]). Sample implementations of both approaches are used to realize the concepts developed within this thesis project (see Chapter 6).

A sensor interface is often designed to provide a unified entry point to access and aggregate data provided by different sensors. The resulting object representations are typically stored in a hierarchical memory structure (cf. [CT06, KW13]). Peters and O'Sullivan [PO02] provided a now well-known memory hierarchy based on a psychological model proposed by Atkinson and Shiffrin. Within the model, entries are swapped between a short-term sensory storage, short-term memory, and long-term memory. The more attention an object receives, the higher they are placed in the hierarchy. An interpretation of the model by Peters and O'Sullivan is integrated into the perception framework presented in this thesis.

A common approach to improve efficiency of virtual perception processes is to include semantic information about virtual objects. By attaching semantics to objects in the environment, agents do not have to interpret sensor data to gain relevant knowledge, e.g., about object affordances. For an overview of semantic modeling see [TBSK08] or [vOVD12]. Within this thesis project, specific semantics are also defined in an application context to make perception and knowledge retrieval more efficient.

2.3 Traffic Simulation

Within this thesis project, the developed cognitive agent architecture concept is realized to simulate road traffic for a virtual bicycle simulator. Traffic simulations are employed in a variety of applications, e.g., road planning, traffic jam prediction, virtual environments, and digital games. Behavior of the simulated traffic participants differs in complexity and type, depending on the application. For example, in digital games that provide an open world for a player to explore, the traffic participants serve only as a backdrop to allow the player to experience a *living* environment. In such cases sufficient realism can be achieved through comparatively simple means, e.g., using finite state machines or scripts. These approaches are sufficient because agents, which are not relevant to the game's objectives, are not observed in detail over a prolonged period of time. In traffic simulations used for

road planning and similar tasks, only the emergent traffic behavior of the entire system is of interest, e.g., to reveal deficiencies of a given road network. Human traits, like reaction time or imprecision, are expressed in a mathematical form, which defines behavior. Especially car following models provide numerous approaches to plausibly convey tasks like acceleration, approaching an obstacle, or keeping the distance to a lead car (see [Bar10] and [TK10] for an overview). Nevertheless, these approaches usually do not try to replicate the cognitive processes involved while participating in traffic, and making individual, observable behavior believable to a user is not a requirement. Aspects, such as personality and emotion, are typically only implicit parts of different driving behavior achieved by varying parameters in microscopic traffic models. One prominent example is Wiedemann's psycho-physical car-following model (see, e.g., [FV10]). Eventually, the multitude of proven traffic models provides important starting points and inspirations for enhancing them with more human-like behavior on the agent level.

Traffic simulations are typically categorized into microscopic, macroscopic, and mesoscopic simulations [Bar10, HHST02]. Macroscopic traffic models do not consider individual vehicles, but rather model aggregated parameters (e.g., average velocity, traffic density). Involving a user as an active participant in traffic is not possible in macroscopic models, which means they are unfit for interactive virtual environments. Microscopic traffic models model each individual vehicle with its individual parameters (e.g., current velocity, desired velocity), especially its current position in traffic and its behavior regarding traffic rules. Since users can represent one of many individual vehicles, integrating them into a microscopic system is straightforward, but not a simple task.

Mesoscopic traffic models bridge the gap between the individual approach of microscopic modeling and the aggregated approach of macroscopic modeling. This results in select aspects, such as traffic flow, being modeled at the macroscopic level to decrease overall computational effort and to achieve good scalability. However, mesoscopic models also include microscopic concepts, which can be used to keep track of associated information, such as individual route choices or vehicle types [BKA05b].

Hybrid approaches combine microscopic and mesoscopic models to simulate specific areas of interest in a traffic network using microscopic modeling, while everything else is simulated using less detail using a mesoscopic model (cf. [BKA05b]). Such a hybrid approach is also discussed in Section 6.3 to handle real-time constraints of the realized cognitive agent application. Another example of a mesoscopic simulation is the traffic flow model for the POLARIS transportation systems simulator developed by de Souza et al. [dSVA19]. It successfully combines the level of detail and accuracy of microscopic models with the computational benefits of macroscopic models. Saprykin et al. developed

a simulation for generic large-scale networks that achieve high computational performance by taking advantage of parallel GPU computing [SCA19]. They based their simulation on MATSim[4], an open-source framework for large-scale agent-based transport simulations.

MATSim is based on queuing theory, which lends a different approach to traffic simulation. In queuing theory, packets travel through a network of service stations and queues. The packets stay in a queue until a service station serves them at certain rates. This approach can also be applied to vehicle traffic. Vehicles are modeled as packets, roads as queues, and intersections as serving stations [BZBP09, CBN03]. Vandaele et al. [VvWV00] simulated a highway using several queuing models. They used their simulation for traffic management, congestion control, and to determine the environmental impact of road traffic. Van Woensel et al. [vWV07] gave an overview of how queuing theory is used for traffic simulations. Furthermore, they suggest combining agent-based simulation and queuing theory to benefit from both approaches. One interesting model they refer to is the FastLane model by Gawron, which limits the number of agents that can reside on the same edge of a traffic graph at the same time [Gaw98b]. FastLane is the basis for MATSim [CBN03, SEN99].

Cetin et al. [CBN03] introduced a parallel implementation of FastLane. By parallelization and execution on a 64 CPU cluster, they can simulate 24 hours of traffic in less than two minutes. Additionally, they introduce a "fair" intersection, which divides the limited space of an outgoing road proportionally to the capacity of the incoming roads. Their research considers large scale transportation scenarios and not urban scenarios, which are of interest to the work presented in Section 6.3. Furthermore, the system realized within this thesis project cannot run on massively parallel systems but on consumer hardware. Grether et al. [GNN12] presented a combination of FastLane and the model by Cremer and Langenfeld [CL97]. This led to a more realistic representation of turning lanes at intersections, since the FastLane model neglects that lanes at intersections often branch into multiple lanes. Their focus was on interactions between agents and intersections and how they can be simulated efficiently. One of their main objectives was to simulate spillbacks with queuing models to realistically represent real-world traffic. In an interactive, virtual training scenario, it is more important to ensure a continuing traffic flow in the background.

Simulating traffic in virtual environments, like the FIVIS bicycle simulator, is mainly concerned with the user's immediate surroundings. Therefore, microscopic traffic models are the most relevant to the work presented here. Most microscopic traffic simulations are based on a combination of car-following and lane-changing models (cf. [Bar10, TK10]). During each step of the simulation, the model is updated by updating each entity representing a traffic participant based on parameters like gap distance to and velocity of the leading vehicle,

[4] `https://www.matsim.org/`, [online: May 2, 2023] Multi-Agent Transport Simulation

acceleration/deceleration capabilities, or response time delay. Due to the increased difficulty of calibration, few car-following models include additional parameters to individualize behavior, such as age, gender, risk-aversity, vehicle size, time of day, weather, hurrying, and fatigue (cf. [PD05]). Most available microscopic models were developed to predict essential traffic parameters for generic vehicle classes (cars and trucks). However, in training scenarios, like those intended for the FIVIS bicycle simulator, special traffic participants (e.g., public transport, delivery vehicles, pedestrians, garbage trucks) are especially interesting. Their individual behavior can disrupt regular traffic and lead to interesting situations that need to be handled by a learner and other simulated traffic participants.

The foundation for the traffic simulation, presented in this thesis, is the work by Kutz et al. [KH08, Kut09]. They conceptualized basic building blocks of a traffic simulation for virtual environments and games. Their description includes a graph structure, which represents the road network and provides perceptual information to the simulated agents. The use of individual personalities for each agent, based on the FFM, is also mentioned but the correlation between an agent's personality and its respective behavior in traffic had not been provided.

Performing traffic simulations is impossible without an underlying representation of the road network. Like traffic simulation approaches, models for road network representations can be divided into models for macroscopic, microscopic, and mesoscopic simulation. In general, the semantic representations are defined in a graph-like manner. While road network representations for macroscopic simulations are often represented by simple graphs without detail, road network representations for microscopic simulations must be very detailed, and the graphs are enhanced by many features such as relations to signs and signals or detailed descriptions of junction areas.

Although road network representation is essential for simulating traffic, the topic is rarely focused on in the literature. The main reason being that the representation is often strongly linked to the simulation system and type. Notable examples range from simple solutions used in digital racing games [Bia02] to memory optimized networks for open world games [Kra10, Kra12] to intricate definitions used in commercial traffic simulations [FV10]. Concepts from these examples are adapted for the semantic road network introduced in Section 6.2, with specific focus on realistic urban roads.

To efficiently generate road networks for traffic simulation, automatic or semi-automatic processes are preferable to manual setup. Three approaches for the setup of road networks can be distinguished. The first is to take existing data (e.g., from public databases), which ideally represent real road network structures, and automatically generate entire networks

taking standardized data as input. While there are many such standards (e.g., by ESRI Inc.[5], Thales[6], or [CTHG10]), the OpenDRIVE® standard [DSG10] was identified as the most versatile and flexible within the scope of this thesis project. Reasons for choosing OpenDRIVE® include, but are not limited to, being open, XML-based, well-documented, well-established, and actively developed. Furthermore, the standard is maintained by driving simulation experts as well as being easily transferable and extensible to custom systems [ASA21, DSG10]. The second approach is to create fictitious but realistic road networks using procedural techniques (e.g., [GPMG10, PM01, CEW+08]). The third approach is to deliver a set of tools for the intuitive manual setup of road networks. This approach can also be used to refine or improve networks generated by (semi-) automatic approaches. Gerdelan [Ger09] described a set of point-and-click editing tools for integrating road network representations into existing virtual environments. Approaches like [ALD11] or [MS09] provide environments where a user can draw lines, which are adapted by the system such that they are shaped like real world roads (i.e., shapes based on lines, arcs, and clothoid curves because straights and curves of real roads are connected by clothoids to avoid abrupt steering maneuvers).

A subset of requirements for a traffic simulation within the scope of this thesis project includes efficiently representing a road network, providing agents with knowledge about said network, and having agents interact with each other to provide an interesting environment for users. Additionally, all these requirements must be realized within an interactive system, like a bicycle simulator. Digital games, especially *open world* games, have similar requirements. In current examples, e.g., Marvel's Spider-Man[7] or the GTA series[8], impressive results are achieved considering the significant limitations of available processing and memory resources. Unfortunately, detailed information about existing game systems is difficult to impossible to acquire outside of the industry. Some developers share insights about their games, but traffic simulation is rarely a topic. The few existing examples, e.g., [Bia02, Kra10, Kra12], usually do not provide enough detail for re-implementation. Nevertheless, these and similar systems are looked to for inspiration throughout the realization of the traffic simulation described in this thesis.

[5] `https://www.esri.com/en-us/home`, [online: May 2, 2023]
[6] `https://www.thalesgroup.com/en`, [online: May 2, 2023]
[7] Sony Interactive Entertainment, 2018
[8] Rockstar Games/Take 2 Interactive, 1997 – 2015

3

Architecture Concept for Attentive Real-Time Cognitive Agents

"You can't build a great building on a weak foundation. You must have a solid foundation if you're going to have a strong superstructure."

- Gordon B. Hinckley

One of the major objectives of this thesis project is to provide means of authoring and simulating intelligent agents for real-time virtual environments, e.g., training simulators. To achieve this objective, the developed solution should fulfill the following requirements:

(1) The generated agent behavior is believable to an observer to preserve the effect of immersion.

(2) The generated agent behavior can be controlled by a domain expert or application designer to support the design goal of the application.

(3) The solution is scalable to enable multi-agent simulations in real-time.

The hypothesis underlying this work is that emulating human cognitive processes in agents generates more human-like and thus more believable behavior. In cognitive architecture research the same approach is taken to achieve agents with human-level intelligence. Thus, the obvious solution seems to be applying a cognitive architecture to virtual characters. In fact, multiple other researchers have investigated this approach with the same rationale (e.g., [ALS09, BGG19, BL06, BKA+05a, CDB+02, LB19, RJSL10, RJ00, SSSS16]). However, the number of examples found in the literature is moderate and the proposed solutions are neither coherent nor consecutive, suggesting that this approach remains difficult and does

not represent an established solution. Reported examples of cognitive architectures applied to real-time applications also typically employ network solutions with each cognitive agent running on a separate machine connected to a common virtual environment. This is due to the fact that cognitive architectures require substantial computational resources, making it difficult to run both the architecture and the virtual environment on the same machine [SSSS16, BGG19]. Other reasons for choosing a network integration are the lack of support for direct integration into external systems and the technological gap to tools used for creating and running virtual environments, e.g., game engines [SSSS16].

While a loose network integration is not a desired solution within the context of this thesis, the benefits of using an existing architecture could outweigh this constraint. However, by studying examples in the literature, it becomes apparent that authoring agent behavior using cognitive architectures is cumbersome and difficult for non-experts, due to their complexity, required production rules, and the level of expertise and experience required. Additionally, even a loose integration usually requires substantial effort. E.g., Smart et al. [SSSS16] report that integrating ACT-R into the Unity[1] game engine required developing a separate API, another research team's network module, and an ACT-R module consisting of 110 production rules and an unspecified number of ancillary functions to realize a straightforward scenario.

Since modeling the human thought process to achieve human-like behavior remains a promising approach, it is desirable to design an architecture that is based on the same principles as cognitive architectures, but in a less resource-demanding fashion: a *constrained* or *lightweight* cognitive architecture. In such an architecture, modeled processes are abstracted to a degree at which they are applicable to real-time solutions. The major challenge is finding a suitable balance between the level of abstraction, plausibility of generated behavior, and required effort for authoring behavior. Since agents built from the developed design should support application-specific goals, such as training a user, challenging a player, or entertaining a consumer, the resulting architecture must be flexible, allowing developers to steer behavior generation in the appropriate directions. Summarizing, the resulting design should provide a lightweight cognitive architecture serving as a basis for building software agents while focusing on plausibility of behavior, real-time capability, and controllability.

An aspect typically ignored by cognitive architecture approaches is personality. However, endowing agents with a consistent personality is a requirement for believable behavior and reportedly affects outcomes like acceptance, credibility, and trust [MDWS00, Ort03, RGA+09, VSB+20, ZMLY19]. In contrast, modeling personality for agents is a standard procedure in IVA research; often combined with a representation of emotion and mood [AKG+00, KMT08, NFdSS10]. More recently, game developers have also started integrat-

[1] `https://unity.com/`, [online: May 2, 2023] Unity Technologies

ing personality models, recognizing the benefits for diversifying behavior of NPCs [BP17]. Personality in humans is a driving factor for their behavior, contributing to an individual's uniqueness [BTD14, CM09, Her09, Mat18]. It is reasonable, that the same observations hold for virtual humans. Therefore, integrating personality as a core aspect of a lightweight cognitive architecture is one of the major objectives of this thesis project. Especially the aspect of linking personality to task-specific behavior is considered psychologically (how to determine a link) and technologically (how to generate behavior from it).

The fact remains that many agent architectures already exist, some are even specifically aimed at virtual environments (e.g., CIGA [vOD11, vOVD12, vO14] or Pogamut [GKB+09, GBK+10]). Why is it necessary then to propose yet another architecture design to answer **RQ1 – What is an appropriate design for an attentive cognitive agent architecture in virtual environments?** The answer is, none of the researched approaches fulfill all three requirements of this thesis project: plausibility, controllability, and scalability. Other solutions either focus on a subset of the specified aspects or provide generalized middleware approaches. Therefore, based on **RQ1** and the current research gap regarding a personality-based, lightweight cognitive architecture, the following research task is formulated:

RT1: A lightweight cognitive architecture for virtual environments.

The main motivation of the work presented in this thesis project is to provide means for simulating autonomous software agents within a virtual world. These agents should blend into the environment in such a way that their presence improves the experience of a user interacting with the virtual world while simultaneously supporting a design goal (e.g., training). Inspiration towards achieving this task is taken from cognitive architecture research by identifying common components and assessing their utility towards the intended solution. The identified components should be arranged in a modular manner to maximize flexibility and adaptability to specific application scenarios. Due to the general application domain of real-time virtual environments, the scalability of solutions should be considered during all stages of the project. Generating consistent, individualized behavior is achieved by integrating personality as a core aspect of the architecture. Furthermore, since controllability is important, the architecture must be customizable through configuration. The result should be a theoretical agent architecture as part of the concept stage as well as a version of the architecture suitable for the application stage.

To outline the development of the cognitive agent architecture presented in this thesis, this chapter is structured as follows. In Section 3.1, the initial situation is described alongside a recap of cognitive architectures, which provide the main inspiration for the concept. This

basic discussion is followed by a theoretical concept in Section 3.2, presenting a methodological definition. A cognitive agent architecture design, which is based on the theoretical concept, is described in Section 3.3. In Section 3.4, the methodological design is reduced to a version that can be realized under the constraints of this thesis.

3.1 Prerequisites

The literal meaning of *agent* comes from the Latin word *agere*, which means "to act." An agent needs to be part of an environment to act, which can be a very abstract definition (e.g., [RN10]), a virtual environment (e.g., [CSPC00, RVP13, HL15]), or the physical world (e.g., [PLI07]). The role of the environment will be discussed when describing the environmental stimulus (see Section 5.2). Being situated in the environment is usually referred to as *embodiment*, which means giving a "physical" presence to an agent within its surroundings (cf. [Li15]). Physicality does not necessarily imply that the body of an agent is made of real-world material or that it exists in the real world. The body can also be rendered and animated using computer graphics. The important aspect is that the agent's body interacts with the environment including the associated limitations.

Traditionally, the body of an agent has been investigated separately from its mind, i.e., the agent's cognitive capabilities are isolated from its ability to perform actions in the environment [HL15]. Whether or not the strict separation of body and mind is useful in general can be debated from several points of view (e.g., see [RVP13]). In the context of this thesis, and from a technological standpoint, the classical separation is certainly beneficial. First, the two layers can be investigated almost independently, which allows focusing on the cognitive aspect while avoiding further complexity from techniques required to perform the actions, e.g., animation, physics simulation, inverse kinematics, etc. Second, if considered sufficiently, the separation can be leveraged to interchange bodies across or within simulations. The separation is not always well-defined and may blur occasionally. For example, sensors should obviously be a part of an agent's body, but in the design introduced in Section 3.4 they are part of the mind due to their close connection to the overall perception process. In cases where the agent's body should determine its sensory capabilities, a mechanism selecting sensors could be based on the current form of embodiment.

Based on the loose definition of an agent as a combination of body, mind, and environment, a closed perception-decision-action loop is defined (cf. [HR95, RN10]). The body provides information from the environment to the mind. The mind then acts upon this information by selecting an available action and tasking the body to perform that action. Once the body has completed the action, the environment has changed, which is in turn

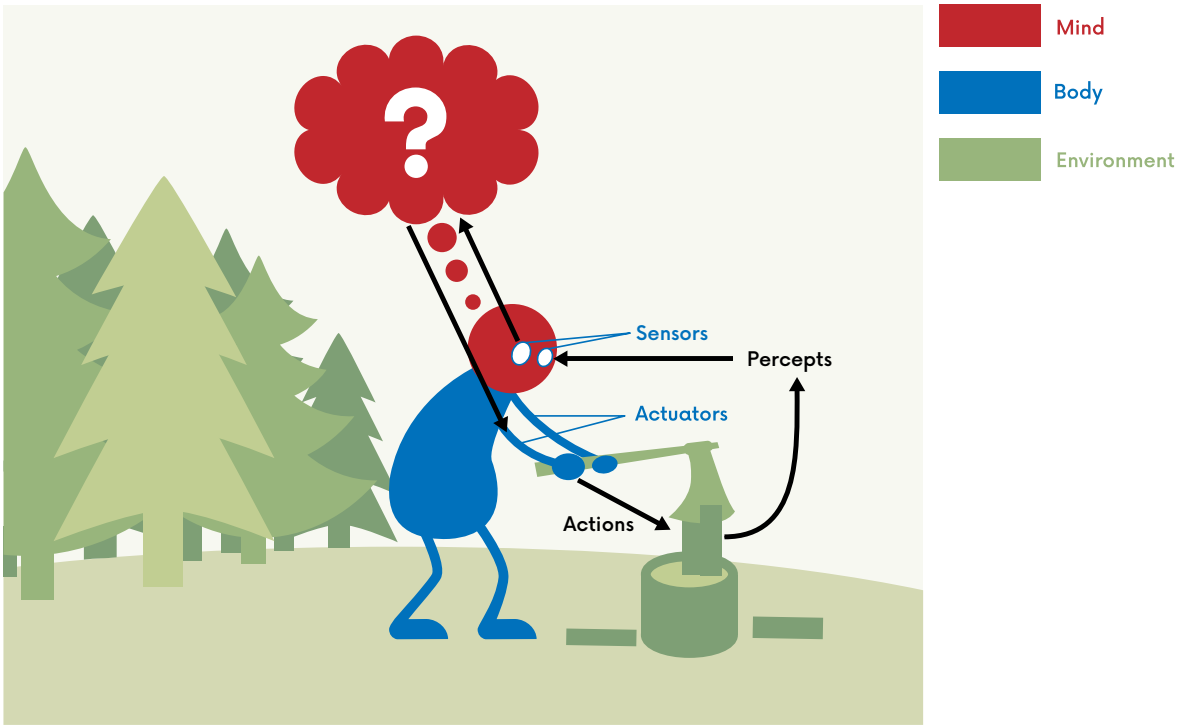


Figure 3.1: The definition of an agent requires a combination of body, mind, and environment. Based on these components, a simple reflexive perception-decision-action loop can be defined. The agent perceives the state of the environment using sensors. Based on this information a decision is made, which changes the environment state by performing an action using the body's actuators. The image is based on [RN10].

perceived by the agent. Figure 3.1 illustrates the separation of body, mind, and environment including a depiction of this simple loop, which defines a reflexive agent [RN10].

Reflexive agents may be sufficient for simple task domains, requiring little to no further components other than the ones indicated in the perception-decision-action loop. Russel and Norvig discuss a simplistic example of a vacuum cleaner agent whose perception is just a tuple of the location it is in and the information whether that location is clean or dirty. The agent's only actions are moving to one of two locations or cleaning and its decision process is one simple if-else rule. However, as the application domain and the desired agent behavior become more complex, the architectural design generating the behavior must become more sophisticated [HR95]. Especially, when attempting to imitate humans in a virtual world, additional structures become necessary, e.g., to keep track of past percepts, learn from experiences, or act towards achieving a goal. Due to their objectives of achieving artificial general intelligence, cognitive architectures (such as Soar [Lai08, Lai12]) seem a logical choice for realizing human-like, cognitive agents. However, designing models of the human mind, capable of solving general problems, naturally results in complex architectures.

These architectures are typically difficult to understand, maintain, and apply. Additionally, they require extensive computational resources making them cumbersome for real-time multi-agent scenarios. Yet, cognitive architecture research is rich in history, examples, and insights that provide useful inspirations for a *lightweight cognitive agent architecture* design (see Chapter 2). Besides cognitive architectures aimed at contributing towards artificial general intelligence, there are examples that focus on isolated aspects of human cognitive processes. The architecture design proposed in this chapter features the elements that are most common in cognitive architecture research, most notably short-term and long-term memory, decision making, learning, and perception [DOP08, KT18, MS11].

3.2 Theoretical Concept

As mentioned at the beginning of this chapter, the requirements for an envisioned solution of authoring and simulating agents in virtual environments are believability, controllability, and scalability. Based on these requirements and the associated scientific gap, a review of existing cognitive architectures and virtual human architectures was conducted to identify essential cognitive skills. From the wide variety of identified skills, a set of necessary cognitive skills and functionality is compiled. This set is described in the following and is subsequently used to put forth the concepts introduced in Sections 3.3 and 3.4.

3.2.1 Perception

The first interesting aspect is to model what an agent should be able to perceive from its environment. To fully simulate an environment, all information about said environment are known and can thus be provided to every agent (cf. [GLBV08]). This, however, is not a realistic reproduction of human behavior. For example, a human cannot know for certain whether other beings are around if they are not perceived by at least one sensory organ. Therefore, a "perception filter" needs to be applied to control which information reaches the agent in a reasonable manner. A typical "filter" designed after their biological examples are sensors. The number and types of sensors available in an architecture determine the amount, type, and complexity of input into the system. Often sensors correspond to human senses (i.e., vision, hearing, touch, smell, taste, proprioception), but they can also include modalities not available to humans (i.e., symbolic via keyboard or GUI, ultrasonic, laser, infrared, GPS, and others) [KT18]. Depending on the application, some modalities make more sense than others, some sensors may function as shortcut to avoid intensive calculations or intricate algorithms (e.g., using speech-to-text instead of modeling the perception of sound), and some may be unnecessary [KT18]. For example, for agents in a traffic simulation, the visual

and auditory senses are the most important. In contrast, simulating smell or touch is a waste of recources during development and during simulation.

In this thesis, attention is considered as part of the perception process, which is sometimes called *perceptual attention* or *external attention*. One biological reason for attention is to decrease the amount of information provided by the senses. In cognitive architectures this relates to selecting and modulating information coming from different sensors with visual sensors being the most common. Tasks covered by this type of attention are typically region of interest selection, gaze control, task-driven attention, and data-driven attention [KT18]. Generally, attention in agent systems serves both aesthetic and functional purposes. One aspect of the functional purpose is reducing information load as mentioned above. Another aspect is to orient the agent's sensors, ensuring that the relevant information is provided to its depending processes (e.g., action selection). The aesthetic purpose is to communicate the agent's internal attention processes overtly to an observer, increasing the *naturalness* of the entity by emulating what we, as humans, are used to observe in other beings [PCR+11].

3.2.2 Memory

According to Atkinson and Shiffrin, memory is the component that distinguishes human beings from each other; that determines who we are [AS16]. What the human brain has stored about the world dictates what we perceive and how we perceive it [GB16], who we talk to, what we eat, what we read, and so forth. The knowledge stored in memory basically determines almost all human interactions and decisions [AS16]. Therefore, it seems obvious that all cognitive architectures contain some sort of memory model serving similar purposes, i.e., storing intermediate results [DOP08, KT18]. Ideally, a realistic memory module would be an effigy of the human original including its abilities and disabilities. One of the major problems with this ideal is the fact that, due to its complexity, the human brain has yet to be entirely understood (cf. [GB16, PO02]). Consequently, memory models can only be simplifications that emulate observed functionality. The main task of memory is storing results, but actual implementations differ widely depending on the task domain. Most memory modules are structurally organized into a hierarchy of sub-modules. While research suggests that a multitude of "memories" exist in the human brain (e.g., iconic, echoic, episodic, face, and voice memories [Bad03]), many examples distinguish between a short-term (or working) memory and long-term memory (e.g., [Lai12, LC06, KT18, PO02]). The exact purposes and naming of the two may differ, but usually their general purpose is implied by the names given above. The long-term memory is used to store information that is meant to reside within the system for extended periods of time. The short-term memory represents information currently needed by the system to operate, e.g., the content of a

goal stack in planners [KT18]. How, when, and where information is stored is defined in accompanying processes. These memory processes include, e.g., rehearsal, memory search and retrieval, or discard in accordance with memory capacity. Together with the structural component, they define the memory model (cf. [AS16, KT18]).

3.2.3 Internal Simulation

Another important skill learned by humans is to anticipate what consequences will result from their actions. Such anticipation requires an internal representation to estimate a possible outcome of the environment's state after an action has been performed. This could, for example, involve spatial reasoning in the form of visual imagery as proposed by Lathrop and Laird [LL07]. In any case, it will require a form of internal simulation. A closely related issue is that of predicting behavior of other agents to plan or act accordingly. To model this ability, an agent must be given the tools to perceive other agents and to anticipate their future actions. In other words, an agent must be able to project its ability to foresee the results of its own actions as well as its reasoning processes onto other agents. During this process, agents should not be able to acquire more information about other agents than humans can derive from observing other humans. For example, agents should not be able to access another agent's memory, although this could be achieved for virtual agents without difficulty.

3.2.4 Alternative Behavior

A major shortcoming of state-of-the-art simulations in virtual environments is the inability to produce alternative behavior to otherwise irresolvable situations. To clarify this issue, consider the following classic example: Either by design or through an action of the player, a dead end is created in a level of a digital game, resulting in NPCs getting stuck due to their programmed behavior. According to [YT18], this result is a problem of level design or AI testing in game development. A simple heuristic to resolve the issue is often to remove the agent from the environment if it cannot advance. However, this solution will certainly break the suspension of disbelief and reduce the player's immersion; the very thing that it is supposed to avoid. Instead, it would be much more interesting if the agents would start to "think" of an alternative plan to resolve the situation that lies outside the scope of the programmed rules. For example, agents could "re-purpose" existing structures to pass around the blockage, such as a car temporarily using the sidewalk or a character climbing through a window. Although, these usages are not part of the agent's regular behavior, they seem appropriate to an observer if the circumstance justifies it. Of course, one or more alternative solution could be added to the regular action-selection process after being discovered and would then have to be covered by a testing or level design process. However,

it is impossible to think of, design, and test every possible scenario. Ideally, agents would be able to derive alternative affordances of structures or objects themselves and utilize them if necessary. However, if agents are allowed to break existing rules for navigation, social protocol, or object usage, they must do so only if it makes sense to an observer.

3.2.5 Individual and Emotional Decision-Making

The need for having individualized behavior is typically not a topic of AI research. In IVA research, however, it is a common theme (e.g., see [KMT08, HR15]). The agent architecture developed in this thesis is meant to be a basis for multi-agent simulations in virtual environments. Consequently, users will be able to observe multiple agents in similar situations. To keep the user's sense of presence, agents should not all behave in the same way. Instead, similar to reality, behavior should be individual to increase believability [KMT08]. In addition to memory (see Section 3.2.2), personality is one further aspect that defines humans as individuals. A person's personality is a relatively stable characteristic that influences that person's responses and decisions, and thus the observable behavior [BTD14, CM09, Mat18]. Therefore, IVA researchers have been investigating the use of personality, typically for driving decision-making processes. However, personality does not only influence decision-making but the entire cognitive system [SFC+10]. This observation should be reflected in the architecture concept accordingly.

Yet another important aspect of everyday life are emotions, moods, and feelings. Although these are non-cognitive factors, the current emotional state can influence memory and perception and thus the decisions someone makes [Bad03, PCR+11]. Therefore, they are considered here as one of multiple cognitive processes. Simulation of these factors ties in closely with the previous goal of dynamic behavior as a specific emotion might increase the probability to break given rules and to behave irrationally; maybe even dangerously. Decisions are influenced by both the current situation and the current emotional state. Therefore, since emotions, mood, and feelings influence perception and decision-making, the entire cognitive process is affected (e.g., see [SFC+10, SFC+11]).

3.2.6 Miscellaneous Factors

In addition to the items listed above, the behavior of agents may also depend on additional factors. For example, an agent's current role can have an influence on its action selection. Depending on whether the agent impersonates a soldier, a tourist, a sidekick of the player, or a trainer in a training exercise, it will be more or less likely to make certain decisions. Other factors could be the time of day, weather conditions, the agent's current objective, its age or gender, and more; too many to be listed or considered here.

3.3 Architecture Design

From the desired characteristics detailed in the previous section, the methodological cognitive architecture concept shown in Figure 3.2 is derived. One important aspect of generating plausible behavior is making sure that agents do not all act the same. To achieve individualized behavior, a **psychological personality profile** is part of each agent. As a result, the architecture design consists of two layers: a foundational layer including the personality profile and on top of this layer, cognitive process modules determine an agent's capabilities. The structure of the cognitive layer is inspired by cognitive architectures as mentioned above. All the cognitive processes in the second layer are influenced by the underlying personality profile. Through this influence, the personality layer diversifies generated behavior between individual embodiments of agents. At the same time, the personality ensures that an individual agent's action selection process is consistent with its past behavior patterns, which avoids the impression of non-deterministic decision making. The general design allows for interchangeable implementations of personality profiles, and it incorporates a model able to transition results of psychological studies to agents. When generating profiles from real studies, it is possible to link personality prototypes to domain-specific behavior patterns that represent real-life behavior (e.g., driving [Her09]). In some instances, it may be sufficient to achieve archetype behavior not specifically linked to real-life behavior. In these cases, a similar approach can be used by assigning personality prototypes to appropriate agent types. More details about this topic can be found in Chapter 4.

Information from an agent's environment (symbolized by the top arrow in Figure 3.2) enters its mind by passing through a **perception** module (see Section 3.2.1). The module filters information coming from the environment; its complexity depends on the level of realism that is to be achieved as well as on the available sensors and sensor modalities. Additionally, a module representing **emotion, mood, and feelings** is coupled to the perception module, because what an agent perceives[2] should influence its emotional state and its emotional state should influence what it currently perceives.

At the center of the cognitive processes is the **short-term memory**, representing the agent's current knowledge about itself and the world it inhabits. The short-term memory is the agent's working memory; the basis for every subsequent process, which must be filled with information from either perceiving the world around the agent or from **long-term memory**. Depicting the long-term memory as one module may be too optimistic as it can include multiple types of information, e.g., episodic memories, social protocols, semantic or

[2]Note that there is a difference between perceiving and sensing information involving where information is stored in memory. However, to keep the description simple at this point, the two are not differentiated in this chapter. Details about the differences can be found in Chapter 5.

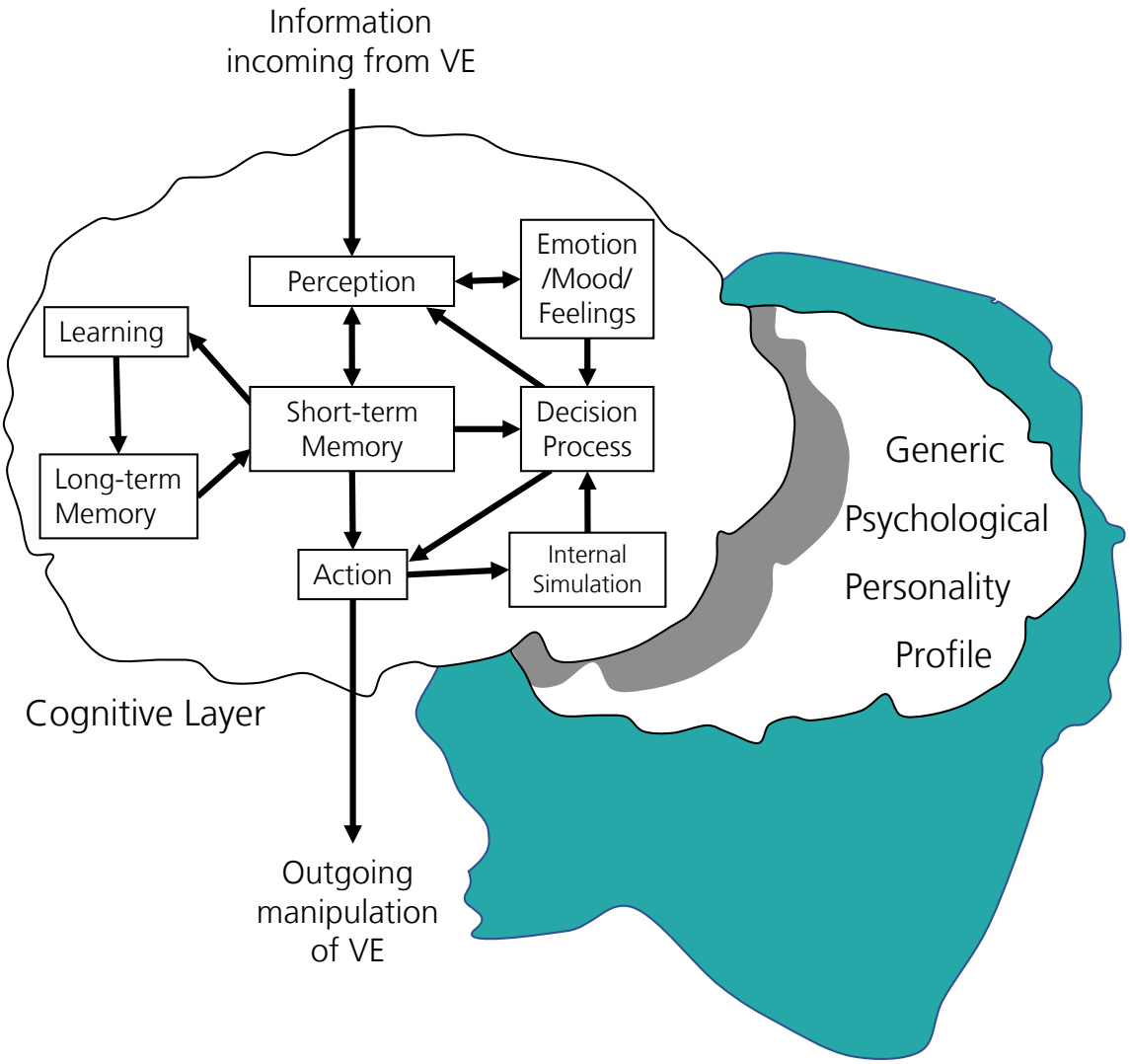


Figure 3.2: A methodological concept of an architecture for cognitive agents in virtual environments. A layer of cognitive process modules is situated on top of a personality profile layer. Interaction with the agent's body and environment are indicated by the incoming and outgoing arrows (above and below the cognitive layer). The cognitive processes define the agent's capabilities, which are all influenced by the underlying personality profile. Image based on [SvS20].

global knowledge, strategies, etc. [KT18, PO02]. Each of these types of long-term information could be stored in different ways and in different modules. At this level of abstraction, it is sufficient to group this information in one module.

If information residing in short-term memory is useful for future situations, e.g., observations of action results, a **learning** module encodes this knowledge into long-term memory. This connection may be more complicated when implemented as it may have to retrieve information from other modules. However, every information available in other modules can be made accessible by encoding it in short-term memory.

A **decision process** module retrieves knowledge from working memory to select the appropriate action from an action module. The agent's current emotional state may also influence this selection process, depending on the type and strength of the emotions, the mood, or feelings. The emotional state can be one factor that triggers the decision process to adapt action selection to the current situation over time. In contrast to the description in Section 3.2.5, memory is influenced by the emotional state only indirectly through emotionally altered perception and decision making, as an additional direct influence on memory is redundant. Alternative behavior generation, as discussed in Section 3.2.4, is considered part of the decision module here. The functionality of this module can be provided, e.g., by straightforward if-then rules, elaborate machine-learning algorithms, or other intricate decision processes. In most cases, application-specific needs will dictate the actual realization of the decision module, especially regarding the trade-off between autonomous behavior and control over generated behavior, which was discussed in Section 1.2. In this thesis project, the decision process is not further elaborated and a behavior tree implementation is used in the realization of the architecture concept.

The **action** module may store all actions available to an agent and provide them for retrieval. The module may also be responsible for executing the actions (including processes like navigation or animation playback). Actions that influence the environment are performed by an agent's body. However, actions should not only be performed in the agent's environment (indicated by the bottom arrow in Figure 3.2). It should also be possible to perform the action within an **internal simulation** whose outcome can influence which action is selected for performing externally. This mechanism can be used to realize the anticipation of results from the agent's own actions or from those of other agents (see Section 3.2.3). The distinction between internal and external action is the reason for having the action module reside within the agent's mind. The arrow pointing to the agent's environment represents *environmental actions*, i.e., actions that use the agent's body to change the state of the environment. The other arrow represents *internal actions*, i.e., actions whose consequences are simulated within the agent's mind.

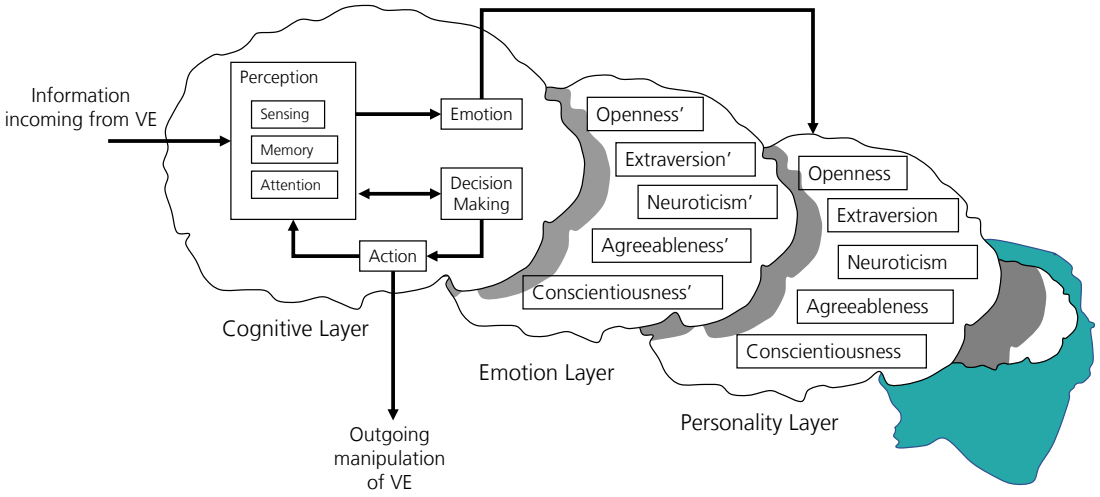


Figure 3.3: Depiction of the CA²RVE architecture design. An underlying personality profile is influenced by the current emotional state, resulting in a dynamic intermediate personality layer. On top, a cognitive layer includes the agent's cognitive processes, which are influenced by its personality and indirectly by its emotional state. Perception "filters" information coming in from the environment. The module includes sensation, memory, and attention because the processes are closely related. A decision-making module represents several high-level processes used for action selection. An action module serves as output interface to the environment.

3.4 CA²RVE - Cognitive Attentive Agents for Real-time Virtual Environments

The cognitive agent architecture concept proposed in the previous sections is directed towards meeting the characteristics discussed in Section 3.2. The concept's main purpose is to provide a theoretical agent architecture for solving the research questions considered in this thesis and is the result of the conceptual research stage. During the application stage, the theoretical design is refined, condensed, and re-ordered to focus on lightweight and robust real-time application without compromising the overall design. Since the intended use for this architecture are virtual trainings, simulations, and (serious) games, computational performance must always be considered. Additionally, depending on the application domain, the definition of plausible behavior may vary. The architecture should reflect that ambiguity by allowing the necessary flexibility by enabling configuration towards specific application scenarios. For that purpose, several control mechanisms are integrated, and components are defined in more detail (see Chapters 4 and 5). Additionally, elements are grouped in specific ways, e.g., to enable interchangeability or efficiently swapping components in or

out. Figure 3.3 demonstrates the changed concept of the **C**ognitive **A**ttentive **A**gents for **R**eal-time **V**irtual **E**nvironments (CA^{2}RVE) architecture.

As in the methodological concept, the foundation layer consists of a personality profile. The generic model is replaced by the Five-Factor Model (FFM). The model is a popular choice in virtual agent research, due to its intuitive and descriptive characteristics [AKG$^+$00, KMT08]. Additional reasons for the model's validity were mentioned in Chapter 2. Additionally, a second personality layer is introduced to capture the aspect of dynamic behavior generation influenced by the current emotional state. The static personality profile of an agent is altered by its current emotional state, resulting in a dynamic personality profile. This approach is not psychologically correct, but has two advantages: (1) The influence of the emotional state can be activated or deactivated without having to alter any other process. This flexibility is useful for customizing the architecture to specific scenarios but can also be utilized for evaluations comparing emotional to non-emotional agents. (2) The influences of the emotional state and the personality profile do not have to be encoded in every cognitive module. Instead, emotion indirectly influences all modules via the altered personality.

Due to the intended use of the CA^{2}RVE architecture, the aspects mood and feelings were removed from the application stage design. The main reason is that the effect, which is observable by users, will only last for short periods of time. Mood and feelings are characteristics that persist or develop over longer periods of time, which users would not be able to perceive. If modeling one or both effects should become necessary, it would be possible to encode them either in the personality model or in memory.

Information reaching the agent from the environment still passes through a perception module, but because perception is closely related to memory storage and processes, the memory module is integrated into the perception module. One example of this connection is that perceiving information – in contrast to simply sensing it – means that the agent needs to become aware of the information (cf. [Enn04]), which occurs once the information resides in short-term memory. The objective for refining the memory module was to keep the structural hierarchy defined in the theoretical concept, but focus on the processes for storage, decay, and information retrieval. Furthermore, attention is also defined as part of the perception module for similar reasons. Perception still directly influences emotion, but due to the changes regarding personality, the emotional state influences perception indirectly through the dynamic personality layer.

During this thesis project, the focus was on designing the foundation consisting of personality and emotion models as well as the perception module. Consequently, two aspects included in the theoretical concept had to be set aside. Learning is omitted from the architecture, because making a significant contribution to this topic would have required

focusing on this aspect alone. Internal simulation is not an explicit part of the cognitive layer and is not discussed further in this thesis, although it could be defined as a part of the decision module if necessary. Selecting certain actions, like turning the head to direct attention towards specific objects or events in the environment, will automatically result in different input to the perception module. This influence is only implied in the methodological concept. Due to the focus on perception, a direct connection between the action and perception modules is included in the CA^{2}RVE architecture.

3.5 Conclusion

In this chapter, the results of **RT1** were presented. First, a selection of required characteristics of a cognitive agent architecture for virtual environments was described. The characteristics were derived from cognitive architecture research, IVA research, and general considerations regarding the intended usage. Second, based on these characteristics, a theoretical agent architecture was designed to fulfill the discussed requirements of believability, controllability, and scalability. The foundation is a personality model to satisfy the intent of providing individualized behavior in a multi-agent setting. On top of this foundational layer, a modular collection of cognitive processes is used to define an agent's set of capabilities. Besides the basic components for perception, decision-making, and actions, further cognitive components were integrated into the design. A hierarchical memory system provides agents with a model of the environment in its current state and the opportunity to draw from global knowledge and experiences. Internal simulation is used to model anticipation of an agent's own actions as well as actions of other agents. A decision module in combination with learning capabilities can enable the ability to find alternative behaviors to resolve situations that the default behavior cannot handle. A module modeling emotion, mood, and feelings is a short-term to medium-term extension of the temporally stable personality profile that facilitates individualized behavior that can also reflect an agent's current situation.

Third, the theoretical architecture was adjusted to focus on specific sub-areas of the design and to consider customization toward the targeted domain, real-time constraints, and controllability aspects. In the resulting CA^{2}RVE architecture design, sensing and memory were integrated into the perception module due to their close interrelation and attention was added as a separate module. High-level processes like learning, internal simulation, and planning were subsumed within the decision-making module. This decision was made to be able to focus on personality, emotion, and perception, while maintaining the option of defining a more detailed design of the module in future work. Due to the limited ability of observing changes based on mood and feeling, these factors were discarded from the design.

Instead, emotion remained as the dynamic factor that can alter an agent's typical behavior, as its effect is short-lived, i.e., observable during a single simulation. An underlying personality profile still defines an agent's typical behavior, but a dynamic version of the profile is inserted between the bottom layer and the cognitive layer. This intermediate layer is a combination of the original profile and the current emotional state. This design is not psychologically sound but was chosen to allow the effect to be switched on or off whenever required and to transfer the current emotional state to any of the cognitive modules.

With the results of the first research task (**RT1**), the intention was to provide a possible solution to **RQ1**. Judging the quality of the results in isolation, i.e., solely based on the provided descriptions, is difficult as it would be a subjective discussion. Therefore, the architecture must be realized and integrated into a real-time application to evaluate it regarding the identified requirements of believability, controllability, and scalability. An exemplary realization of the architecture is presented in Chapter 6 and several evaluation scenarios are described in Chapter 7. However, before the realization and evaluation, an in-depth look at three essential architecture components is provided in the following two chapters. The static personality profile is described in Chapter 4 and expanded by explaining the role of emotion in the dynamic personality layer. Chapter 5 provides a detailed look at the perception module and the included components and processes.

4

Personalized and Emotional Agents

"A person cannot change his personality. The only thing he can change is 'habit'. He can play around with behaviors though."

- Andrea L'Artiste

The motivation for this thesis project is to define an architecture capable of generating more believable agent behavior for real-time virtual environments. To achieve this objective, a lightweight cognitive agent architecture is devised and described in Chapter 3. The methodological concept of the architecture includes modules for emulating perception, memory, emotion, decision-making, action, internal simulation, and learning. To consider customization toward the targeted domain, real-time constraints, and controllability aspects, the number of components is further reduced to perception, emotion, decision-making, and action as described in Section 3.4. Expanding the notion of believability of IVA behavior to multi-agent scenarios additionally requires individualized behaviors. Otherwise, all agents act the same, likely hindering immersion or even breaking a user's suspension of disbelief. One approach of achieving individualization is to endow each agent with its own personality. Personality is one characteristic that makes a human unique (cf. [BTD14, CM09, Mat18]). It determines a human's performance in specific tasks (e.g., [MDWS00]), it influences how susceptible someone is to emotions (e.g., [WC92b]), and it is a driving factor for behavior (e.g., [Her09]). Similarly, according to Matthews et al. [MDWS00] and Ortony [Ort03], a consistent personality is a requirement for believable agent behavior with the additional benefit of increasing acceptance, credibility, and trust in virtual characters (e.g., [RGA+09, VSB+20, ZMLY19]).

Cognitive architecture research, which is the main reference for the agent architecture presented in the previous chapter, generally does not consider personality for agents. Therefore, this field cannot be used as reference in this matter. Furthermore, there are opinions

stating that human-derived personality models may not be applicable to virtual humans at all (e.g., [VSB+20]). In contrast, there is evidence that utilizing human personality models in agents is possible (e.g., [CVB+12, CHLC18]) and it is a standard tool in IVA and ECA research [AKG+00, KMT08, NFdSS10, VSB+20]. For example, Rushforth et al. [RGA+09] extended virtual humans with a personality model based on the NEO-PI-R, a variant of the Five Factor Model (FFM) [MC08]. The personality parameterization determined the choice of dialog options for a virtual character in a military tactical questioning scenario. They found that different personalities produced discernible behavior that could be observed by test subjects. Bevacqua et al. [BdSP+10] used Eysenck's three-factor model to control so called backchannel preferences of agents when reacting to user input. They anticipated that this extension would improve user perception of virtual characters. Castillo et al. [CHLC18] used similar methods and found that standard personality questionnaires can be applied to measure the personality of virtual humans. However, they also reported that the precision of the perceived personality depends on the available communication channels (e.g., voice only, visual only, multimodal). Zhou et al. [ZMLY19] did not only equip a chat-based interviewer agent with a FFM personality, but also had the agent deduce personality profiles of users to investigate how similar personalities affect user perception of and their trust in the agent. Völkel et al. [VSB+20] even developed their own personality model dedicated to speech-based conversational agents, such as digital assistants. Knob et al. [KBM18] used the FFM to diversify group behavior in crowd simulations. Gebhard described ALMA, a layered affect model that integrates personality, mood, and emotion into virtual humans to increase their believability. The employed personality model was the FFM, emotions were represented using the OCC model, and the pleasure, arousal, dominance (PAD) model was used to formalize mood. In ALMA, mood is affected by emotion and personality in PAD space and influences agent behavior.

Despite its common usage, the derivation of behavior from personality representation is often anecdotal, based on personal experience, or determined by trial and error (e.g., [RGA+09, BdSP+10]). Additionally, several human personality models exist and are employed within IVA/ECA research. The most common are the Five Factor Model (FFM) or Big Five [MJ92, MC08], Eysenck's personality model [Eys47, Eys70], and Cattell's 16 Personality Factors [Cat46, Cat57]. But what makes one model more "correct" or more suitable for virtual agents than another? Even if one model is chosen, different reporting techniques and measuring scales can exist. For example, the FFM can be measured using the NEO FFI [CM92] or the Big Five Questionnaire [CBBP93], among others. The situation is complicated by the fact that there can be model variations, such as the NEO-PI-R inventory that further divides each personality trait of the FFM into six facets. One of the research

questions asked in this thesis is: **Can the proposed agent architecture systematically generate individual, dynamic agent behavior (RQ2)?** More specifically, the question is whether personality profiles can be integrated into a lightweight cognitive architecture to achieve this objective. The investigation of this question is part of the research tasks **RT2.1** and **RT2.2**:

RT2.1: Generating individual, consistent behavior patterns using a lightweight cognitive agent architecture.

To integrate individualized behavior, an agent's personality should be the core layer for all further functionalities. Consequently, agents select possible actions not only based on their current surroundings and situation, but also according to their personality. This selection process will create variety in an otherwise deterministic process. Since actions are chosen based on personality, individual behavior patterns vary from agent to agent, but do not seem random as they are consistent with each agent's previous actions. As a result, the architecture is able to generate individual, consistent behavior patterns.

Furthermore, instead of having to select a specific personality model, a generic model for computational use of personality profiles in autonomous agents is developed. Such a generic model would be independent of employed reporting techniques or measurement scales allowing different personality models to be related to each other. The additional purpose of using a general personality model within the architecture is mapping psychological personality-based studies to a realization of the formulated concept. These mappings can be used to reproduce realistic domain behavior reported by such studies. To demonstrate the utility of the approach, a typical realization of the general model is provided using the commonly used FFM. Based on the personality definitions, task parameters are used to utilize the model in other cognitive processes, like decision-making.

RT2.2: Adapting personality-based behavior to an agent's current situation.

An agent's underlying personality model defines its individual behavior. This mechanism by itself will result in identical action selection when an agent is presented with identical situations (as presented by its own perception capabilities). Consequently, if a selected action does not change an agent's situation, the agent will repeatedly select the same action, which may also degrade the plausibility of observed behavior.

To overcome this issue, agents can be provided with enough alternative actions and a mechanism to choose between them. An obvious realization of such a mechanism is probabilistic selection. However, without any further adjustments, this approach would disconnect the action selection from the situation resulting in a less comprehensible process for an observer. While the effects can be mitigated, e.g., by introducing a utility function, the

most realistic solution would be providing agents with the ability to become aware of and understand their situation and find solutions that are not pre-defined by a system designer. Unfortunately, these approaches are too complex to be considered within the constraints of this thesis project regarding computational requirements and available resources. Instead, the desired alteration mechanism is provided by combining the personality model with an emotional state that dynamically changes based on an agent's circumstances. Other affective agent architectures often also model mood, a lasting state of feeling created by complex cognitive processes [Tha89]. However, since users are not able to observe changes of such lasting states during a simulation in the intended domain, a mood model has not been included in the agent architecture. The proposed architecture includes only emotions – short-lived feelings caused by experienced events [Tha89]. These predefined events influence the behavior for a limited time and fade until normal behavior, as defined by the personality, is restored. This approach avoids the aforementioned disconnect and is consistent with the chosen personality-based approach.

The remainder of this chapter contains the results of **RT2.1** and **RT2.2** realized as two layers: (1) A static personality layer that generates consistent behavior patterns (see Section 4.1) and (2) a dynamic personality layer influenced by an emotional state enabling an agent to adapt to events in its environment (see Section 4.2).

4.1 Psychological Personality Profiles

Within the lightweight cognitive architecture concept, personality is used to individualize agent behavior. Before personality can be used to generate behavior, it needs to be represented in a useful way. Additionally, the intent is to incorporate psychology studies that link task behavior to personality to emulate realistic behavior patterns[1] (e.g., the correlation of personality and driving behavior [Her09]).

For both purposes, trait theory is useful as it views human personality as a combination of mostly independent trait factors. Based on such a combination, a computational model can be defined that is easy to understand, configure, and adapt. Each factor can be represented as a real number, allowing the definition of trait vectors that form a personality. Trait theory is also the most common approach to psychology research [Mat18], which means a large collection of studies that relate a trait model to other cognitive aspects (e.g., emotions, task performance, perception) is available. The results of these studies can then be used to

[1]Note that realistic does not automatically mean plausible. The perception of plausibility depends on the circumstances and the rules of the virtual world that is experienced by an observer.

configure the generation of agent behavior. In the following, a general trait-based personality model is defined that does not restrict the number or type of traits as different applications may require different profiles or numbers of trait dimensions. Furthermore, it is shown how the model is substantiated for its integration into the proposed agent architecture including its connection to the processes in the cognitive layer.

4.1.1 Representing Personality

Although, trait theory is very common among psychologists, there are various models utilizing different numbers and different types of traits. Even within the same model, scales and values differ depending on the means deployed to map personality to traits of the model. For example, NEO-PI-R, NEO-FFI [CM92], or TIPI [GRS03] are different questionnaires that measure FFM personalities. In the following, a general trait-based personality model is defined to avoid committing to a specific set of traits and to provide a generic solution:

$$\begin{aligned} &\mathbf{p} = \langle p_1, \ldots, p_n \rangle \in \mathbb{R}^n \\ &D = \{\mathrm{d}_1, \ldots, \mathrm{d}_n\}, \forall\, \mathrm{d}_i \in D : \mathrm{d}_i(\mathbf{p}) = p_i, 1 \leq i \leq n, \mathrm{d}_i : \mathbb{R}^n \rightarrow \mathbb{R} \end{aligned} \tag{4.1}$$

A personality profile $\mathbf{p}$ is defined by an n-tuple of traits, also referred to as dimensions. The number of dimensions may be derived from an existing personality model or depend on the intended application. Each function of the set D maps a profile to a single value representing one particular dimension. In the context of this thesis project, the projections are restricted to $\mathrm{d}_i = p_i$.

One of the motivations is to be able to use psychological studies about human behavior and apply them to the CA^{2}RVE architecture. This requires a mechanism to utilize the dimensions of a personality profile for decision-making and a mapping from a study to the chosen personality model. The former will be discussed in Section 4.1.2. To achieve the latter, a study is defined based on the generic profile from Definition 4.1 as follows:

$$\begin{aligned} &\mathscr{S} = \langle P, C, l, h \rangle \text{ with} \\ &P = \{\mathbf{p}_0, \ldots \mathbf{p}_q\}, \\ &C = \{\mathscr{C}_0, \ldots, \mathscr{C}_m\}, m \leq q \\ &l, h \in \mathbb{R}, \forall\, \mathrm{d} \in D \wedge \forall \mathbf{p} \in P : l \leq \mathrm{d}(\mathbf{p}) \leq h \end{aligned} \tag{4.2}$$

According to this definition, a study $\mathscr{S}$ consists of a set of personality profiles P, a set of profile classifications C, and the theoretical minimum l and maximum h of the study. The set P contains all profiles that were found in the study, i.e., each profile represents one test subject. These profiles are divided into classes $\mathscr{C}_0$ through $\mathscr{C}_m$, which should not

only represent the measured profiles using a much smaller set C but are typically used to differentiate behaviors. Depending on the study and the measurement techniques used therein, ranges and distributions of values may vary between studies. Thus, it is useful to define classes in a standardized manner by using a z-score profile:

$$\mathbf{z} = \{z_1, \ldots, z_n\} \text{ with } z_i = \frac{p_i - \mu_i}{\sigma_i}, \tag{4.3}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are average and standard deviation profiles with respect to a set of profiles P:

$$\begin{aligned} \boldsymbol{\mu} &= \{\mu_1, \ldots, \mu_n\} \text{ with } \mu_i = \frac{\sum_{\mathbf{p} \in P} \mathrm{d}_i(\mathbf{p})}{|P|} \\ \boldsymbol{\sigma} &= \{\sigma_1, \ldots, \sigma_n\} \text{ with } \sigma_i = \sqrt{\frac{\sum_{\mathbf{p} \in P} (\mathrm{d}_i(\mathbf{p}) - \mu_i)^2}{|P|}} \end{aligned} \tag{4.4}$$

Using the equations from 4.3 and 4.4, a class of a study $\mathscr{C}$ is defined by:

$$\mathscr{C} = \langle Q, \dot{\mathbf{p}}, \mathbf{z} \rangle \text{ with } Q \subseteq P, \forall\, \mathrm{d}_i \in D : l \leq \mathrm{d}_i(\dot{\mathbf{p}}) \leq h, \tag{4.5}$$

where $\dot{\mathbf{p}}$ is a prototype personality profile representing class $\mathscr{C}$ and $\mathbf{z}$ is the z-score profile of $\dot{\mathbf{p}}$ regarding the set of personality profiles P of the study. The prototype profile $\dot{\mathbf{p}}$ does not have to be contained in P and therefore not in any set Q, because it is typically a statistical representation. Q is a subset of profiles from P, which are associated with class $\mathscr{C}$. Note that not every profile determined in study $\mathscr{S}$ must be associated with a class identified in or applied to the study. However, every profile from the study can only be associated with one class, i.e., for the set of classes $C = \{\mathscr{C}_1, \ldots, \mathscr{C}_m\}$ the following holds:

$$\forall \mathscr{C}_i, \mathscr{C}_j \in C : Q_i \cap Q_j = \emptyset, i \neq j \tag{4.6}$$

These definitions provide a theoretical tool to realize personality profiles for agents that can be derived from different psychological studies. However, to be able to utilize studies in a meaningful way, it is necessary to choose one model for the realization that is also used in all referenced studies. One model commonly used in psychology and also IVA research is the FFM [HR15]. The model has also proven to be consistent across observations, self-reports, and interviews in different languages [SGW12]. Additionally, the FFM has been used in studies to link human personality to behavior (e.g., [MDWS00], [Her09], [LL01]). Together with the provided definitions, these links can be used as indication of how to use trait values of an agent's profile to drive action selection and therefore behavior. Due to these reasons, the FFM was chosen as the specific personality model in the CA^{2}RVE architecture.

According to the definitions above, a FFM profile is defined as follows:

$$\begin{aligned} \mathbf{p}_{FFM} &= \langle p_1, p_2, p_3, p_4, p_5 \rangle \in \mathbb{R}^5 \\ D_{FFM} &= \{\mathrm{o}, \mathrm{c}, \mathrm{e}, \mathrm{a}, \mathrm{n}\} \text{ with} \\ \mathrm{o} &: \mathbb{R}^5 \to \mathbb{R}, (\mathbf{p}) \mapsto p_1 \\ \mathrm{c} &: \mathbb{R}^5 \to \mathbb{R}, (\mathbf{p}) \mapsto p_2 \\ \mathrm{e} &: \mathbb{R}^5 \to \mathbb{R}, (\mathbf{p}) \mapsto p_3 \\ \mathrm{a} &: \mathbb{R}^5 \to \mathbb{R}, (\mathbf{p}) \mapsto p_4 \\ \mathrm{n} &: \mathbb{R}^5 \to \mathbb{R}, (\mathbf{p}) \mapsto p_5 \end{aligned} \tag{4.7}$$

where the dimensions correspond with the model's five traits: **o**penness to experience, **c**onscientiousness, **e**xtraversion, **a**greeableness, and **n**euroticism.

4.1.2 Utilizing Personality in Cognitive Agents

Personality is one attribute that determines a human's performance in specific tasks (cf. [MDWS00]). With the defined model and the ability to map psychological studies to it, the same approach can be used for virtual humans. Therefore, *task parameters* are introduced in the CA^{2}RVE architecture. These parameters consolidate high-dimensional personality profiles into a single continuous parameter used by cognitive processes, such as perception or decision-making. With the help of personality profiles and task parameters, it is possible to define an agent's individualized behavior for specific tasks or task domains. How a profile's dimensions are transformed into a task parameter depends on the study or theory used to correlate behavior and personality. Alternatively, transformations can be defined without any underlying evidence, but could instead be used to define intended behavior patterns or prototypes, e.g., for specific agent types. A straightforward mapping from a FFM profile $\mathbf{p}$ of an agent a to a task parameter α is shown in the following:

$$\begin{aligned} \mathbf{c} &= \langle c_1, c_2, c_3, c_4, c_5 \rangle \in \mathbb{R}^5 \\ \alpha(a) &= \mathbf{c} \bullet \mathbf{p} \end{aligned} \tag{4.8}$$

where $\mathbf{c}$ is a 5-tuple of coefficients, which can be derived from a personality study using appropriate range limits for coefficients and the task parameter and $\bullet$ denotes the dot product. A similar mapping is used in a sample realization of the CA^{2}RVE architecture to define a politeness parameter, which influences the decisions of traffic agents (see Chapters 6 and 7). This parameter weighs an agent's own advantage against other agents' disadvantages that would result from its actions in lane changing and right-of-way scenarios. Coefficients were determined using a study by Herzberg, which investigated the correlation between personality and driving behavior using the FFM [Her09]. He identified three personality

prototypes that predict accident proneness and driving behavior. These prototypes and their associated driving behavior were the basis for choosing coefficients for the politeness parameter. To design and assign personality profiles that are consistent with the prototypes reported by Herzberg, the study was mapped to the definition introduced above. Herzberg provided only relative descriptions of trait scores within the three prototype classes, but references another of his studies that provides more details [HR06]. This second study was used for the mapping, but it included only the z-scores for each prototype profile. To be utilized within a realization of the concept, a fully defined study $\mathscr{S}$ is derived from the z-scores using normalization, scaling, and assumptions about the underlying trait value ranges. For details about the transformation process, please refer to [IVC13]. Using these transformations, the following theoretical study $\mathscr{S}_{FFM}$ is one example that can be used to generate personality profiles for an implementation:

$$
\begin{aligned}
&\mathscr{S}_{FFM} = \langle P, C, l, h \rangle \text{ with} \\
&\quad P = \{\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4, \mathbf{z}_5\}, \\
&\quad \mathbf{z}_1 = \langle 0.15, 0.66, 0.6, 0.48, -0.84 \rangle, \mathbf{z}_2 = \langle -0.09, -0.24, -0.75, -0.06, 0.9 \rangle, \\
&\quad \mathbf{z}_3 = \langle -0.06, -0.6, -0.12, -0.54, 0.3 \rangle, \mathbf{z}_4 = \langle 0.48, 0.18, 0.36, 0.12, -0.03 \rangle, \\
&\quad \mathbf{z}_5 = \langle -0.39, 0.15, -0.18, 0.15, -0.21 \rangle, \\
&\quad C = \{\mathscr{C}_1, \mathscr{C}_2, \mathscr{C}_3, \mathscr{C}_4, \mathscr{C}_5\}, \\
&\forall \mathscr{C}_i \in C : \mathscr{C}_i = \langle \{\mathbf{z}_i\}, \mathbf{z}_i, \mathbf{z}_i \rangle, \\
&\quad l = -1, h = 1, \\
&\quad \mu = \langle 0, 0, 0, 0, 0 \rangle, \sigma = \langle 1, 1, 1, 1, 1 \rangle
\end{aligned}
\tag{4.9}
$$

Note that each class $\mathscr{C}_i$ includes only one profile, which is at the same time the prototype profile and the associated z-score profile of that class. This circumstance is due to the performed transformations. In the context of an application, this is all that is required as agent profiles are generated using the prototype profiles. In other words, all sets Q_i for each class are filled during a simulation, every time a new agent is created.

4.2 Emotion Model

Integrating a personality model including task parameters into the proposed architecture provides a way of generating individual but consistent behavior patterns for agents, which answers **RT2.1**. However, when provided with the same task in an unchanging situation, a personality-based agent will continue to exhibit the same behavior. For example, a polite agent may continue waiting behind a parked vehicle to not interfere with oncoming traffic.

While the decision may be consistent with the agent's previous behavior at first, it will eventually become implausible to an observer if the agent's situation is not resolved. This consideration leads to **RT2.2**: Adapting personality-based behavior to an agent's current situation. Emotions – short-lived responses to events in the environment – are introduced as one possible solution to this task. They provide a dynamic component that allows the agent to change its typical behavior if a situation calls for it. An agent's current emotions temporarily influence its personality profile, which changes the derived task parameters and lastly the generated behavior. This allows agents to adjust to encountered situations that they would not be able to solve with their static behavior. While there are many models for representing emotion in virtual humans (e.g., see [KMT08, NFdSS10]), a model is selected for this thesis, whose correlation to the chosen personality model has been confirmed. Since the CA^{2}RVE architecture uses the FFM and Watson and Clark [WC92b], Rusting and Larsen [RL97], and Yik and Russell [YR01] are examples that correlated FFM profiles with a two-dimensional emotion model, the same number of dimensions is considered here to distinguish negative and positive emotions. An additional advantage of a two-dimensional model is its moderate complexity compared to other common models, like the OCC [KMT08]. At the same time, it provides sufficient utility to intuitively map events to changes in the emotional state while remaining accessible and comprehensible. If more complexity or comparability to other approaches is required, the model could easily be extended to more than two dimensions. In the following, a formal representation of the emotion model is introduced before describing how emotions are experienced, how they fade over time, and how they influence behavior within the architecture.

4.2.1 Representing Emotions

Like the representation of personality profiles, an emotional state is defined as a tuple, but with only two elements:

$$\mathscr{E} = \langle \mathbf{e}_1, \mathbf{e}_2 \rangle \in \mathbb{R}^4 \times \mathbb{R}^4 \tag{4.10}$$

Unlike personality profiles, each dimension of an emotional state $\mathscr{E}$ is itself a 4-tuple:

$$\mathbf{e} = \langle e_1, e_2, e_3, e_4 \rangle \in \mathbb{R}^4 \tag{4.11}$$

The element e_1 defines the current emotion by a numerical value. Values e_2, e_3, and e_4 are required to describe how emotion is faded back to a neutral state, which is described in Sections 4.2.2 and 4.2.3. In a realization, all values should be within ranges in accordance with the chosen personality dimension and task parameter ranges. Since the proposed model differentiates between positive and negative emotions, the following set of functions

is defined to provide access to both dimensions of the emotional state:

$$\begin{aligned} E &= \{\mathrm{e}^-, \mathrm{e}^+\} \text{ with} \\ \mathrm{e}^- &: \mathbb{R}^4 \times \mathbb{R}^4 \to \mathbb{R}^4, \mathscr{E} \mapsto \mathbf{e}_1, \\ \mathrm{e}^+ &: \mathbb{R}^4 \times \mathbb{R}^4 \to \mathbb{R}^4, \mathscr{E} \mapsto \mathbf{e}_2 \end{aligned} \tag{4.12}$$

Since the model was defined such that positive and negative emotions are independent, it is possible to model each dimension's influence on behavior separately. For example, in a case where personality is modeled using the FFM, negative emotions could be used to increase an agent's neuroticism, but positive emotions would not decrease neuroticism. Instead, positive emotions could increase an agent's agreeableness. Note that the influence of each dimension on a personality profile can be considered independently, but since both negative and positive emotions are applied to a profile, their effects are not independent.

4.2.2 Experiencing Emotions

In a simulation, an agent would perceive information from the environment that induces an emotional response. To model this process, experiences that are emotionally relevant to an agent are modeled as incidents $\mathbf{i}$. An incident controls how the current emotional state of an agent is modified whenever the event occurs. Thus, an incident is of the same dimensionality as the emotional state. For the current example, this translates to:

$$\begin{aligned} \mathbf{i} &= \langle i_1, i_2 \rangle \in \mathbb{R}^2, \\ \mathrm{i}^- &: \mathbb{R}^2 \to \mathbb{R}, (\mathbf{i}) \mapsto i_1, \\ \mathrm{i}^+ &: \mathbb{R}^2 \to \mathbb{R}, (\mathbf{i}) \mapsto i_2 \end{aligned} \tag{4.13}$$

It is the designer's responsibility to define meaningful incidents. For example, incidents that induce the same amount of positive and negative emotion make little sense.

As indicated before, personality influences the entire cognitive apparatus, including the perception of emotions. This intuitive relationship has been confirmed by studies, such as that by Watson and Clark [WC92a], which showed that tendencies of perceiving specific emotions is correlated to a subject's personality. To model this tendency, the following n-tuples of coefficients are defined; one tuple for each dimension of the emotional state.

$$\mathbf{s}^- = \langle s_1^-, \cdots, s_n^- \rangle \in \mathbb{R}^n, \quad \mathbf{s}^+ = \langle s_1^+, \cdots, s_n^+ \rangle \in \mathbb{R}^n \tag{4.14}$$

The dimensionality n of both tuples must correspond with the personality model that is used. To provide a definition for perceiving emotions, a function s is required, which determines how each emotion dimension $\mathbf{e}$ is updated. It updates the current emotion value e_1, as well as the first fading value e_2, but does not alter the other fading values, e_3 and e_4. The reason

for not changing e_3 and e_4 and the role of all fading values is explained later in Section 4.2.3.

$$\mathrm{s} : \mathbb{R}^4 \times \mathbb{R} \to \mathbb{R}^4, (\mathbf{e}, x) \mapsto \langle x, x, e_3, e_4 \rangle \tag{4.15}$$

Using the above definitions, a perception function $\mathrm{w}(a, \mathbf{i})$ transforms the emotional state $\mathscr{E}_a^t$ of an agent a from the set of all agents A at time t to a new emotional state $\mathscr{E}_a^{t'}$ at time t'.

$$\begin{aligned}
\mathrm{w} : A \times \mathbb{R}^2 \to \mathbb{R}^4 \times \mathbb{R}^4, (a, \mathbf{i}) = \mathscr{E}_a^{t'} \text{ with} \\
\mathrm{e}^-\left(\mathscr{E}_a^{t'}\right) &= \mathrm{s}\left(\mathrm{e}^-\left(\mathscr{E}_a^t\right), \mathrm{v}\left(\mathrm{e}^-\left(\mathscr{E}_a^t\right)\right) + \max\{l^-, c^-\}\right) \\
l^- &= l_i \cdot \mathrm{i}^-(\mathbf{i}) \\
c^- &= c_i \cdot \mathrm{i}^-(\mathbf{i}) \cdot (\mathbf{s}^- \bullet \mathbf{p}_a) \\
\mathrm{e}^+\left(\mathscr{E}_a^{t'}\right) &= \mathrm{s}\left(\mathrm{e}^+\left(\mathscr{E}_a^t\right), \mathrm{v}\left(\mathrm{e}^+\left(\mathscr{E}_a^t\right)\right) + \max\{l^+, c^+\}\right) \\
l^+ &= l_i \cdot \mathrm{i}^+(\mathbf{i}) \\
c^+ &= c_i \cdot \mathrm{i}^+(\mathbf{i}) \cdot (\mathbf{s}^+ \bullet \mathbf{p}_a) \\
l_i, c_i \in \mathbb{R}, 0 < l_i \leq 1, c_i > 0
\end{aligned} \tag{4.16}$$

The parameters l_i and c_i prevent combinations of correlation coefficients $(\mathbf{s}^-, \mathbf{s}^+)$ and the current personality profile configuration to cause *negative experiences*. A negative experience is defined as an incident causing the opposite of its declared effect. For example, if the effect of an incident on the positive dimension of the emotional state is positive (i.e., $\mathrm{i}^+(\mathbf{i}) > 0$), perceiving the incident with certain combinations of coefficients and personality could decrease positive emotion instead of increasing it. The parameter l_i defines the minimum effect of any incident that is experienced, regardless of the current profile or the configuration of the correlation coefficients. With c_i an additional tool is provided for calibrating the model. Consequently, l^+ and l^- are the minimal effect of the current incident and c^+ and c^- are the calibrated incident values multiplied by the personality influence, which is the dot product of the personality profile and the correlation coefficients. The function $\mathrm{v}(\mathbf{e})$ identifies the current emotional value of an emotion dimension tuple:

$$\mathrm{v} : \mathbb{R}^4 \to \mathbb{R}, (\mathbf{e}) \mapsto e_1 \tag{4.17}$$

Using all the above, an emotional state can be assigned to each agent and the state can be changed by experiencing incidents that bear a predefined emotional meaning. The incident's effect can be adjusted using calibration values and is further influenced by an agent's personality. For example, a neurotic agent may experience negative emotions more intensely than others, while extroverts may influence more by positive emotions.

4.2.3 Fading Emotions

Emotions are short-term, immediate reactions to experienced events (cf. [Tha89]). Thus, their effect on the cognitive processes of an agent should vanish over time. For this purpose, the emotional state is returned to a neutral state by modeling a fade function f that transforms an emotion dimension $\mathbf{e}^t$ at time t to a new tuple $\mathbf{e}^{t'}$ at time t':

$$\begin{aligned} &\mathrm{f} : \mathbb{R}^4 \times \mathbb{R}^4, (\mathbf{e}^t) \mapsto \mathbf{e}^{t'}, \text{ where} \\ &\qquad e_1^{t'} = \max\{0, \mathrm{g}(\mathbf{e}^t), \mathrm{h}(\mathbf{e}^t)\} \\ &\qquad e_2^{t'} = e_2^t \\ &\qquad e_3^{t'} = e_3^t \\ &\qquad e_4^{t'} = e_4^t \end{aligned} \tag{4.18}$$

The curve of the fading is controlled by a set of parameters encoded within the emotion dimension as indicated in Section 4.2.1 (e_3 and e_4) and a pair of functions (g and h).

$$\begin{aligned} &\mathrm{g} : \mathbb{R}^4 \to \mathbb{R}, (\mathbf{e}) \mapsto e_1 - e_3 \\ &\mathrm{h} : \mathbb{R}^4 \to \mathbb{R}, (\mathbf{e}) \mapsto \frac{e_1^2}{e_2 + 10^{-e_4}} \\ &0 \leq e_1, e_2 \leq 1, e_3, e_4 \in \mathbb{R}^+ \backslash \{0\} \end{aligned} \tag{4.19}$$

Only the first element of an emotion dimension tuple is changed by the fading function. However, the other elements are also part of the tuple because it may be desirable to control the fading for each dimension separately. For example, the model may be configured in such a way that negative emotions fade slower than positive emotions. The first element e_1 represents the current emotion value as stated before. This element can be set together with e_2 using the function defined in Equation 4.15. While e_1 will change with the fading function 4.18, e_2 is not altered as it represents the base value of the emotion, i.e., the value where the fading started. The additional elements of the emotion dimension tuple (e_3 and e_4) are required to control functions g and h. No scientific evidence for the dissipation of emotional experiences could be found during this thesis project. Therefore, g and h were defined empirically with the intention of providing a plausible model. Due to the short-term nature of emotions, it is reasonable to assume that a human is influenced strongly by an emotion immediately after experiencing an event that caused it. After some time passes, the effect of the emotion will start to fade. Thus, function g models a straightforward linear decrease with e_3 (linear parameter) controlling how fast this process progresses. Function h models how long the emotional effect is sustained as the resulting value will stay nearly constant before rapidly decreasing to zero. The duration of this effect is controlled by e_4 (sustain parameter) in combination with the result of g, because as soon as the decrease

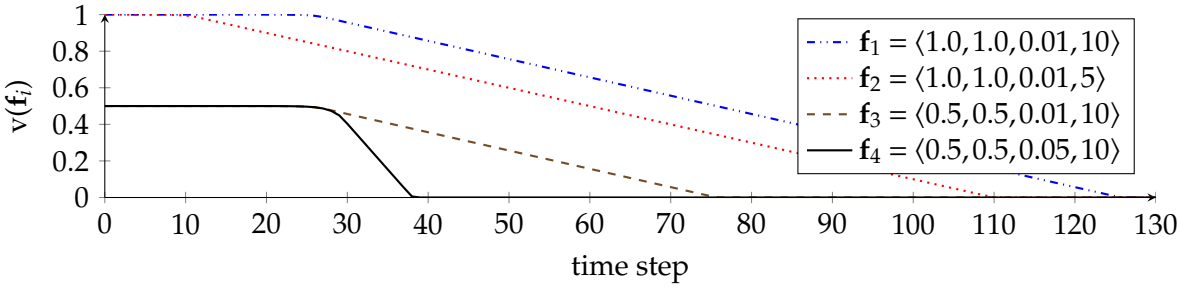


Figure 4.1: Curve shapes of the fading process for examples of emotion tuples $\mathbf{f}_1$ to $\mathbf{f}_4$. All examples except $\mathbf{f}_4$ have the same linear fading value (0.01). Thus, $\mathbf{f}_4$ illustrates the steepest decline, because of its larger value. Due to their identical sustain parameters, values start decreasing linearly at the same time. The exception is $\mathbf{f}_2$, which is the first to decrease linearly because of a smaller sustain value (5). Figure based on [SvS12].

calculated by h surpasses the decrease calculated by g, the linear decrease defines how the emotion is faded going forward (s. Equation 4.18). To generate meaningful fading processes, it is necessary to choose e_3 and e_4 according to defined ranges for personality profiles and emotion dimensions. In the proof-of-concept realization described in Chapter 6, e_1, e_2, and e_3 are within the range $[0;1]$ and e_4 is defined to be greater than 0. Figure 4.1 shows four examples of fading an emotional value over time.

4.2.4 Utilizing Emotions in Cognitive Agents

The intention of modeling emotions is to introduce a dynamic aspect to behavior generation. As mentioned in the previous chapter, it is beneficial to introduce an intermediate personality layer that is generated from the original static profile of an agent. The entire layer can be omitted, if necessary, without altering anything else within the architecture. Additionally, the concept of task parameters as introduced in Section 4.1.2 already defines a means to influence behavior. Using a profile altered by an emotional state leverages the existing concept, existing decision processes, and potentially existing parameters without having to define individual parameters for each cognitive module or even each emotional event.

To generate a dynamic profile $\mathbf{b}$ from the agent's original profile $\mathbf{p}$, additional n-tuples are required for each dimension of the emotional state. In the considered case this means one tuple for positive and one tuple for negative emotions:

$$\begin{aligned} \mathbf{t}^- &= \langle t_1^-, \cdots, t_n^- \rangle \in \mathbb{R}^n, \\ \mathbf{t}^+ &= \langle t_1^+, \cdots, t_n^+ \rangle \in \mathbb{R}^n \end{aligned} \tag{4.20}$$

Using these coefficients an agent a's personality profile $\mathbf{p}_a$ can be transformed to its dynamic

profile $\mathbf{b}_a$ based on the current emotional state $\mathscr{E}_a$:

$$\begin{aligned} &\forall \mathrm{d} \in D : \mathrm{d}(\mathbf{b}_a) = \mathrm{d}(\mathbf{p}_a) + \mathrm{n}(\mathscr{E}_a, \mathrm{d}) + \mathrm{p}(\mathscr{E}_a, \mathrm{d}), \\ &\mathrm{n} : \mathbb{R}^4 \times \mathbb{R}^4 \times D \to \mathbb{R}, (\mathscr{E}, \mathrm{d}) \mapsto \mathrm{v}(\mathrm{e}^-(\mathscr{E})) \cdot \mathrm{d}(\mathbf{t}^-) \\ &\mathrm{p} : \mathbb{R}^4 \times \mathbb{R}^4 \times D \to \mathbb{R}, (\mathscr{E}, \mathrm{d}) \mapsto \mathrm{v}(\mathrm{e}^+(\mathscr{E})) \cdot \mathrm{d}(\mathbf{t}^+) \end{aligned} \tag{4.21}$$

Depending on the value ranges of the static personality profile and the study $\mathscr{S}$, the transformed dynamic profile $\mathbf{b}_a$ may have to be adjusted to fit theses ranges. A straightforward solution is to clamp transformed values at the original range limits (l and h) of the study $\mathscr{S}$. As a result, the dynamic profile $\mathbf{b}_a$ can be used in any of the cognitive processes without adjustments as it is structurally identical to the original profile $\mathbf{p}_a$.

Using the definitions above, the personality of an agent influences how it perceives emotional incidents, i.e., how its emotional state changes by experiencing defined incidents. The current emotional state is used to transform the agent's original static personality profile into a dynamic profile. This correlation produces the desired interdependency between personality and emotion as the dynamic profile will change how emotions are perceived by the agent. Figure 4.2 shows an exemplary progression of all traits of a dynamic personality profile when invoking an emotional incident of $\mathbf{i} = \langle 0.1, 0 \rangle$ at each time step. All values were chosen for demonstration purposes only and values for positive emotions are omitted since the incident only includes negative emotions.

$$\begin{aligned} \mathbf{p}_a &= \langle -0.25, -0.6, 0.12, -0.54, 0.3 \rangle \\ \mathbf{t}^- &= \langle -0.1, -0.3, 0, -0.75, 1 \rangle \\ \mathbf{s}^- &= \langle 0, 0, 0, 0, 0.341 \rangle \\ l_i &= 0.35 \\ c_i &= 2 \end{aligned} \tag{4.22}$$

4.3 Conclusion

In this chapter, two research tasks were examined that were derived from the second research question: **Can the agent architecture systematically generate individual, dynamic agent behavior?** It was considered how behavior can be individualized to each agent (**RT2.1**) and how a dynamic component can be added to adapt this behavior (**RT2.2**). The proposed solution contained two aspects: a static trait-based personality profile and a dynamic emotional state influencing the profile. The static profile was defined, first in a generalized manner allowing an arbitrary number of traits and second, specified to the common Five Factor Model (FFM). The formal description of a *study* was provided, which allows the integration

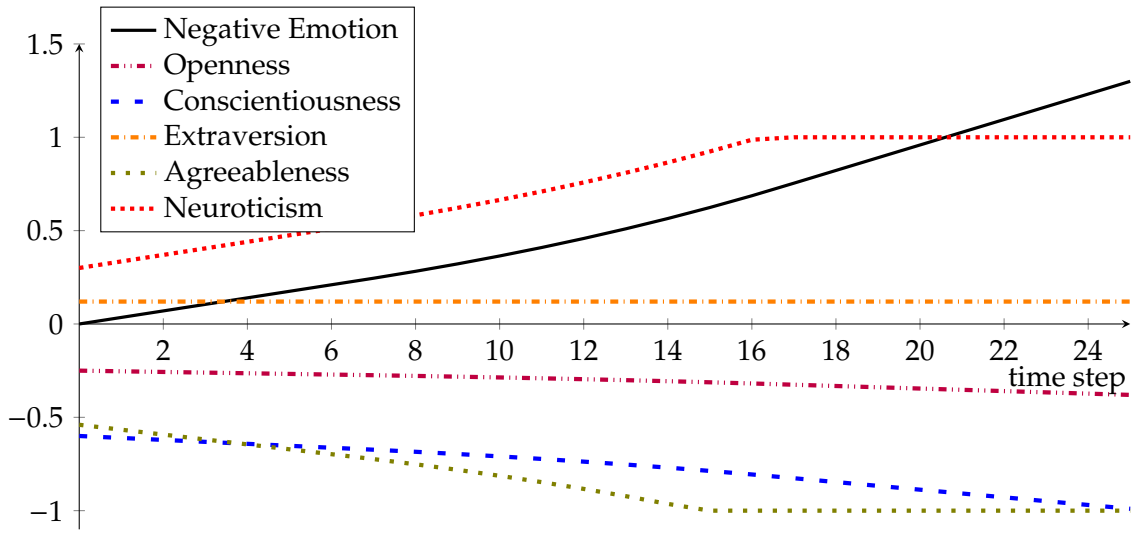


Figure 4.2: Exemplary progression of an agent's dynamic FFM personality profile. At each time step an emotional incident $\mathbf{i} = \langle 0.1, 0 \rangle$ is perceived, resulting in an increase of negative emotion and neuroticism according to their defined correlation. When the range limit of the personality profile is reached at time step 16, neuroticism no longer increases and the negative emotion increases linearly. All other traits except extraversion decrease according to the defined correlations. Figure based on [IVC13].

of personality profiles into realizations of the architecture concept. Additionally, it provides the opportunity to use psychological studies as input for behavior generation, e.g., to emulate real-life behavior patterns. Task parameters were introduced to define the influence of personality on other cognitive processes such as perception or decision-making.

Using task parameters, different personality prototypes can be used to generate distinguishable behavior. However, the generation of behavior from these prototype profiles depends on multiple factors. Task specific or situational behavior is linked to a personality prototype, which means the more prototypes are defined, the more differences can be expressed. The number of choices (i.e., the actions to choose from) in a certain situation directly affects how well behavioral differences can be expressed. Furthermore, the agents' capabilities influence the observability of behavioral differences, i.e., the possible parameterization of available actions. For instance, assume the chosen action is to accelerate a vehicle. Based on personality, one agent chooses a higher acceleration than another agent would, but because they drive the same type of car the final acceleration may be identical. In this case, the difference generated by the profile cannot be observed. Due to the complexity and effort required for providing a diverse set of options for the mentioned factors, only a limited range of behavior can be realized within the scope of this thesis. Nevertheless, the presented definition of personality profiles and their integration within a cognitive agent architecture

provides a general tool for individualizing behavior. The contribution of this claim towards answering research question 2 will be demonstrated in Chapters 6 and 7.

A two-dimensional emotion model adds a dynamic component to static profiles. An agent's emotional state consists of negative and positive emotion dimensions, which are independent of each other. This independence is useful to model their effects separately. Formal descriptions were provided that define how emotions are perceived and faded. Perception of emotions depends on *incidents*, that describe an event's effect on an agent's emotional state. N-Tuples of coefficients relate an agent's personality to its perception of these incidents. While personality is a stable component, emotions are temporal reactions to experiences whose effects will fade with time until the neutral emotional state is reached.

Other emotion models, e.g., the OCC model of emotion, can represent a wider spectrum of emotions (cf. [KMT08]). However, the objective of enabling agents to adapt their personality-based behavior to their current situation was achieved with the presented model (**RT2.2**). A dynamic personality profile is used to incorporate an agent's emotional state into its other cognitive processes. The emotional state modulates an agent's static personality traits, forming an intermediate, dynamic profile, which influences subsequent processes identical to the static case. This technique circumvents the need for encoding influences of personality and emotion separately for each cognitive module. Additionally, it is straightforward to exclude emotional states if an application requires it, without having to adjust other process. Since perception of emotions depends on personality and the emotional state creates a new personality, an interdependency between emotion and personality is achieved.

An ideal solution to adaptive behavior involves advanced inference and decision-making processes. The ability to improvise when faced with unknown situations, i.e., to deviate from pre-programmed behavior, would be a significant step towards human-like behavior. Due to the complexity of this endeavor, it is currently unclear whether this level of autonomy can be achieved at all. The proposed approach may not generate true "alternative behavior", in the sense indicated above, but the intended objective is achieved as the emotional state of an agent changes its typical, i.e., "default" behavior if the agent's selected actions do not resolve a problematic situation. This capability of the agent architecture is evaluated in Chapter 7.

5

Perceptual Agents

"We must look at the lens through which we see the world, as well as the world we see, and that the lens itself shapes how we interpret the world."

- Stephen R. Covey

Making decisions within an environment and predictions about the future state of an environment requires gathering knowledge about the environment's current state (cf. [KT18]). Humans must accumulate knowledge about their environment and its current state via perception. Deliberation processes additionally augment this information using, e.g., previous knowledge. In contrast, virtual agents generally have access to their entire environment, which may further be augmented by semantic information (cf. [GLBV08]). Thus, it may seem unnecessary to model human-like processes that gather perceptual information about the agent's world. However, creating, simulating, and maintaining all-knowing agents quickly becomes a problem if combinatorial consequences and more importantly the quality of depicted behavior are considered [PT97]. Furthermore, it is generally agreed upon that using perception to form a plausible world model for agents is the foundation for believable agent behavior [Blu97, POS03, PCR+11, Rey87, vO14]. This process includes organization, identification, and interpretation of sensory information as well as finding an appropriate representation [SGW12]. It is therefore not surprising that modeling perception is a standard in developing agents and virtual characters across different research domains, such as cognitive architectures (e.g., [Lai08, Lai12, RTO19]), virtual humans (e.g., [LA00, PCR+11, vO14]), and digital games technology (e.g., [Leo03, LB19, Pal18]).

Despite the common consideration of perception, it is often only utilized to collect data with limited sensor capabilities to increase believability, especially in interactive applications. One important component of human perception that is often neglected in agent

perception is attention. A virtual attention process allows agents to identify objects of importance regarding both reactive behavior and cognitive processes. Furthermore, synthetic attention can improve computational efficiency by reducing the amount of sensory data that is processed by an agent – comparable to human attention. However, attention is often only used to simulate external indicators like eye gaze to improve a user's impression of a virtual character [APMG12, BA13, Kim06, PAGM15]. Consequently, available approaches for virtual humans are mostly focused on bottom-up attention (i.e., reflexive attention) and are not considered on a cognitive level [CKH+15, YT15]. To convey plausible and application specific behavior, agents must appear target oriented. Models operating on a cognitive level could guide task-oriented attention mechanisms (i.e., top-down attention) to create that impression. Neurobiological models provide important, realistic clues about how and why attention is directed towards specific parts of a scene, but exceed the requirements of real-time, multi-agent environments (c.f. [KW15]). Solutions for interactive environments typically trade precision for efficiency often resulting in less realistic results or focus on certain aspect, like simulating eye gaze (e.g., [PAGM15]).

Perception and attention are important aspects of modeling agent behavior for aesthetic and/or functional reasons [PCR+11]. These two reasons are also requirements in this thesis project (i.e., believability and scalability). In addition to these two reasons, a third aspect is added within the context of the work presented here. Agents in virtual environments usually fulfill a specific purpose like emulating a living virtual world or supporting a training goal. Consequently, the agents' behavior must not only be plausible and computable in real-time, but it must also be controllable. Therefore, the perception module must facilitate authoring user-centric, goal-driven agent behavior. Furthermore, existing approaches usually do not consider an agent's current state of mind as part of the perception process. However, these aspects influence human perception and attention processes [Bad03, PCR+11, BSGS13, Han89] suggesting an integration into synthetic perception for agents as well.

Based on the described situation, the third research question – **What is the role of perception and attention in behavior generation for cognitive agents?** – should be answered by considering the two following research tasks. The main objectives are to clarify how and where perception and attention fit into the developed cognitive architecture and how they can be utilized to support the three requirements: believability, controllability, and scalability.

RT3.1: A flexible framework for synthetic perception in a cognitive agent architecture.

Starting from the large quantity of available approaches to agent perception, it should be determined what the core characteristics of synthetic agent perception are and how they can be assembled and extended to build a flexible framework. Furthermore, to integrate well with the proposed cognitive agent architecture, the perception framework must support the architecture's main principles. The most important principle being that all cognitive processes are based on an agent's personality and emotional state. The influence of personality and emotion on cognitive processes was described in Section 4.2 and an example was provided that utilized the influence for decision-making. A similar approach should be applied to perception. At the same time, a resulting perception module should follow the same modular structure as the architecture, i.e., it should integrate well with the other cognitive modules, and it should be modular itself for maximum flexibility.

RT3.2: Perception and attention for virtual agents considering plausibility, controllability, and real-time performance.

Simply combining existing agent perception approaches is not sufficient due to the discussed shortcomings. Addressing these shortcomings, it should be possible to design a synthetic perception framework that contributes to believability, but supports controllability, e.g., by facilitating customization and extension, without compromising scalability. An open sensor interface should allow an agent to perceive multiple stimuli emanating from a single object. These stimuli could be received either through different sensor modalities or by multiple sensors of the same modality. The approach should also not exclude the integration of additional "supernatural" information sources, like world semantics. Such cues can be modeled to provide simplifications where appropriate or necessary, e.g., to increase computational performance. Sensors should also be parameterizable to allow balancing performance against accuracy, supporting believability while regarding real-time constraints. However, parameterization especially facilitates application-specific solutions. The same applies to attention, where customizable dwell-time, inhibition, and memory capacities could be used to individualize the perception process for each agent and provide control over the process to designers. The customization on various levels of the framework should also scale well to different application scenarios providing an efficient and controllable approach. The framework should further combine bottom-up and top-down attention independent of utilized decision processes or sensor implementations to generate believable and controllable behavior. In summary, the provided perception mechanisms should support the objectives without restricting them to a specific approach or a certain complexity.

5.1 Synthetic Perception for Cognitive Agents

To provide a synthetic perception module that integrates well with the developed cognitive agent architecture and fulfills the associated requirements, the following aspects of perception should be addressed:

I. A unified interface to allow for the implementation and attachment of arbitrary virtual sensors.
II. A mechanism to accumulate and process environmental stimuli, i.e., virtual sensors.
III. A mechanism to filter aggregated stimuli, e.g., attention.
IV. A structure to store stimuli and percepts on various levels of abstraction, i.e., a memory hierarchy.
V. A procedure to influence perception and attention based on an agent's personality and current emotional state.
VI. An interface for *high-level* processes to acquire and process information.

Based on several publications from cognitive architecture, cognitive science, and IVA/ECA research, a perceptual framework for cognitive agents should address at least sensing (II.), attention (III.), and memory (IV.). These three aspects are the focus of the presented perception framework for cognitive agents, but all six aspects are realized.

A generalized perception framework must support a variety of sensor modalities. These should ideally include the five main human senses (sight, hearing, touch, smell, and taste) and perhaps even more to perceive application-specific stimuli (e.g., semantic information). Since sight is the most important and most studied human sense [Enn04, Fri11, KT18], the focus of this work is on the integration of visual information into the synthetic perception process. However, the integration of multiple sensor modalities is also discussed. Additionally, a model of attention is considered; an important component of human perception that is often neglected in other perception approaches for agents.

The main purpose of the perception framework discussed in this chapter is to support the cognitive agent architecture in plausibly recreating human behavior regarding various aspects of perception. To generate plausible behavior in a complex virtual environment, additional *high-level* concepts are required, such as decision-making, planning, and navigation. The presented framework is part of the cognitive agent architecture described in Chapter 3. Examples of high-level processes are given in Chapters 6 and 7. This section describes how the proposed perception framework fits into the overall agent architecture. An isolated view of the perception aspect is presented in Figure 5.1.

Details of the presented perception framework are described throughout this chapter. The focus is on the three identified major perceptual components: virtual sensing via a

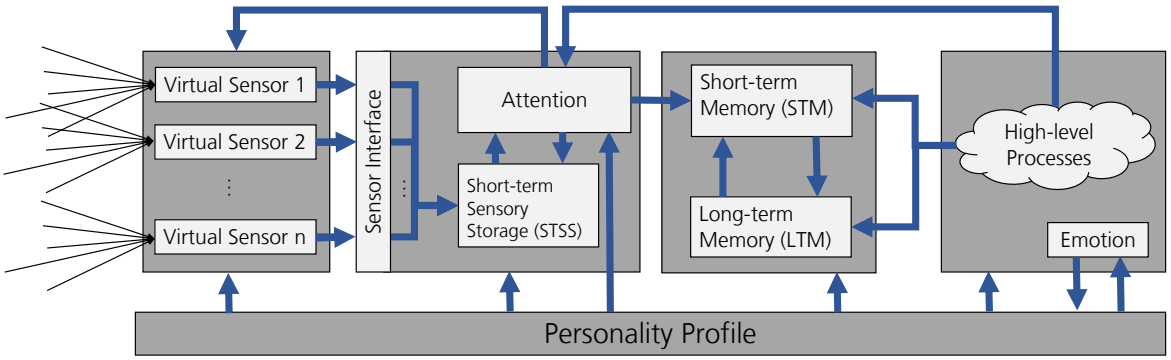


Figure 5.1: Virtual sensors gather stimuli from an environment which are delivered to a short-term sensory storage (STSS). From the unfiltered information, objects of interest are selected and inhibited by an attention process. Attended objects become percepts residing in short-term memory (STM). From there, percepts can be retrieved by high-level processes, e.g., decision-making. The attention mechanism can request orientation of sensors and high-level processes can bias attention for top-down perception. An underlying personality profile, which is affected by an emotion component, can influence all perceptual processes and elements. A long-term memory (LTM) can store information that need to persist across multiple simulation sessions. Image based on [SvS22].

unified sensor interface, attention modeling, and memory layout. At the same time, the three agent architecture requirements – plausibility, controllability, and scalability – are constantly considered in the design. Furthermore, the integration of a personality profile, combined with an emotion model, is discussed and how these aspects can be used to further customize the generation of behavioral patterns. Considerations of efficient information retrieval from the virtual environment are included in the framework at an abstract level. In Chapter 7, sample implementations are presented to evaluate and demonstrate the framework's capabilities in real-time game engine scenarios.

5.2 Perception Cycles

The perception framework proposed in this thesis is designed to coarsely resemble the process of human perception. Therefore, it is based on the perceptual cycle as described by Goldstein and Brockmole [GB16]. A distal stimulus, originating from the environment and reaching a sensory organ, can be considered the starting point of the cycle. Since it is caused by something in the environment, the stimulus is referred to as *environmental stimulus*. Sensory organs process these stimuli until they reach their according primary receiving areas in the brain. Up to this point, the processed information still represents the environmental stimulus. This complex process is simplified and modeled as the *environmental stimulus cycle* (ESC) within the proposed perception framework. The ESC describes the extraction

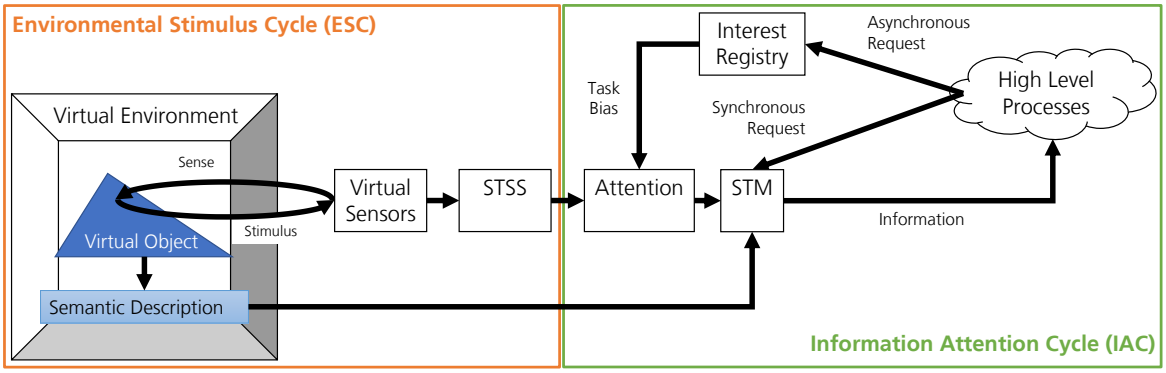


Figure 5.2: Abstract overview of the perception process. The left side represents the agent's connection to its environment (the environmental stimulus cycle). The right side describes the agent's internal information retrieval process (the information attention cycle).

of stimuli from the environment via virtual sensors. From their primary receiving areas, signals are transmitted to various brain areas resulting in conscious experiences where the perceiver becomes aware of the stimulus' origin and recognizes it. This process is categorized here as *perception* and modeled by the *information attention cycle* (IAC) as an internal information retrieval process. In human perception, knowledge is another important factor that influences many steps in the perceptual cycle, e.g., to label objects or fill in missing information. This type of knowledge-based processing is also known as *top-down processing*. The counterpart is *bottom-up processing* or data-based processing which is solely based on stimuli reaching a sensor. Both directions and their importance to behavior generation will be revisited later. An overview of the ESC and IAC is provided in Figure 5.2. The structural components of theses cycles are described throughout the course of this chapter.

The ESC connects the agent to the world it inhabits. In reality, information from objects reaches the sensors in the form of stimuli. To faithfully represent this effect, stimuli would have to be sent from every object to every possible direction. Even if discretized, only a small fraction of generated stimuli would reach a sensor. Therefore, the process is typically inverted: Sensors determine sensible objects in the environment by sending signals, which determine if an object emits stimuli that match the sensor's modality and ability. From the response, corresponding stimuli are extracted and stored in a short-term sensory storage.

The IAC acquires information, obtained via the ESC, that the agent is currently interested in. The main component of this process is the short-term memory (STM) representing the agent's current knowledge. High-level processes can actively query the STM for information they require as a synchronous request. Alternatively, these processes can register their interest for specific types of objects with an *interest registry* and receive a notification when objects of that type are perceived. An advantage of this passive approach is that the registry

can be used to bias an agent's attention module to raise the importance of registered object types (see also Section 5.5).

Goldstein and Brockmole's perceptual cycle [GB16] is called a cycle because the outcome of the process influences the input. The ESC and IAC represent continuous processes defining how information about virtual objects are handled. They do not represent processes that exert influence on themselves like the perceptual cycle. This cycle is only implicitly included in this view since the consequences of an agent's action selection process are not explicitly considered. For example, based on the attention module's stimulus selection, the agent may choose to orient its primary sensors towards the associated virtual object. This action likely causes the sensors to sense different information, which in turn might affect following modules and processes. As indicated before, the ESC and IAC also represent a separation between low-level processes of the sensory system (*sensation*) from high-level brain functions used to interpret sensed information (*perception*). While this distinction can still be found in the literature, a clear separation is often not possible. Consequently, modern perception research does not differentiate between those terms (cf. [GB16]) or argues that they cannot be separated at all (e.g., see [Enn04]). However, in the context of this thesis it is beneficial to uphold these separate views to define strict responsibilities and interfaces between components and modules as illustrated by the ESC an IAC.

5.3 Sensing

Sensing is part of the ESC and must occur before an agent is able to utilize any information provided by its sensors. Many techniques are possible and available to provide virtual sensing capabilities for an IVA. However, instead of choosing one specific sensor set, a unified sensor interface represents a solution that is open to customization and extension. Similar approaches can be found in the work of Conde and Thalmann [CT04, CT06] as well as Kuiper et al. [KW13, SKW10a, SKW10b]. From an architectural point of view, the common interface represents a single entry point where stimuli enter an agent's mind. Therefore, the number and type of sensors is unspecified and open to an application developer's interpretation or adjustable to requirements of a specific use case. For example, depending on the specifications of an application, sensors can provide complementary information for maximum coverage, or they can provide redundant information for increased reliability. Sensors may also vary in complexity. Some situations may demand realistic results, requiring sensor implementations to mimic existing biological sensory system as closely as possible. However, the more complex such a set of sensors becomes, the more computational resources are required to simulate perception. Especially in the realm of virtual environments, where

interactivity is a key requirement, the cost of emulating neuro-biologically correct perception often outweighs its benefits. Consequently, interactive applications typically require more efficient, less complex solutions. A sensor in its most simple form would only provide the information whether an object is within the agent's vicinity or environment. Adding additional information to a sensor, such as spatial position, saliency, or perception clarity of an object, increases the accuracy of information at the cost of additional resources. As discussed in [SvS5], the realization of a sensor should ideally allow an application designer to parametrize the sensor to balance accuracy against performance.

Further performance related considerations are the direction of processing and semantic information. In nature, environmental stimuli will reach a sensor triggering a corresponding reaction. In the context of a real-time virtual environment, it makes sense to reverse the processing direction to save computational resources as discussed above. Additionally, by enriching perceivable objects with semantics, agents can be provided with additional information that would otherwise be difficult or impossible to calculate by the agent itself [vOD11]. A typical example is associating objects with affordances or even concrete interaction procedures (e.g., [KT99]).

To demonstrate the described sensor concept, examples of visual sensors are presented in Section 7.3. In this section, it is also discussed how sensors can be combined to balance performance, controllability, and correctness in a neuro-biological sense.

5.4 Memory Hierarchy

For processing, humans organize sensed and perceived information hierarchically. Thus, Atkinson and Shiffrin [AS68] formulated a psychological memory model called *stage theory*. According to a more recent review of the authors [AS16], their stage theory model has remained one of the most influential since its inception in 1968, to this date. The agent memory module of the CA^{2}RVE architecture is based on Peters and O'Sullivan's memory model, who adapted the stage theory model to virtual humans [PO02]. During sensing, sensors generate stimuli from sensed objects, which contain semantic properties. All stimuli from all sensors attached to an agent are gathered within a *short-term sensory storage* (STSS). At the end of each sensing step, the STSS contains a specific subset of all environmental stimuli. This subset represents all information that an agent could theoretically perceive in a given simulation state. Note that the state does not only include the relationships between objects in the virtual environment, but also the agent's internal state, specifically its sensor capabilities and sensor orientations. It is also important to point out that while it is possible

for an agent to perceive every information stored in the STSS, it only becomes aware of the information that it focuses on. Stimuli in the STSS are only stored for a brief period of time.

Once an agent becomes aware of an object, the stimulus turns into a *percept* and will reside in *short-term memory* (STM). Percepts represent a status change from unattended to attended stimuli, but in addition, they only store copies of dynamic object properties contained in the source stimuli. This approach ensures that agents are only aware of information that it perceived. If a property changes, the agent can only become aware of the current information by sensing it again, i.e., there must be a corresponding stimulus within the STSS. To become aware of an object, an agent needs to focus its attention on the object (see Section 5.5) and the STSS must contain stimuli related to that object. Once in STM, high-level processes, like decision-making, can retrieve percepts. Therefore, the content of the STM describes an agent's model of its virtual surroundings and its current working knowledge. Percepts stored in STM are removed if they are not attended to repeatedly, like stimuli in the STSS. However, percepts are retained longer than stimuli.

General concepts about a virtual world or inferred correlations (see, e.g., [PO02]) are typically stored in *long-term memory* (LTM). Examples of such information could be traffic rules, social interaction protocols, or optimal navigation paths. The LTM structure can also hold information that needs to persist across multiple simulation sessions, e.g., when a conversational agent is supposed to remember a particular user. To retrieve long-term knowledge, it is usually decoded and loaded into STM instead of being retrieved from LTM directly. In multi-agent systems, long-term knowledge is often identical for all simulated agents. Therefore, it is often beneficial to provide it using a structure outside of the agents, e.g., using *global semantics* (cf. [vOD11]).

5.5 Attention

Humans receive an incredible amount of information via their available sensors (see, e.g., [AEO05, BI13]). Unfiltered, the quantity of information would be impossible to process. Attention is a filter mechanism that allows living beings to focus on the subset of information that is currently relevant. What is relevant and what can be filtered out is determined by bottom-up and top-down attention processes.

Bottom-up attention is a reactive process that subconsciously directs attention towards *salient* objects in an environment by causing reflexive eye movements that are almost impossible to suppress ([Fri11, KT18, PCR+11]). These objects stand out from its surroundings due to one or multiple contrasting features, e.g., color, movement, or shape [KT18]. Top-down attention allows humans to consciously influence what is perceived and what is filtered. This

type of attention is important if a current goal needs to be achieved, like finding a specific object (see, e.g., [Fri11, KT18]). Similarly, the ability to control the perception process using top-down attention is also an important aspect for simulating human-like perception in IVA [vO14]. Bottom-up stimuli are often stronger than top-down influence due to the relevance for survival. If a viscous predator is rushing towards you, it is certainly more important to perceive it than finding the best fruit on the shrub in front of you. Despite the general prevalence of bottom-up attention, it is important to suppress bottom-up stimuli to a certain degree, to maintain a level of focus in a current task (cf. [Fri11]). Both directions of attention serve specific purposes that should be included in a synthetic model of human attention to generate believable agent behavior (cf. [vOD11, KvVH05, Fri06, PCR+11]).

Furthermore, as is the case with human attention, a similar synthetic process can reduce the amount of important information at a given time to a manageable size – effectively reducing computational cost. The main purpose of synthetic perception is typically to emulate the abilities as well as disabilities of a real-world counterpart. Given a certain situation, an IVA should ideally perceive the same objects as a human in the same scenario. It is likely impossible to match the precision and performance of the human perception system in a real-time application, which means abstraction of the process is inevitable. The amount of abstraction from the human system generally depends on a variety of factors, e.g., application domain, purpose of the system, and available resources, among others.

According to these considerations, the proposed framework is based on the main ideas of Treisman's *Feature Integration Theory (FIT)* and Wolfe's *Guided Search Model*. A summary of these models can be found in [Fri11] and [Kim06]. A structural overview was presented in Figure 5.1, but Figure 5.3 demonstrates the process of perception within the framework with specific focus on attention. Early in the process, features are registered in parallel by the available sensors. Here, parallel refers to the occurrence of events regarding the sense cycles, meaning that all stimuli reaching an agent's sensors are available in the STSS after a single sense step. The importance of a stimulus S_i is described by a *saliency value* s_i, which is assigned by each sensor during the acquisition process. Saliency describes how conspicuous an object or feature is within a given surrounding (cf. [IDP03]), which can be treated as a kind of utility for selecting one stimulus over others. Each stimulus S_i is associated with exactly one virtual object o_j from a set of virtual objects $O = \{o_1, \ldots, o_m\}$, but multiple stimuli with independent saliency values can be sensed from a single object (see Figure 5.3). Due to the open design of the sensor interface, stimuli can be of different modalities (e.g., visual and auditory), but also of the same modality (e.g., visual ray-casting and visual false-coloring).

After sensing, the actual perception of a stimulus' source object can only be retrieved sequentially, i.e., an agent must direct its attention to each object of interest one after the

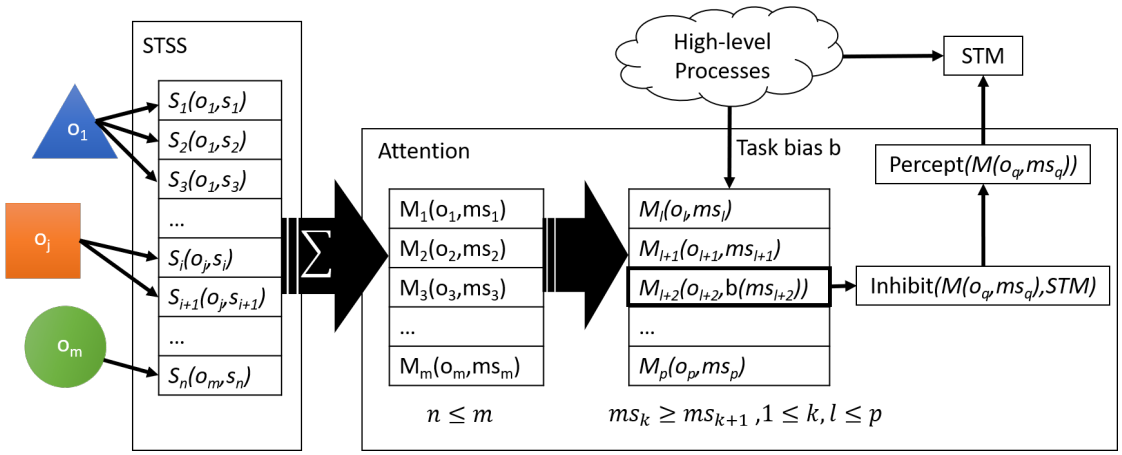


Figure 5.3: Depiction of the proposed perception process. Sensors create stimuli S_i for each object that is sensed, which are associated with a saliency value s_i and stored in STSS. Stimuli are then integrated across all sensor modalities to create one multi-sensory stimulus $M_k(o_k, ms_k)$ for each sensed object o_k with a combined saliency ms_k. The stimuli are sorted according to their associated multi-sensory saliency. Top-down attention is emulated using a task bias, which can increase the saliency of a multi-sensory stimulus. A percept is created from the stimulus with the highest biased saliency and moved to STM making the agent aware of the associated virtual object. An additional inhibition process decreases the saliencies for objects that have been attended to recently.

other (cf. [Enn04]). To simulate the focus effort, an attention process will select the most salient object from the STSS as the stimulus of interest and the agent aligns its main sensors towards the source of the stimulus. To realize the sequential selection, the stimuli in the STSS are first integrated across all sensors to find a multi-sensory saliency ms_k for each virtual object o_k, which is represented by at least one stimuli in the STSS, generating a multi-sensory stimulus M_k.

The multi-sensory saliency ms_k is a weighted sum of the individual sensor saliencies, similar to the approach presented by Balint and Allbeck [BA13], but here a linear combination is used:

$$ms_k = \sum_{t \in T} w_t \frac{\varsigma(o_k, t)}{\sum_{j=1}^{m} \varsigma(o_j, t)} \tag{5.1}$$

where T is the set of available sensor types (visual, auditory, olfactory, etc.) and $\varsigma(o_j, t)$ is the sum of stimulus saliencies for an object o_j and a sensor of type t. The weights for each sensor type (w_t) can be specified by either using educated guesses, deriving them from the current context, or extracting them from test data. Weights can also be adjusted based on a modality's reliability. For example, in a dark room, visual sensors do not function to their full capacity and may therefore become less significant than other sensors, like audio sensors.

The set of multi-sensory stimuli is then sorted according to their saliency, such that

$ms_k \geq ms_{k+1}$. The attention process selects the combined stimulus with the highest saliency value and makes the associated object the current object of interest. After focusing on the object, the according stimulus is moved to STM as a percept. Only after this percept is passed to STM, is the agent aware of the corresponding object, i.e., high-level processes can query information about the object.

With this approach, the most recent object remains in the center of attention, making it highly unlikely that other stimuli will capture the agent's attention. Therefore, the stimulus may be inhibited for the upcoming cycles, which means the associated multi-sensory saliency is decreased. The moment and level of inhibition as well as the number of objects that can be inhibited simultaneously are customizable within the framework at runtime. By altering these parameters, various search behaviors and degrees of attention can be modeled and simulated. For example, if only a few objects can be inhibited, the agent will repeatedly focus a set of similar objects appearing forgetful or nervous. Another aspect influencing the inhibition of an object are object properties. Some objects may be complex requiring more attention or certain properties may change continuously (e.g., position), which means they may remain salient and should not be inhibited after being perceived for the first time.

Implementing the definitions above represents a flexible and adaptable approach to modeling bottom-up attention. However, attention based solely on bottom-up mechanisms would not allow other processes to influence attention allocation (cf. [Kim06, PCR+11]). Therefore, to simulate plausible agent behavior, the ability to direct attention based on current knowledge or goals must also be integrated into the perception framework. To simulate this controlled behavior, bottom-up saliency determined by the virtual sensors is biased within the attention step based on the agent's current priorities or preferences. This process is based on the Dynamic Perceptual Attention (DPA) model developed by Kim [Kim06]. Here, the process is adapted to apply a bias function b to multi-sensory saliency values associated with certain objects as depicted in Figure 5.3. The bias function elevates the saliency[1], giving the stimulus a higher importance than that assigned to it by the bottom-up saliency. In Figure 5.3, the biased saliency is high enough for the stimulus to be selected over other stimuli that have higher priority. The amount of deviation from the original saliency can either be pre-determined at design-time or generated in an agent-centric way at runtime, e.g., by evaluating object ontologies (cf. [vOVD12]). This allows prioritizing stimuli related to an agent's current goals. At the same time, highly salient bottom-up stimuli will still be perceived if they are larger than the biased saliency value.

[1]Currently, the bias function only increases saliency, because it makes sense in the context of the application. However, the realization does not prevent a designer from providing negative biases to decrease saliency.

5.6 Personality and Emotion

In a multi-agent scenario, even plausible behavior becomes implausible if all agents behave in the same way. In Chapter 4 the benefits of including personality and emotions into an agent's cognitive processes were modeled. These arguments also apply to agent perception, which is a part of the cognitive apparatus and arguably the foundation for agent behavior. Therefore, personality and emotion should also influence perception and vice versa. *Task parameters* are used to control how personality affects agent behavior in specific tasks or task areas. One such parameter could be "focus" describing how rapidly memory entries decay or how many objects an agent can attend to. Using this approach, agent perception can be individualized and help create more diverse and dynamic behavior patterns. Furthermore, personality and emotion could modulate stimuli or saliency shifting attention and perception towards specific objects (see [PCR+11]). Perceived objects could also influence an agent's emotional state similarly to emotional incidents defined in Section 4.2.2.

5.7 Semantic Modeling

One benefit of virtual worlds and virtual objects is that information that cannot be sensed directly must not be learned or derived by an agent using complex high-level processes. Instead, such information – often referred to as *semantics* – can simply be "attached" to objects and retrieved whenever required. Semantics are commonly integrated into a virtual world at the object level, i.e., virtual objects are augmented with additional information like appearance, physical properties, roles, behavior, services they provide, or affordances (see, e.g., [BA13, TBSK08, vO14]). In the CA^{2}RVE architecture, semantic information provides the characteristics of a perceivable virtual object or determines how it is perceived. In the context of VE, dimensions, color, loudness, or saliency are some examples of object semantics that are used in the perception process (cf., [TBSK08]). World semantics can add information on a larger scale, e.g., weather conditions in an area of the virtual world. This type of semantics can be used by sensors during the sensing process to model effects like limited range of sight due to darkness or fog. For more details about semantics in the proposed perceptual framework see [IVC8].

5.8 Conclusion of Integrating Perception into IVA

In this chapter synthetic perception has been introduced and it was explained how it can be used to improve the behavioral realism of intelligent agents in real-time virtual environments. For this purpose, a generic model for synthetic virtual perception was proposed

to solve **RT3.1**. The focus of the model are a virtual sensor interface, a hierarchic memory module, and an attention model to limit an agent's knowledge about its environment in a believable, yet controllable way. Since sight is the most important sense for humans, visual perception was primarily addressed, although it was explained how multi-sensory information is integrated into the process. Regarding **RT3.2**, the proposed perception framework offers a flexible and customizable solution for modeling virtual perception. When designing sensors, the focus can be either on accuracy, scalability, or controllability. Virtual sensors can be straightforward for efficient processing or provide redundant information for increased fault tolerance. However, sensors can also be designed to be an elaborate emulation of biological sensors to realize realistic sensing. Although these focus areas may not be mutually exclusive, it is usually difficult to model sensors that excel in all of them. The strength of the presented model is the flexible, dynamic combination of multiple sensors to mitigate the drawbacks of individual sensors. As a result, the model facilitates solutions that are accurate, scalable, and controllable, allowing for a wide range of potential applications. Possible examples are interactive games, perception research, education, and simulation.

An integrated memory module stores both stimuli and percepts generated from stimuli that have been attended to. Although high-level processes, like decision-making, have not been addressed in this chapter, the memory module also acts as interface from and to these subsequent processes. The included attention model combines bottom-up reactive attention through object-based saliency values and top-down task-oriented processing by biasing sensed object saliency values. Memory and attention are configurable to fit application specific needs, achieving the desired controllability.

The configuration of each perception module can also be used to reflect an agent's personality or emotional state. This option has not be explored in depth, but an approach for including this influence was indicated. The concept of task parameters (see Chapter 4) can be readily applied to the discussed perception modules.

One limitation of the perception framework regarding plausibility is the object-based design. Stimuli and percepts are always linked to virtual objects, which means perceiving parts of an object is equivalent to perceiving the entire object. Similarly, while stimuli saliencies are compared, there is no inherent context to the comparison. Relationships, like proximity, must be explicitly included in stimuli acquisition. Due to this design, it is difficult to model certain perceptual effects such as figure-ground relationships, camouflaging, or grouping. Considering these principles within the confines of the devised framework is not impossible but would, in many cases, require a non-negligible effort. Additionally, the design does not prevent a user from defining a sensor that perceives every information within the virtual world, creating *all-knowing* agents. Such a sensor would defeat the intention of

emulating human perception, but it would be a conscious decision made by the designer. It is also the sensor designer's responsibility to balance believability with scalability and to consider controllability.

Real-time capability was considered during the design process of the perception framework as stated in **RT3.2**. The next chapter demonstrates that the overall cognitive agent architecture, including perception, is suitable for real-time application.

6

Application of the Agent Architecture Design

"Knowing is not enough; we must apply. Willing is not enough; we must do."

- Johann Wolfgang von Goethe

In the previous chapters, motivations and foundations for a real-time attentive cognitive agent architecture were discussed. Based on these considerations, a concept was put forth with the intent of providing a flexible architecture to facilitate the generation of plausible agent behavior based on personality, emotion, and perception. Throughout this thesis, it was emphasized, that besides plausibility and controllability of behavior, real-time constraints of the target domain must be considered. In the context of interactive experiences in virtual environments, a minimum framerate of 30 frames per second (FPS) is typically considered real-time. To achieve these framerates, calculating each new frame must be generated in 33.33 ms/frame or less. To answer **whether the proposed concepts scale well enough for real time, multi-agent applications (RQ4)**, the validity of the proposed concept is demonstrated by implementing it in a real-time traffic simulation scenario. The steps and consideration required for this realization are discussed throughout this chapter.

FIVIS[1] is a bicycle simulator developed at the Institute of Visual Computing of Hochschule Bonn-Rhein-Sieg [HSK+10, HSK+12] (see Figure 6.1). Used as a tool for road safety education of elementary school children, simulating challenging and instructional traffic situations has always been an integral part of the FIVIS system. The original scenarios developed for FIVIS lacked variety as agent behavior was trigger-based and purely scripted.

[1] Fahrradfahrsimulator in der Immersiven Visualisierungsumgebung Immersion Square (Engl.: Bicycle driving simulation in the immersive visualization environment Immersion Square), vc.h-brs.de/fivis

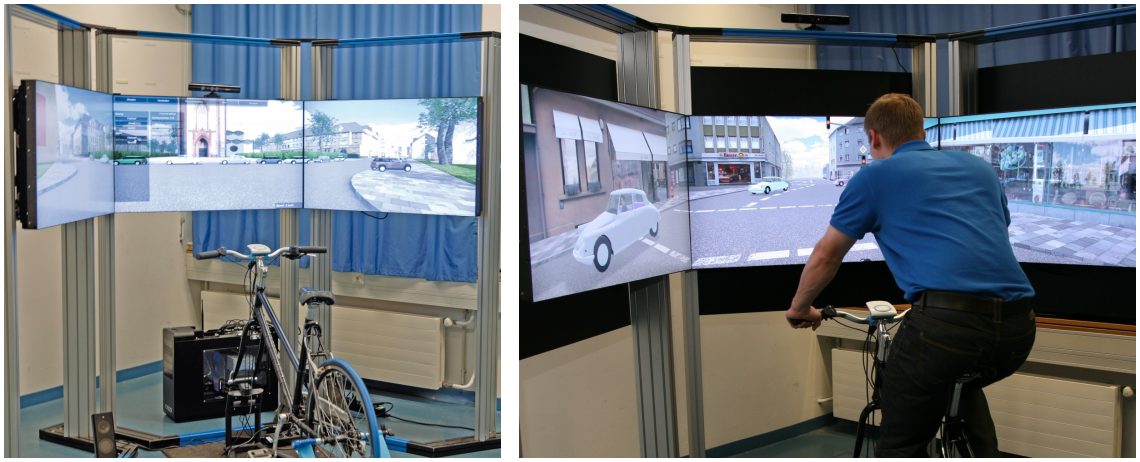


Figure 6.1: Three-display setup of the FIVIS bicycle simulator including bicycle input at the Institute of Visual Computing of Hochschule Bonn-Rhein-Sieg in Sankt Augustin.*
*3D assets by IVC.

The implemented scripting interface only allowed designing static behaviors in a tedious and time-consuming process. The AVeSi[2] project provided an opportunity to continuously apply the concepts developed within this thesis with the goal of simulating more life-like traffic within FIVIS scenarios.

A traffic simulation for road safety education is an interesting area of application for cognitive agents as it requires a mixture of game-like agents and traffic simulation. Traffic agents in games are typically used as a backdrop to the game experience, conveying the illusion of a living world. They usually do not serve any educational purpose and may even try to clear space for players to avoid hindering them.

Traffic simulations are used in a wide array of tasks. Consequently, the field of traffic simulation research is advanced, but applications typically focus on analytical investigations of traffic phenomena. While incorporating aspects of human behavior in traffic, these aspects are usually modeled as randomization of normal behavior or modeled at an abstract level. For example, in the original Nagel-Schreckenberg model, the velocity of cars is reduced with a certain probability to emulate dawdling (cf. [TK10]). This may sufficiently re-create human behavior to analyze certain traffic characteristics, but it is usually independent of the actual situation that would lead to driver-dependent adaptation of behavior. The latter is what leads to interesting and perhaps dangerous situations when observed up close. Human drivers make mistakes, violate or bend traffic rules, or take risky choices; the same should be true for traffic agents if trainees are to learn from observed and experienced behavior.

[2] Agentenbasierte Verkehrssimulation mit psychologischen Persönlichkeitsprofilen (Engl.: Agent-based traffic simulation with psychological personality profiles), vc.h-brs.de/avesi

Applying the presented cognitive agent architecture to FIVIS' traffic simulation ideally achieves this level of believable driver behavior. Due to their approaches and the general intended domains, cognitive architectures require substantial computational resources [SSSS16, BGG19]. Approaches in IVA/ECA research are often also limited to only a few individual agents, in most cases only one or two agents are considered (e.g., [CHLC18, Geb05, GBK+10, tSKT+20, ZMLY19]). Based on these observations, the developed cognitive agent architecture and especially CA^{2}RVE was designed as a modular, lightweight cognitive architecture. The intention of these design choices was to provide a scalable solution that is applicable to scenarios with larger agent populations. At the same time, the plausibility and controllability constraints should not be compromised. Finding solutions to this problem is subject of **RQ4**. In this context, a perception related solution is to be investigated first, before considering a more general answer.

RT4.1: Efficient acquisition of knowledge about an agent's environment.

It is not clear whether human-level cognitive capabilities can be realized on conventional computer hardware even without focus on efficiency [DOP08]. Therefore, realizing the developed concepts for interactive applications generally means finding ways of simplifying processes without compromising plausibility and controllability or reducing scale, e.g., by focusing on a single virtual agent.

Despite the variety of available solutions, a common approach is to simplify the decision-making and perception processes. When dealing with virtual worlds and objects, information about them can be associated at any time, preferably at design time. This information is often referred to as *semantics* and is a major advantage of synthetic perception in virtual worlds, since semantics can simply be looked up at runtime instead of having to intricately extract them from sensor information, which is typically the case in robotics. Semantics are commonly integrated into a virtual world at the object level, i.e., virtual objects are augmented with additional information like appearance, physical properties, roles, behavior, services they provide, or affordances (see, e.g., [BA13, TBSK08, vO14]). Extending virtual objects with sensible semantics is one approach taken in this work towards answering this research task. Examples should include measures, color, loudness, and saliency. Additionally, world semantics can add information beyond the object level, e.g., to model effects like limited range of sight due to weather conditions. Sample realizations should showcase the efficient acquisition of knowledge about an agent's virtual world utilizing the perception framework. Using the integrated sensor interface, it should be possible to realize sensors that reduce task complexity for agents while maintaining plausibility.

RT4.2: Realizing the cognitive agent architecture concept as real-time application.

To verify that the proposed design is applicable to interactive, virtual user experiences, the design should be realized as part of the FIVIS bicycle driving simulator or more specifically applied to its embedded traffic simulation. By means of this application, the initial claim of providing cognitive agents for real-time virtual environments should be verified. While parts of this task are addressed in Chapter 5, applying the proposed concepts to an interactive, virtual user experience should be the result of this task. Agents realized using the proposed architecture concept are to be used as traffic participants populating scenarios within the FIVIS bicycle simulator. The modular and extensible design should allow adjusting simulations to application-related constraints. Furthermore, a prototype implementation demonstrates the realization of the architecture design in a possible production environment. This includes handling additional constraints imposed by the utilized tool set, like a game engine. Using exemplified techniques, it should be possible to meet real-time and project-specific requirements.

The following sections describe how the presented concepts are implemented to realize several traffic scenarios regarding plausibility of generated behavior. First, the application of the cognitive agent concept to traffic simulation is described, followed by several application-specific extensions and a look at another possible application domain. At the beginning of the project, the most common 3D game engines were surveyed to find the best fit for the traffic simulation as well as FIVIS. At that time, the Unity game engine[3] was selected as it was the most promising. Unity offered an affordable license system and an easy to learn workflow. Especially the latter turned out to be useful, as students joining the project did not need previous experience using the engine. Unity has evolved substantially over the course of this thesis, and it seems to have become one of the most popular game engines, especially for small studios, independent developers, and in education. Choosing Unity at the beginning of the project also allowed tightly integrating a sample realization with the game engine. This integration, as opposed to middleware approaches (e.g., [GBK+10, vO14]), avoids performance issues caused by communication latency, unnecessary abstractions, and costly conversions. Furthermore, it simplifies efforts for newcomers and potential users of the system, as they do not have to familiarize themselves with multiple platforms.

[3] `https://unity.com/`, [online: May 2, 2023] Unity Technologies

6.1 Cognitive Traffic Agents

Based on the intended application, traffic agents realized using the CA²RVE architecture should possess the following properties:

- Psychological personality profile for individualized, consistent behavior according to **RQ2**.
- Generally following traffic rules, since human traffic participants generally follow these rules.
- Ability to handle specific traffic scenarios to facilitate the creation of learning scenarios for trainees.
- Separation of body and mind to allow exchangeability and extensibility.
- Plausible driving behavior to support immersion and the suspension of disbelief.
- Plausible perception of other traffic participants to create give agents a believable knowledge of their environment, which improves the plausibility of their behavior (cf. Section 5).

To achieve these requirements, the modular design depicted in Figure 6.2 was developed.

Incorporating users is achieved by representing them as *player agents*, allowing other agents to interact and communicate with users similarly to how they interact amongst each other. For *artificial agents*, behavior is strictly determined by their decision strategies and associated behavior models, but extensions can include a static personality profile (see Section 4.1), or a dynamic profile based on an emotional state (see Section 4.2).

Static and dynamic personality profiles were realized in accordance with the concepts presented in Chapter 4. Five dimensions are used to represent the FFM traits (openness, conscientiousness, extraversion, agreeableness, neuroticism). *ArtificialAgents* do not possess a personality profile, their decisions are based on default behavior models. A derived *PersonalityAgent* class is extended with a personality module, which is composed of a FFM profile. To include an emotional state and exert influence on the static profile, an *EmotionPersonalityModule* is derived, which is a component of an *EmotionalAgent*. The perception of emotional incidents is realized using an *ImpressionModule*. Coefficients for relating the perception of emotions to personality, emotional state to dynamic profiles, or personality to task parameters are contained in a global configuration module (not depicted in the diagram).

Most elements of an agent are realized as components to maximize versatility. Every agent includes a mind component, which consists of cognitive modules for navigation, decision-making, and traffic memory[4]. Traffic rules are followed by the implemented nav-

[4]The module is deliberately called traffic memory to distinguish it from perceptual memory. It contains information about the current driving state, e.g., current lead agent or obstacle or the previous acceleration.

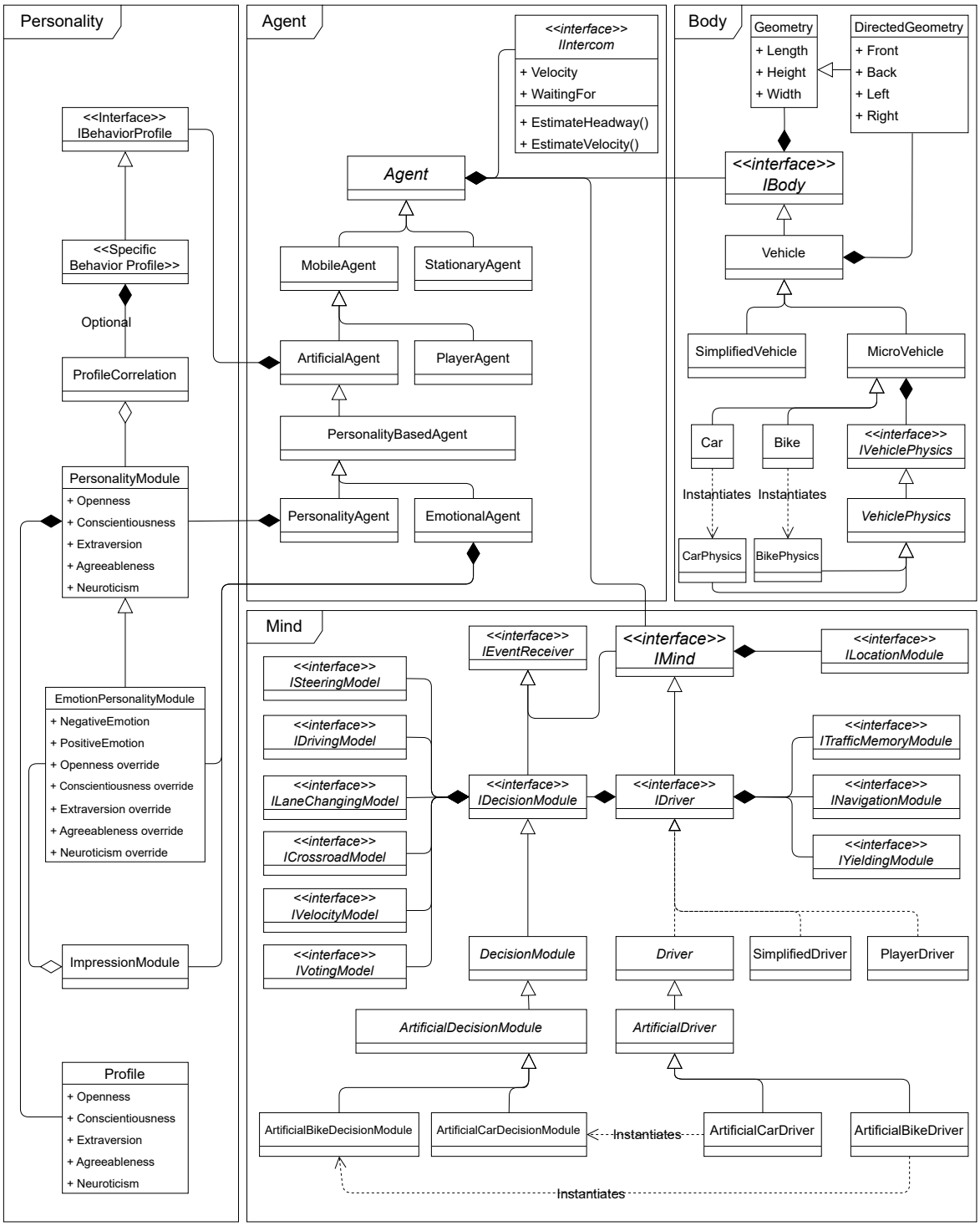


Figure 6.2: Simplified view of the realized CA^{2}RVE architecture for cognitive traffic agents. The separation of thought processes (mind) from an agent's physical representation (body) in combination with the modular design allows flexible usage of agents. Users are considered as *player agents*.

igation module and *driving strategies*, e.g., freeflow, following, or lane changing. Strategies are implemented using behavior trees, which are explained at the end of this section. The number of strategies can be extended if necessary, and they are configured using appropriate models for steering, acceleration, lane changing, etc. Each module interface must be realized for the according agent and mind type. The diagram in Figure 6.2 shows an example of this process for the *IDecisionModule* interface. The decision-making module also uses models to configure actions. In turn, each model requires a behavior profile that is attached to an agent (*IBehaviorProfile*). For example, an acceleration model may calculate an agent's desired velocity while following another agent based on an agent's minimum headway to its lead, an acceleration coefficient, or comfortable deceleration. All these behavior profile parameters may be associated with an agent's personality profile.

Politeness ϕ is one behavior profile parameter that is used as configuration parameter for the lane change model. The parameter's value can be globally set for all lane change decisions, or it can depend on the agent's personality profile. The realization of lane changing is based on MOBIL ("Minimizing Overall Braking Induced by Lane Changes") by Kesting et al. [KTH07], which includes a politeness parameter and integrates well with the realized car following model (see below). Using this model, an agent a_c decides to change to a neighboring lane with the same driving direction if the following holds:

$$\ddot{x}_c - \ddot{x}'_c + \phi \cdot \left(\ddot{x}_g - \ddot{x}'_g + \ddot{x}_f - \ddot{x}'_f\right) > \Delta\ddot{x}_{th}, 0 \leq \ddot{x}_{th}, 0 \leq \phi \leq 1, \tag{6.1}$$

or if the neighboring lane's driving direction is the opposite of the agent's driving direction, the model is adapted as follows:

$$(1-\phi) \cdot (\ddot{x}_c - \ddot{x}'_c) + \phi \cdot (\ddot{x}_o - \ddot{x}'_o) > \Delta\ddot{x}_{th}, 0 \leq \ddot{x}_{th}, 0 \leq \phi \leq 1, \tag{6.2}$$

where $\ddot{x}_i$ denotes the current acceleration of agent a_i and $\ddot{x}'_i$ the predicted acceleration of agent a_i after the considered lane change was performed. Figure 6.3 provides an overview of agents c, f, g, l, m, and o[5] involved in the lane changing decision and the according parameters. Agent a_c is the agent currently considering a lane change, a_l is the agent that a_c is currently following, and a_f is the agent following a_c. Agent a_g would become a_c's follower after the lane change and a_m would become a_c's lead. In a scenario with opposite driving directions, a_o is the agent driving in the opposite direction to a_c. The parameter $\Delta\ddot{x}_{th}$ is a threshold, which is a part of the MOBIL model preventing lane changes that result only in marginal advantages. The politeness factor ϕ determines how much the acceleration

[5]The indices indicate the role of each agent in the current decision: c – currently considered agent, f – following agent, l – lead agent for agent c, g and m are the indices following f and l, respectively, since they would be follower and lead in the event of a lane change. Agent o is the one driving in opposite direction to agent c.

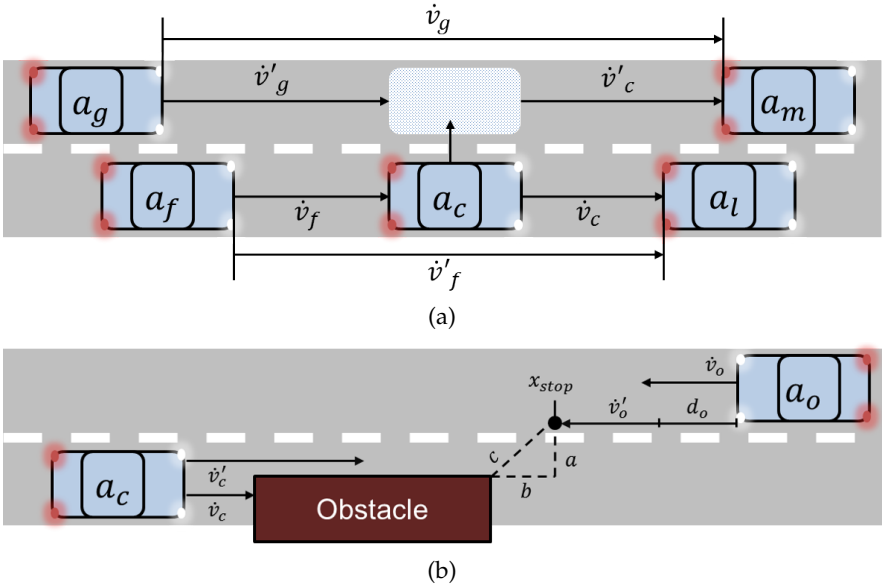


Figure 6.3: Overview of parameters used in lane change decision for agent a_c. The MOBIL model by Kesting et al. is used for equal driving directions (a). For opposing driving directions, the MOBIL model was adapted (b). Arrows indicate accelerations before the lane change ($\ddot{x}$) and after a lane change ($\ddot{x}'$) for all agents relevant to the decision. For the opposing lane scenario, additional parameters are required to calculate the latest possible stopping point for a_o. Images from [SvS14].

change of other agents is considered by a_c. For maximum politeness ($\phi_a = 1$), the agent considers the acceleration change of a_g and a_f by 100% in equal driving direction scenarios and completely disregards its own advantage when waiting behind an obstacle for opposite driving direction lanes. The latter fully acknowledges that a_c does not have priority in this situation. For the minimum politeness, a_c only cares about its own advantage, but it does consider safety criteria (e.g., minimum gap distances to prevent accidents). The politeness factor can be derived from an agent's personality profile using a tuple of coefficients $\mathbf{c}_\phi$ (cf. Section 4.1.2):

$$\phi_a = \frac{(\mathbf{c}_\phi \bullet \mathbf{b}_a) + 1}{2} \tag{6.3}$$

Using this equation, a medium politeness ($\phi_a = 0.5$) is calculated if personality does not have any influence on the parameter ($\mathbf{c} = \langle 0,0,0,0,0 \rangle$) or if an agent has an average profile (i.e., the z-scores for all traits are zero). Since profile traits and coefficients are within the [-1;1] range, the result of Eq. 6.3 must additionally be constrained to the [0;1] range.

An agent's embodiment in traffic is determined by an implementation of the body module. Due to this component-based realization, several classes and subclasses of traffic participants can be set up for simulation (e.g., car, bicycle, bus, etc.). By defining a separate agent

body, including the according properties, agents can even change their means of travel at runtime. Since an agent is aware of its body component, it would be possible to associate different behaviors and task parameters with specific components. For example, an agent could act more cautiously while driving a bus, because it is responsible for all its passengers. For the prototype, four different agent types were realized: car, bus, bicycle, and pedestrian. Note that in Figure 6.2, sample implementations are only depicted for *Car* and *Bike*.

Every vehicle is represented in two simulation layers to improve scalability (see Section 6.3). In a simplified representation, bodies are mainly differentiated by their visual representation, i.e., the 3D model of an agent. For increased realism, physical properties and processes are integrated into a microscopic representation and simulated using Unity's physics engine. Examples of these physical aspects are collision geometry, weights, and torque curves for acceleration and deceleration.

According to Chapter 5, the perception module consists of virtual sensors and a memory hierarchy. To take advantage of the loose coupling of Unity's component mechanism, the perceptual components were attached to agent objects. This approach is orthogonal to the class structure introduced above for maximum decoupling. Virtual sensors are attached as child objects to a perception object. When the simulation is started, the sensors are attached to the perceptual systems and stimuli are sensed periodically based on a specified time interval. All sensed stimuli are kept in STSS until requested or until they are forgotten. A MemoryHierarchyModule, which contains the STM, handles STM decay, maintains an *interest* registry, and relays synchronous information requests from other processes to the perceptual module. If the STSS contains a stimulus that matches a requested predicate (e.g., a certain type of object), a percept is generated from the stimulus and the percept is stored in STM. Additionally, an attention process as described in Sections 5.5 and 7.3.2.1 was implemented to control which stimuli are moved to STM. For this traffic simulation prototype, *observer* components are attached to agents, which request task-based objects regularly and then feed them to the traffic memory or navigation modules. Figure 6.4 demonstrates this approach and its relationship to the agent class structure.

To realize the action selection process for traffic agents, behavior trees, which are a common tool for game AI [LB19], were chosen as they present a balanced compromise between development effort and complexity of generated behavior. Behavior trees are straightforward to implement, maintain, and extend, while at the same time being sufficient at generating complex behavior. The overall hierarchic structure, typically a directed acyclic graph, reveals behaviors in an intuitive way. At the same time, they are highly customizable as logic can be added at any node within a tree. Thus, using simple control nodes and modular actions, complex behaviors can be achieved with manageable effort (cf. [CS09, Cha07, Isl05]).

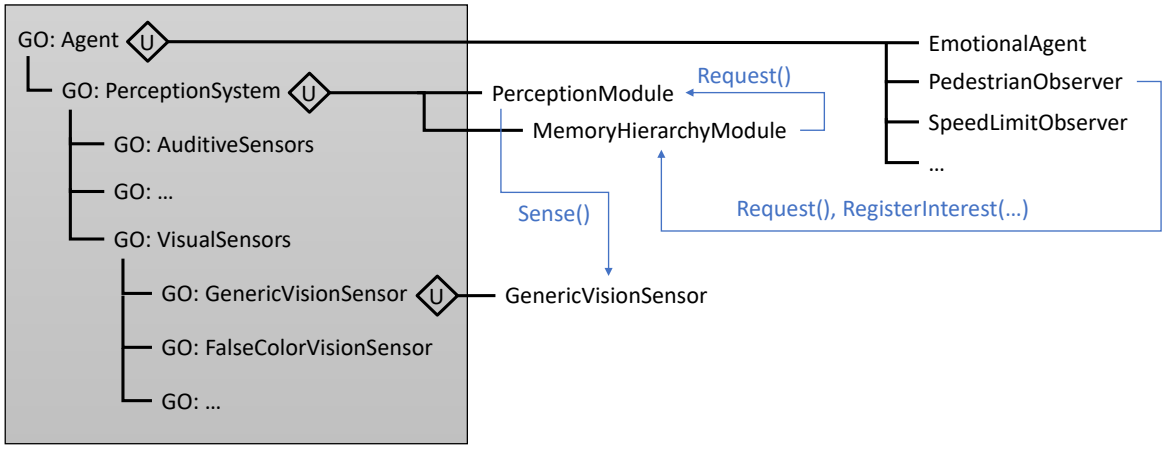


Figure 6.4: Component layout and game object hierarchy of traffic agent. Unity game objects are indicated using a "GO" prefix and Unity components are indicated by an aggregation symbol with an embedded "U". Components are gathered along the game object hierarchy at the beginning of the simulation. Interactions between perception modules are depicted using blue arrows.

Note that the latter does not exclude the fact that a behavior tree itself can become very complex if many behaviors are included. For the presented traffic simulation, the decision module includes a behavior tree, which is used to select between the aforementioned driving strategies. Just as the body of an agent can be exchanged to a specific embodiment, the behavior tree can be exchanged as well to express the according traffic behavior. Figure 6.5 shows an example of a behavior tree used in the realized traffic simulation. Examples of strategies present in the realized vehicle tree can be single actions, like determining the current desired velocity when following a lead vehicle using the Intelligent Driver Model (IDM) [THH00]. However, strategies could also be combinations of selections, sequences, conditions, and actions. For example, the lane change implemented in the prototype is a sequence of a condition, an evaluation action, and a selector based on lane direction. If the lane to change to is of the opposite driving direction, the lane change strategy becomes a sequence of checking the direction of the lane, then waiting to initialize the lane change based on safety and politeness criteria, then changing over to the opposite lane, then possibly staying on the other lane if multiple obstacles must be passed, and finally, changing back to the original lane. Other strategies may consist of only one action but might still be more complex than entire sequences. For example, the yielding strategy realized in the simulation is an action that requires agent negotiation based on a voting mechanism and may induce emotional incidents, which can affect strategy selection.

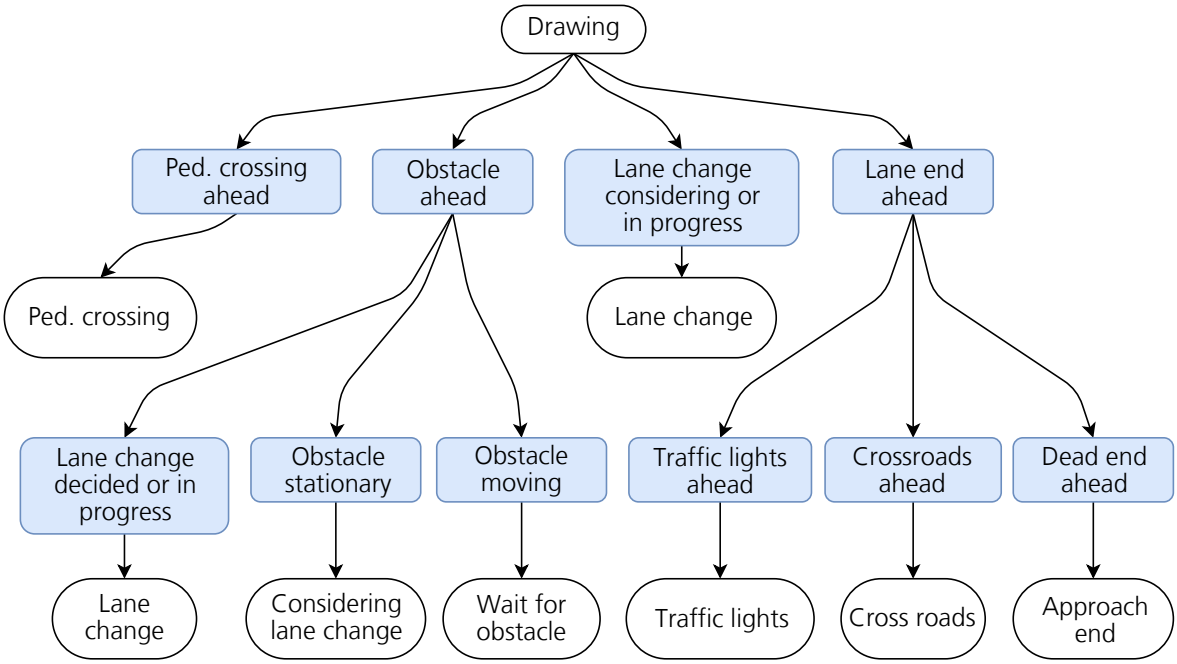
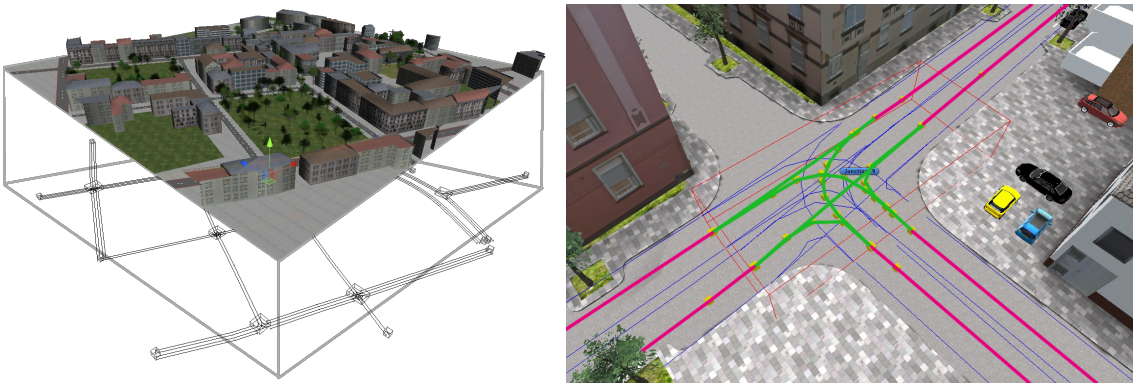


Figure 6.5: A behavior tree used for decision-making in traffic agents. Colored nodes are conditions and white nodes are behaviors or strategies. Note that behaviors can be re-used. Each leaf node represents another sub-tree that is omitted for brevity.

6.2 Semantics as Extension of Long-term Memory

In the current version of the presented CA^{2}RVE architecture, long-term memory is not explicitly considered. However, traffic agents need to understand general driving behavior and concepts that humans learn in driving school, e.g., the concept of a lane or right-of-way priorities, knowing what the allowed speed limit is, how to drive when following another traffic participant, or how to perform a lane change. Realistically, this knowledge is stored in every driver's long-term memory and accessed while participating in traffic. In the context of a simulation, this knowledge can also be considered global knowledge that applies to all agents within the virtual world. Both approaches are used in the presented traffic simulation. Driving strategies represent behaviors or knowledge common to all agents, e.g., vehicle following or lane changing. However, agents can apply the general behavior to their own situation, personality, and emotional state. These agent-centric deviations can include maintained distance to a leading vehicle, when to initiate a lane change, or how to handle speed limits.

While it is useful to modulate driving strategies, the same does not necessarily apply to other global knowledge. What actual speed limit applies to a given road, how wide a lane is, which state a traffic light is in, or on which side to drive on are examples of facts that



(a) (b)

Figure 6.6: (a): A semantic road network added to a virtual world as additional layer. Information from that layer can be accessed via object semantics using an agent's perceptual system. Image from [SvS8].* © 2014 IEEE (b): A closeup of the road network representation showing lanes (red) and connectors (green) on three roads and a connecting intersection. Boundaries of lanes and connectors are indicated in blue.**

*Terrain assets generated using Trian3DBuilder [Tri].

**Vehicle assets based on designs by Dosch Design (pink car) [Dos], "Underground Lab" (blue car) [Und], and "kilastaras" (black car) [kila]. Terrain assets and yellow car by IVC.

describe the current state of the world and the rules of traffic. Agents may interpret and use this knowledge differently, but that does not change it for other agents. So instead of storing this information in LTM and having agents decide what to make of it, it is represented globally as an additional semantic layer in the virtual world (see Figure 6.6). This layer is not visible to a user of the system, but agents are able to perceive elements of this layer using an application-specific sensor (cf. Section 7.3.2.1). Although the information in this layer is global, the semantics are object-based as each object of that layer contains the information. Providing such an approach avoids holding this information redundantly and provides it to each agent efficiently. Regarding the latter, agents do not have to employ complex processes that deduce lane topologies, right-of-way priorities, or what path to take across an intersection. Agents can *sense* this information and integrate it into their model of the world to utilize it for navigation and decision making. The provided information is exact and correct, but agents are free to ignore or modulate it internally. For example, while the exact course of lanes is encoded in the network, agents are not restricted to it; they can *choose* to drive elsewhere.

The semantic road network layer is based on the OpenDRIVE® standard [DSG10] and the road network representation used in VISSIM [FV10]. The network $\mathfrak{G}$ is defined by a set of lanes L, a set of connectors C, a set of roads R, a set of paths P, a set of information

elements I, a set of nodes N, and a set of junctions J resulting in a graph-like structure given by

$$\mathfrak{G} = (L, C, R, P, I, N, J). \tag{6.4}$$

The fundamental elements are nodes $n = (\vec{\boldsymbol{p}}, \mathscr{I})$, whose most important intrinsic information are their positions in 3D space $\vec{\boldsymbol{p}}$. Additionally, each node is associated with a set of information elements $\mathscr{I} \subseteq I$ made available to an agent that perceives the node, e.g., speed limits, associated traffic signs or traffic lights, lane width, etc. A sequence of nodes defines a segment S, which connects the semantic representation to the 3D geometry of the virtual world. The order of nodes in S defines the direction and course of the segment represented by a polygonal chain. Formally, each segment is a partitioning set of N, i.e., for a set of all segments $\mathscr{S}$ the following holds:

$$\begin{aligned} N &= \cup_{S \in \mathscr{S}} S \\ \emptyset &= S_i \cap S_j, \forall S_i, S_j \in \mathscr{S}, i \neq j \end{aligned} \tag{6.5}$$

Segments represent either lanes $l \in L$ or connectors $c = (S, l_p, l_s)$. Connectors have the special property that they have exactly one preceding lane l_p and exactly one succeeding lane l_s. Consequently, although lanes can have multiple incoming and outgoing connectors, the connector property provides a one-to-one correspondence between connectors and lanes. Lanes and connectors are further grouped to roads and paths, respectively. The main intent is to provide an additional structure on which relationships like neighborhoods or other properties can be defined. Roads are connected by junctions $j = (\mathscr{C}, \mathscr{F})$, which are associated with a set of connectors $\mathscr{C}$ that describe all possible paths across the junction. Features $\mathscr{F}$ associated with a junction are, e.g., right-of-way priorities for its connectors and therefore its connected lanes. An example of several road network elements is depicted in Figure 6.7.

Since the semantic road network representation is utilized for simulating traffic, it needs to be aligned with the virtual world to be meaningful. This alignment can be achieved by a manual workflow, e.g., using an appropriate editing tool. However, manually creating the semantic network becomes a tedious and error-prone process when larger road networks are realized. To increase efficiency and accuracy, the semantic information can be generated in combination with the 3D geometry of the road network automatically. To realize the traffic simulation prototype, both approaches have been implemented with the latter transforming OpenDRIVE® data to the format described above. More details about the model and the automatic generation process can be found in [SvS7, SvS6, SvS8].

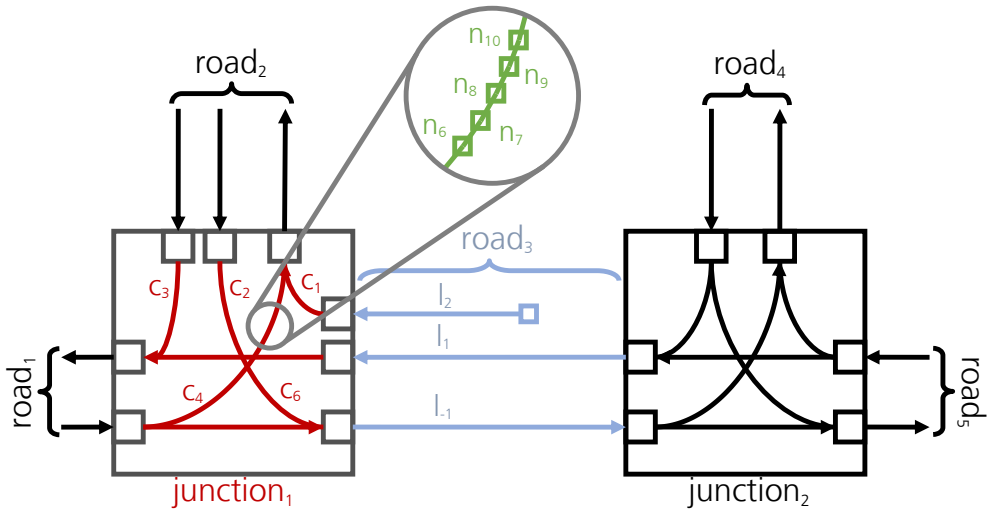
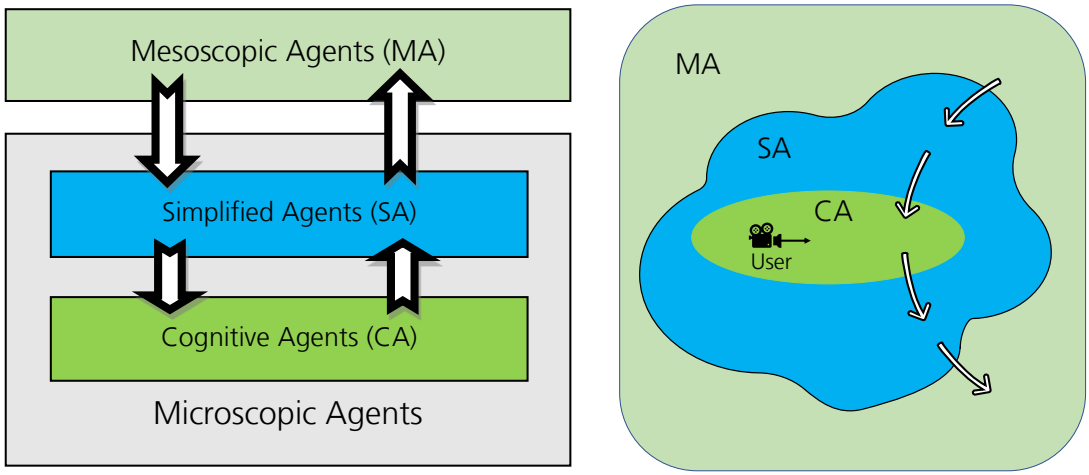


Figure 6.7: A schematic view of road network representation. Lanes are combined to roads, where the sign of the lane describes the driving direction with respect to the direction of description. $Road_3$ consists of one right lane (l_{-1}) connecting both junctions and two left lanes (l_1 and l_2) with the turning lane l_2 beginning in the middle of the road. Connectors c_i are assigned to junctions, describing all possible paths across a junction (exemplified for the junction on the left). Direction and geometry of each segment (lane or connector) is determined by a sequence of nodes n. Image based on [SvS9].

6.3 Scaling Simulations by Level of Detail

Traffic simulations in virtual environments often only simulate the visible surroundings of the user with a microscopic traffic simulation. However, it is undeniable that the level of detail required for microscopic, behavior-driven simulation requires significant computational resources even if efficiency is considered throughout the design and development phase. Especially in multi-agent simulations, a system will cease to be interactive once a certain number of entities is simulated. The more complex the simulation, the faster this threshold is reached. A standard approach for dealing with this issue, is to remove agents from the simulation once they exit a user's visual field, i.e., they are no longer visible on screen. One requirement for the FIVIS traffic simulation was to continuously simulate all traffic participants in an urban area to maintain realistic traffic densities and avoid inconsistent situations. Especially, agent-specific characteristics and states should be maintained throughout the simulation. Similar requirements have been adapted recently by open world video games, emphasizing their role in simulating a believable virtual world. For example, the game Assassin's Creed Origins[6] included a system that avoids deleting specific agents from a large open world [Lef18].

[6] Ubisoft, 2017



(a) (b)

Figure 6.8: Depiction of the level of detail approach to traffic simulation realizing a hybrid simulation. A microscopic simulation of cognitive agents (CA) and simplified agents (SA) and a simulation of mesoscopic agents (MA). In (a) the boundaries and transitions between the three layers are illustrated. Transitions are triggered based on distance to a user and visibility criteria, visualized in (b). Images based on [SvS9].

For the implemented virtual world used in FIVIS – a recreation of an area of the city of Siegburg – it was estimated that about 500 traffic participants must be simulated to represent rush hour conditions and 200 agents for regular traffic. Using agents described in Section 6.1 to populate the road network, an interactive simulation is achievable until about 40 agents are simulated. Simply removing agents from the simulation when the threshold is reached would contradict the requirements stated above. Therefore, to fulfill the requirement and preserve an interactive application, the simulation is divided into three layers as shown in Figure 6.8. The result is a hybrid traffic simulation, which combines microscopic and mesoscopic simulation (cf. [BKA05b]) with the microscopically simulated area centered around and moving with the user.

6.3.1 Level of Detail Simulation of Traffic Agents

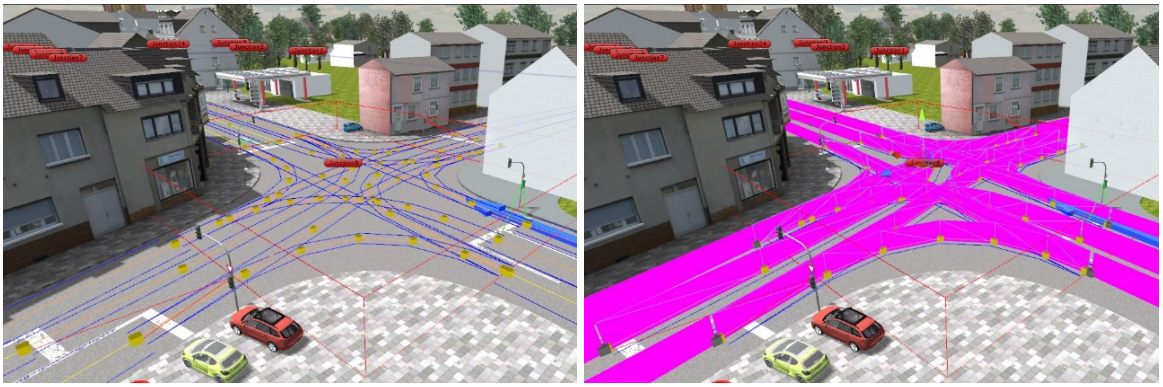
Cognitive agents (CA) take full advantage of the CA^{2}RVE architecture and the techniques and properties included therein. Additionally, to provide a realistic visual representation, vehicle physics are simulated, e.g., to achieve the typical pitching motion of a vehicle when braking. This layer is the most resource intensive to simulate, but since behavior of other drivers can only be observed in detail for a few agents at a time and in close proximity to a user, only a limited number of cognitive agents is required. While this number depends on

the application, the layout of the road network, and the structure of the world, it should be below the mentioned system limit for most situations.

Once agents leave the vicinity of a user, they will still be visible and thus cannot be removed or hidden. However, since their behavior is not observed as closely, they need not be simulated in such detail as agents in the CA layer. Specifically, a simplified vehicle representation without realistic physics can be used, driving dynamics can be simplified to following the nodes of the road network as described by the semantic road network layer, and perception can use simplified sensors designed toward efficiency, or the perception process can be simplified. The purpose of these *simplified agents* (SA) is to reduce computationally intensive components, while keeping a rudimentary visual representation that should be indistinguishable from CA when observed from a distance.

Finally, after agents become occluded by world geometry, it does not make sense to keep a visual representation of an agent. Furthermore, without a body many additional functions become obsolete as well, e.g., lane changing, car following, perception, etc. However, due to the required persistence, agents cannot be removed from the simulation. Using a macroscopic traffic simulation for occluded agents did not suffice as all individual characteristics of an agent are lost when transitioning to a macroscopic simulation (cf. [SvS2, IVC4]). Instead, agents are tracked throughout the road network using a mesoscopic traffic simulation. A mesoscopic approach was chosen as it preserves enough microscopic detail to maintain individual information (e.g., an agent's personality) as well as evolving parameters (i.e., emotional state), but reduce enough detail to provide an efficient low-overhead simulation. *Mesoscopic agents* (MA) are separated from the CA^{2}RVE architecture as their simulation is based on a different approach. While both CA and SA are used for microscopic simulations, where they actively decide what to do and where to go, MA are moved passively through the road network like packets by corresponding components.

The most critical aspect in combining the microscopic and the mesoscopic simulation layers is determining when transitions should occur between the two. Straightforward heuristics can be used to determine transition points that are connected to user proximity. For example, all roads ahead of a user's current travel direction could be simulated microscopically up to a certain distance along the network. It could be argued that if the distance is large enough, users would not be able to notice appearing or disappearing agents. However, such a heuristic would not explicitly include visibility between a user's location and road network elements. As a result, road sections that are beyond the specified distance away, could still be visible, e.g., through gaps in buildings or across open terrain. Additionally, user may not even be constrained to the road network itself but could navigate to any area of the world. Visibility information must be available in these areas as well.



(a) (b)

Figure 6.9: View of a junction area in a virtual environment. In (a)* a visualization of the road network elements used by agents for navigation and decision making is shown. To determine visibility of these elements, proxy geometries are generated that can be used for hit testing using raycasting in (b)*. Images from [SvS9].

*Vehicle assets based on designs by Dosch Design (pink car) [Dos] and "3DJunior" (green car) [3DJ]. Terrain assets by IVC.

To determine all visible network elements for a given position in the world would be a very resource demanding task at run-time, especially since it would require constant visibility checks to all directions. Spending substantial resources to perform these checks would defeat the purpose of reducing simulation complexity. Therefore, a *visibility set* V_A is determined for a specified area A during an offline process, which includes all network elements that are visible from area A. Given a function $r : \mathbb{R}^3 \times \mathbb{R} \times \mathbb{R} \rightarrow 2^E$, which describes a subset of all network elements E that are visible from a position x viewed in the direction of the polar angle θ and the azimuthal angle φ, the visibility set is defined as:

$$V_A = \bigcup_{x \in A, 0 \leq \theta < \pi, 0 \leq \varphi < 2\pi} r(x, \theta, \varphi) \tag{6.6}$$

For the traffic simulation prototype, r is realized using a raycast approach that tests intersections between rays and bounding boxes of scene geometry. Since road network elements are not physically represented in the virtual world, temporal proxy geometries are used for determining visibility (see Figure 6.9). The continuous parameters x, θ, and φ are stochastically sampled, and the road network geometry can be used to further restrict parameter ranges. For example, if the network is defined within a plane, x becomes two-dimensional and directions outside of the plane do not have to be sampled, i.e., θ is neglected. Areas are defined by dividing the virtual scene using a uniform grid of adjustable cell sizes.

Knowing the visibility sets for all grid cells, a user's current and adjacent cells are used to determine which agents are currently visible to a user and need to be simulated

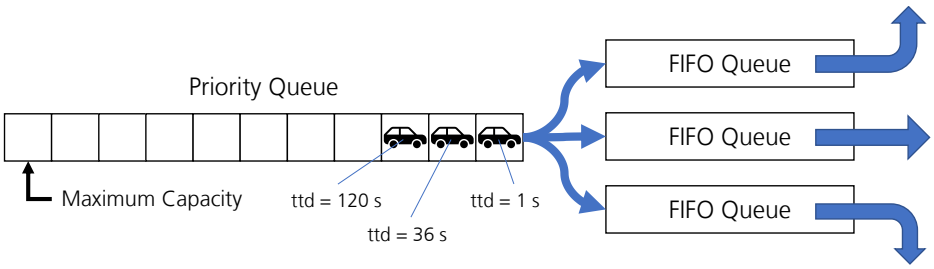


Figure 6.10: Schematic representation of a road for mesoscopic traffic simulation using a priority queue and following FIFO queues. The size of priority queues is limited by the maximum capacity of the road segment ϱ_{max}. The sort key for the priority queue is each agent's *time to destination (tdd)*. FIFO Queues exist for each connected road element and agents are queued based on their routing decision. Illustration based on [SvS3].

microscopically. From this information, it is deduced where agents must transfer between the microscopic and the mesoscopic layer, without the user noticing it.

Even with the addition of the MA and SA simulation layers, the number of simulated CA remains the limiting factor of the simulation. The current realization uses a fixed size ellipsoid around a user to simulate CA. In Chapter 7, it is shown that this approach may still lead to situations where the number of CA becomes too large to simulate in real-time. To mitigate this issue, instead of using a fixed size vicinity, the CA region could be defined to include a maximum number of CA, i.e., as soon as the number of CA passes this number, the size of the vicinity is reduced to match the specified maximum number. Other alternatives could be to keep the size fixed but use a scheduling algorithm to distribute the calculation of all CA across several frames, or similar agents could be grouped, and simulations could be performed on an aggregated representation. Both optimization options were not integrated into the described realization, but are interesting approaches for future improvements.

6.3.2 Mesoscopic Simulation of Traffic Agents

The mesoscopic simulation realized for this traffic simulation prototype is an adapted version of the FastLane model, which is based on queuing theory [Gaw98a, Gaw98b]. Roads are represented by directed edges E, with the direction indicating the driving direction. Each edge connects two vertices V, which are intersections. The resulting road network description is a directed graph $G = (V, E)$. The travel time a vehicle spends on a road is simulated using a combination of *priority queues* and *First In, First Out (FIFO) queues* (see Figure 6.10). Furthermore, each road is described by its length L, a maximum vehicle capacity ϱ_{max}, the current number of vehicles ϱ, and the number of lanes n. While the maximum number of vehicles that can travel on a road is calculated from a traffic density parameter in FastLane,

the value is derived from an average vehicle length l in the realization presented here:

$$\varrho_{max} = \frac{L}{l} \cdot n \tag{6.7}$$

The average vehicle length can be based on information found in the literature (e.g., [CBN03, Gaw98b]) or be derived from vehicle types deployed within the traffic simulation. Once the maximum number of vehicles is reached for a road, agents cannot transfer to it from connected roads. Instead, they must stay on their current road until a transfer is possible, resulting in spill-back effects. At the same time, the number of vehicles leaving a road is limited by a capacity parameter to simulate queuing effects. The capacity can additionally be randomized between simulation steps to emulate flow fluctuations of real traffic.

An agent's *time to destination ttd* (where "destination" is the end of the current road) is calculated once the agent enters a road based on the road's length and the agent's desired velocity v_0:

$$ttd = \frac{L}{v_0}. \tag{6.8}$$

In the original FastLane implementation, the current number of vehicles on a link is considered in addition to road length and desired velocity when determining *ttd*. However, Gawron also mentions that the number of vehicles can be neglected to sufficiently reproduce other traffic models [Gaw98b]. The *ttd* values are the priority parameter of the road's queue, being reduced until they reach zero. In the presented simulation, agents can have different desired velocities, e.g., based on personality, vehicle type, or simply randomization. For the presented context, this means the order of vehicles in the priority queue may change, which simulates vehicles passing each other, which is not unreasonable. In some cases, this may not be possible (e.g., overtaking may be prohibited or there is only one lane), but the effect is tolerated here. If overtaking needs to be restricted, desired velocities must be adjusted to agents present on the road or set to be equal for all vehicles entering a road.

As soon as the *ttd* has expired, agents are queued for leaving the road using FIFO queues. In the presented prototype, one FIFO queue is used for every connection to another road and capacities are shared according to road layout. For this reason, junction elements control the transfer processes between roads. Which FIFO queue an agent is added to can depend on various criteria. A probability distribution can be used for selection, but choices can also be based on route plans. Distributions can also be adjusted dynamically to provoke or prevent certain phenomena. For example, traffic jams are uninteresting situations for a road safety education simulator. Therefore, traffic can be distributed more evenly across the network to keep traffic flowing (cf. [SvS3]). When agents are transferred to the microscopic simulation

layer, their position in a FIFO queue and their current *ttd* are used to calculate their exact spatial position on a lane by iterating over all nodes of a lane.

The current number of vehicles in FIFO queues must be included in the maximum number of vehicles for the entire road. Turn pockets may increase the maximum number for short sections of a road, but this is effect is ignored for the sake of simplicity. If vehicles cannot exit a road, FIFO queues might fill up such that the sum of vehicles across all FIFO queues equals the maximum number of vehicles on that road, i.e., none of the vehicles on the road are currently moving. Further details on the mesoscopic traffic model can be found in [SvS2, IVC4, SvS3], and [SvS9].

6.3.3 Road Network Representation for Mesoscopic Simulation

To facilitate MA simulation, an additional road network layer is required. This layer can be generated automatically from the existing semantic road network (see Section 6.2) as all necessary information is available, such as lane-connector relationships, junctions, and segment lengths. An additional advantage gained from generating one network from the other is that mismatches during the transfer process between the two layers are avoided. Details about the generation process can be found in [SvS7, IVC7, SvS6, SvS8].

The geometric representation of lanes is encoded implicitly by the length L of a road. Since the employed queues support only one driving direction, roads with lanes in opposite driving directions are represented using two directed edges.

To simulate traffic, control is transferred from agents to the network elements (edges and vertices). Vertices representing junctions store distribution probabilities used for routing decisions. Consequently, agents behave like packets that are being passively transported across the network by the controllers. Since the tasks performed by the network elements have little complexity and agents do not perform any calculations, large networks containing large numbers of agents can be simulated efficiently, which will be shown in Section 7.4.

6.4 Conclusion

This chapter discussed a realization of the cognitive agent architecture concept and its integration into a traffic simulation. The motivation was provided by the FIVIS bicycle simulator developed at the Institute of Visual Computing of Hochschule Bonn-Rhein-Sieg. The simulator provides a tool for road safety education, which should include other traffic participants that emulate human traffic behavior as closely as possible. To this end, the architecture described in Chapter 3, including the concepts for static and dynamic personalities (Chapter 4) as well as perception (Chapter 5), was implemented using the Unity game engine. The

main focus was on real-time capability and a loose coupling of modules. Driving decisions are implemented using a behavior tree containing general and situation specific driving strategies. The influence of politeness on lane changing is described as one example for connecting driving decisions to an agent's personality and emotional state. Additional world knowledge is encoded using a semantic road network layer that can be perceived by agents using an application-specific sensor. The idea is to augment information gathering without accurately modeling the according combination of perception and reasoning. Designing such "super-human" sensors help maintain performance by reducing the complexity of a task for an agent. If the sensor is defined carefully, the plausibility of the perception process is not compromised.

To further consider real-time and consistency requirements in multi-agent simulations, a three-layered model is proposed that realizes a level of detail approach. Fully detailed cognitive agents are simulated only in the vicinity of a user. Agents outside the vicinity are reduced to a visual representation and simplified traffic behavior. An offline process is used to determine the visibility of road structures for every discretized position in the virtual world. This visibility information is used to transfer simplified agents to a mesoscopic simulation layer. In this layer, agents are no longer visualized but their individual information is preserved and restored once they transition back to the visible layer. A queuing model based on Gawron's FastLane model is used to perform the mesoscopic simulation.

By providing the described implementation using a game engine, it is argued that the proposed architecture concept is suitable for use in real-time applications, indicating an answer to **RQ4**. By developing a modular and extensible agent design, simulations can be adjusted to application-related constraints, especially real-time capability. While multiple cognitive traffic agents can be simulated simultaneously, it was impossible to simulate the required number of 200 agents and achieve more than 30 frames per second. It could therefore be argued that **RQ4** could not be answered successfully, as the agent simulation does not scale to the required population size. To increase performance, several steps could have been taken towards improving integration of the agent architecture into the game engine. Optimizations could have been more GPU utilization, simulating groups of agents together, or scheduling and balancing update cycles for each agent (see, e.g., [Cou15]). Other systems that are intricately connected to the game engine but not the agent architecture, e.g., rendering, physics, and animation, could have been optimized as well. However, within the scope of this thesis project addressing these optimizations was infeasible. Instead, the modular design of the architecture allowed meeting target framerates by successively disabling certain systems and modules, including resource intensive modules of the agent architecture, such as action selection and communication. Additional requirements, like

not discarding agents after they leave the user's field of view, were met by introducing additional systems, e.g., the presented mesoscopic simulation. Using these techniques, it is possible to meet the requirements. In addition to the answer to **RQ4** given in this chapter, the next chapter provides quantitative measures supporting the claim.

7

Evaluation and Results

"Without proper self-evaluation, failure is inevitable."

- John Wooden

The objective of this thesis project was to design an architecture for attentive cognitive agents that answers the research questions stated in Chapter 1. The main concern was to provide consistent but individualized behavior with additional focus on perception and attention, which can be applied to real-time virtual environments. In this chapter, the strategies used for evaluating the proposed solutions are presented. A thorough evaluation of an IVA architecture is a difficult endeavor. Primarily, evaluating believability of simulated agent behavior is always subjective and situational, which introduces bias. Evaluations are therefore based on surveys and questionnaires that also depend on user expectations and experience, the evaluated application, and what the agent is trying to imitate (cf. [Liv06]). Besides the issue of subjectivity, judging how human-like an agent behaves is another difficult challenge [ALS09]. One possible solution to this problem is a version of the Turing test. In the test's original form, a computer program must win the "imitation game" against a human, judged by a human interrogator. A variation of the test that had often been used by researchers to judge the humanness of virtual characters was the 2K BotPrize[1] [ALS09, AML+12, GKB+09, Hin09, Liv06]. The contest had a mix of human players and artificial players (bots) compete against each other in the first-person shooter game UT 2004[2]. During play, a panel of expert judges decided for each player whether they were a human or a bot. As elaborated by Livingstone [Liv06], this type of Turing test is only suitable to evaluate whether agents can play a game like a human would. However, in situations where it is known that agents are not controlled by other users or

[1] `http://botprize.org`, [online: May 2, 2023]

[2] Epic Games, 2004

players, judging believability is a separate challenge; one where research has been even more limited [Liv06]. Defining objective measures for believable agent behavior is difficult and measures must be re-defined with each new application.

Furthermore, behavior is only one aspect that influences IVA believability. Other factors are, e.g., animation, visual fidelity, variability, physics, environmental design, and more. A study that asks whether observed behavior is believable would have to eliminate all other factors, which requires extensive resources and expertise. One solution would be a comparative study that overcomes this complication by providing equal conditions. However, what would the proposed architecture be compared against? There is no off-the-shelve solution that addresses the same aspects. While there are other academic approaches that could be chosen for a comparison, e.g., CIGA [vO14] or SOAR [Lai12], these architectures would have to be adapted to the same virtual environment using the same technological means. Such an adaptation is not possible within the scope of this thesis project. However, even if unlimited resources were available, it would be difficult to define objective evaluation criteria. Additionally, as discussed in Section 1.2, multiple research domains are considered within this thesis. Connecting these domains is in itself a complex task, further complicated by the fact that all are areas of ongoing research. Designing appropriate evaluation strategies for the overall approach is even more complex.

Instead, the approaches detailed in the following sections focus on the evaluation of specific architecture components connected to certain research tasks. To generate and evaluate behavior for multiple agents in non-trivial scenarios, a realization of the architecture concept is required. Therefore, most of the discussed evaluation strategies are based on the traffic agent application described in Chapter 6. The realization of the traffic simulation inherently represents a proof-of-concept evaluation of the presented architecture concept as it shows that the required components are available and work as intended, and that the concept can be applied to a real-time environment. Two evaluations are not specifically based on traffic agents but were realized using the same technological components (see Sections 7.3.1 and 7.3.2). Further evaluations performed within the context of this thesis projected that are not directly related to evaluating behavior generation can be found in Appendices A and C.

7.1 Evaluating the Agent Architecture Design

The foundation of the work presented in this thesis, is the blueprint of an architecture for attentive cognitive agents in real-time virtual environments presented in Chapter 3. The developed concepts represent an appropriate design for such an architecture, providing

an answer to the first research question. As discussed in the introduction of this chapter, evaluating this solution is a difficult task.

To address some of the mentioned issues, steps were taken toward an evaluation framework for cognitive virtual agents. A preliminary draft for a framework was devised that allows gathering behavior-related data in virtual scenarios from both autonomous agents and human subjects (cf. [IVC12, SvS27]). Examples for collected data are navigation, head movement, gaze direction, and performed actions. Data produced by human subjects can be used for requirement analysis and improving the realization of cognitive agents within the according reference scenarios. However, more importantly, gathered data can be used to perform quantitative comparisons between CA^{2}RVE agents, agents realized using other architectures, and human subjects. Furthermore, the data can be used to play back recorded data to a human observer for qualitative comparison studies. The latter allows evaluating the plausibility of generated agent behavior when compared against data from human-controlled entities. Figure 7.1 summarizes the evaluation framework concept. In [SvS11], a similar approach was assessed using human data acquired from a driving simulator. In this approach, detailed comparisons between artificial agents and human subjects were performed by mapping observed human behavior to parameters of the agents' personality profiles using evolutionary strategies. While promising results were achieved by this and other initial attempts, the framework's potential needs to be investigated further in future work.

Instead, the application of the concept, which is described in Chapter 6, corresponds to an empirical evaluation in the form of a proof-of-concept. The underlying objective was to simulate traffic agents beyond typical statistical approaches, but to generate emergent behavior based on the individual actions of the agents. In this context, multiple evaluations were conducted ranging from observing the influence of personality and emotion on predefined behavior to investigating scalability of the approach to assessing the applicability to specific traffic scenarios. Details about these evaluations will be presented in the following sections. These evaluations and the successful realization of the concept show that the proposed architecture is a versatile and flexible foundation for creating real-time cognitive agents within a specific virtual setting. Therefore, the concept fulfills the requirements expressed in the first research question.

7.2 Evaluating Personality and Emotions

In Chapter 4, viable solutions to research tasks **2.1** and **2.2** are provided. A personality profile assigned to each agent ensures behavioral consistency on an individual level. An

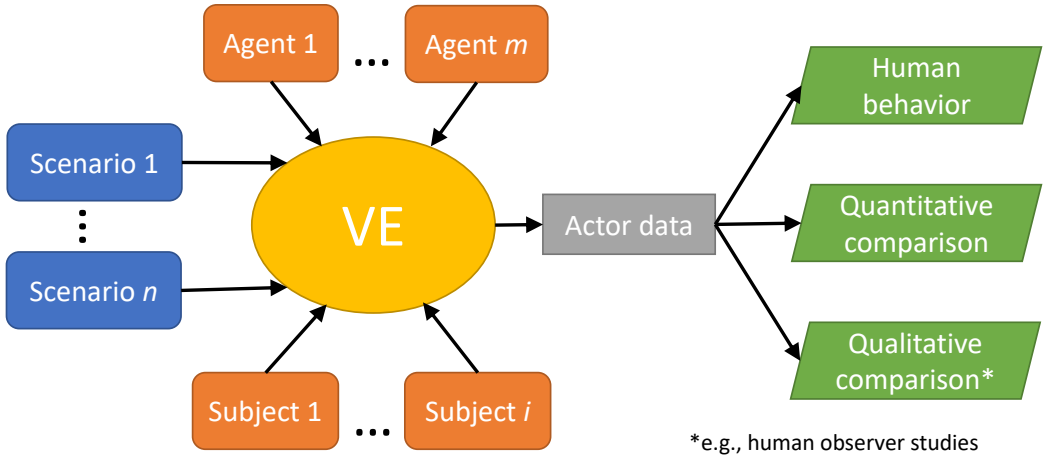
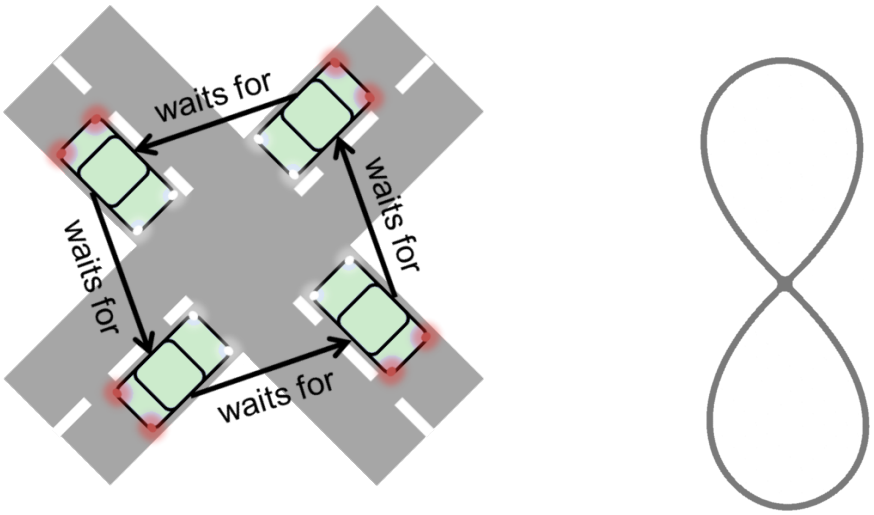


Figure 7.1: Depiction of a concept for a cognitive agent evaluation framework. Autonomous and/or human-controlled agents interact within a virtual environment in scenarios relevant to an intended application. The different autonomous versions could be varying configurations of CA^{2}RVE agents or agents realized using other available agent architectures. Data from all agents is stored as *actor data*, which includes, e.g., navigational data, head movements, and gaze directions. This data can be used to analyze human behavior within application scenarios, perform quantitative comparisons between the different types of actors, or for human observer studies to perform qualitative comparisons.

emotion model dynamically adjusts the underlying personality to be able to react to changing circumstances. To evaluate the addition of personality and emotion to agents, two road traffic scenarios are defined and presented here.

Adding additional characteristics to agents, like personality and emotion, enables the augmentation of standard behavior with alternative mechanisms. To investigate whether this augmentation improves plausibility, a communication mechanism was based on the agents' personality profiles for a typical crossroads scenario. In this scenario, four roads meet at an intersection and right-of-way priorities are not regulated by traffic signs. The results show that plausible behavior can only be achieved if agents coordinate their actions. By adapting decision-making, agent behavior can be improved even further.

A second scenario was designed to apply and extend the implemented mechanisms to another traffic situation. The scenario is similar, but slightly more complex. A two-lane road is blocked by an obstacle either on one or on both lanes. Typical road traffic examples for this scenario are vehicles parked on a lane, a double-parked delivery vehicle, or construction sites. Similar to the crossroads scenario, the evaluation should investigate whether agent behavior is improved by the utilization of personality and emotion. As a result, it was not only possible to observe improved behavior, but also plausible emergent behavior that was not anticipated or intended. Both scenarios are described in the following sections.



(a) (b)

Figure 7.2: Depiction of the "Crossroads" evaluation scenario. (a) Strictly following traffic rules in priority-to-the-right systems frequently results in deadlock situations at an unregulated four-way intersection. (b) The scenario is laid out as a closed loop. Images based on [SvS12].

7.2.1 Personality-based Traffic Behavior in a Deadlock Scenario

A crossroads scenario consisting of a four-way intersection and a priority to the right-system is interesting for evaluation because it frequently creates a deadlock situation if traffic rules are followed unconditionally. If agents approach the intersection from all four sides at approximately the same time, according to German road traffic regulations StVO § 8, each agent needs to yield to the agent approaching from the right.[3] Figure 7.2 (a) illustrates the deadlock situation. The deadlock can only be resolved if one of the four agents yields its right-of-way, or if one of the four agents breaks the traffic rule and starts crossing anyway.

The network created for this scenario consists of two loops of 500 m of bi-directional driving lanes connected by the crossroads creating a closed system, which is equivalent to 2 km of total length (see Figure 7.2 (b)) (cf. [SvS12, SvS13, SvS20, SvS23, IVC13]). At the start of each simulation, vehicles are randomly distributed throughout the network and the desired velocity is set to 30 km/h. Each agent is assigned a personality profile associated with one of three personality classes that were correlated with certain driving behavior and accident involvement in studies by Herzberg and Roth [HR06, Her09]. The studied personality prototypes were *overcontrollers*, *undercontrollers*, and *resilients*, which

[3]Note that the deadlock does not occur if all agents want to turn right. However, it may occur with only three agents if one agent wants to turn left and the other two want to go straight. The described deadlock resolving mechanism works identically for the latter case.

have often been identified as a sufficient minimal set of personality types in psychology research (cf. [DR10]). Overcontrollers tend to exhibit overly precautious behavior, whereas undercontrollers lack self-control and tend to externalize problems, which often leads to more anti-social behavior. People from these two groups have difficulty with regulating emotion. In contrast, resilients are emotionally stable and easily adjust to all domains [DR10, Her09].

Using the results provided by Herzberg and Roth [HR06], the following prototype profile tuples were derived for the simulation: $\dot{\mathbf{p}}_u = \langle -0.06, -0.6, -0.12, -0.54, 0.3 \rangle$ for undercontrolled, $\dot{\mathbf{p}}_o = \langle -0.09, -0.24, -0.75, -0.06, 0.9 \rangle$ for overcontrolled, and $\dot{\mathbf{p}}_r = \langle 0.15, 0.66, 0.6, 0.48, -0.84 \rangle$ for resilient agents. Each value represents one personality dimension as detailed in Chapter 4 as follows: ⟨Openness, Conscientiousness, Extraversion, Agreeableness, Neuroticism⟩. Further details about the definition and generation of personality profiles can be found in Chapter 4 and in [IVC13, SvS12, SvS13]. At the beginning of each simulation, each agent's personality traits are randomized based on one of the three prototype personalities, resulting in individual profiles that are still associated with their prototype. Herzberg showed that the general behavior of the three prototypes relates to traffic behavior [Her09]. According to their results, overcontrolled drivers closely follow traffic rules resulting in more cautious driving and fewer accidents. In contrast, undercontrollers are the most risk-averse and aggressive drivers. The ability of resilients to adjust also translates to their behavior in traffic. Within the scope of this evaluation, resilient driving was interpreted in such a way that resilient drivers disregard certain traffic rules if it helps to resolve conflicting situations, like deadlocks. To incorporate these findings, an agent's politeness factor φ (see Sections 4.1.2 and 6.1) is derived from its personality profile using the following correlation tuple:

$$c_\phi = \langle 0, 0.2, 0, 0.8, 0 \rangle \tag{7.1}$$

The idea is that agreeableness inherently describes someone's tendency of helping others and should have substantial influence on polite behavior. At the same time, conscientiousness was also included, because politeness in traffic is interpreted as abiding to traffic rules. Choosing the coefficients in accordance with Herzberg and Roth's study and the above interpretation yields the lowest politeness for undercontrolled (0.22), a medium politeness for overcontrolled (0.45), and the highest politeness for resilient prototypes (0.76).[4]

An agent's politeness is not fixed, it is influenced by emotion. The CA^{2}RVE architecture provides the ability to define emotion incidents that alter an agent's emotional state, which in turn alters the agent's personality profile and lastly its politeness factor. In a deadlock

[4]Politeness values are normalized to a range between 0 and 1.

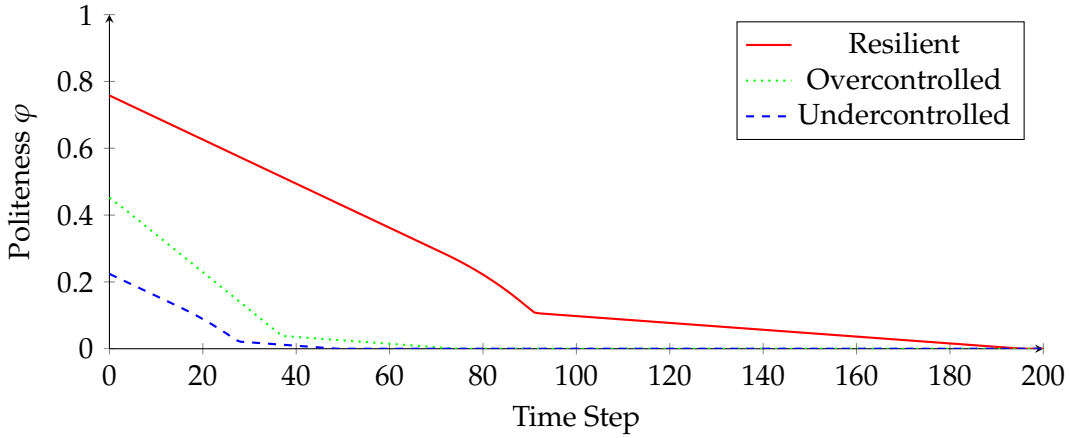


Figure 7.3: While waiting at the crossroads, the politeness φ decreases because negative emotion incidents are perceived. The graph indicates the progression for each of the three personality prototypes as an example. Image based on [SvS12].

situation, positive emotions may be induced if an agent receives its right-of-way from the agent on its left. However, since the effect of this emotional incident would fade away before completing its loop and arriving at the intersection, positive emotions are disregarded in this scenario. However, negative emotions are considered for agents waiting at the intersection, simulating impatience. The negative emotion perception tuple is defined as $\mathbf{s}^- = \langle 0,0,0,0,0.341 \rangle$ since Watson and Clark [WC92a] showed that only the neuroticism trait of the FMM affects the perception of negative emotions. The calibration parameters are defined as $l_i = 0.1$ and $c_i = 0.5$ for this evaluation scenario and the emotional state is updated every second. The correlation between an agent's personality profile and its emotional state was chosen based on further studies by Watson and Clark [WC91, WC92a, WC92b], who correlated the FFM and PANAS (Positive and Negative Affect Schedule):

$$\mathbf{t}^- = \langle -0.1, -0.3, 0, -0.75, 1 \rangle. \tag{7.2}$$

For this proof-of-concept evaluation, three agent types were compared: rule-based (RB), static personality-based (PB) and dynamic emotion-based (EB) agents. While rule-based agents strictly follow traffic rules, PB and EB agents utilize their politeness factor φ connected to their personality profile to make decisions in deadlock situations. In this evaluation, agents can resolve the situation only by yielding their own right-of-way to another agent. Other possibilities like agents taking the right-of-way are not considered. EB agents perceive an emotion event while waiting at the crossroads, which induces negative emotions: $\mathbf{i} = \langle 0.2, 0 \rangle$. The politeness factor's change over time is depicted for each prototype in Figure 7.3.

Simulations were separately run for each agent type (RB, PB, EB). Each simulation lasted 60 minutes and included either 30 or 60 agents, which means an average density of 0.015 or

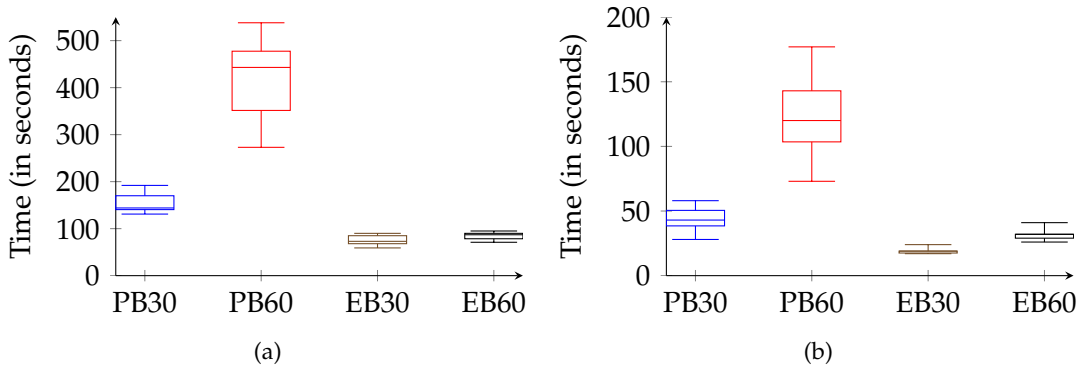


Figure 7.4: (a) The maximum waiting times (at the front of a line) that occurred during each of the 10 simulations for 30 PB agents (PB30), 60 PB agents (PB60), 30 EB agents (EB30), and 60 EB agents (EB60). (b) Instead of the maximum waiting time, the average waiting time of all agents across every test run and configuration is depicted. Images based on [SvS13].

0.03 vehicles/m, respectively. Ten simulations were performed for each of these 6 combinations of agent type and number of agents. According to Herzberg's study, the prototypes were distributed evenly among the population [Her09].

As expected, traffic always stopped completely for the entire system when simulating RB agents. The reason being a deadlock that occurred every time within the first few minutes of the simulation, which cannot be resolved if traffic rules are followed without exceptions. Therefore, results for this agent type are omitted here, because their discussion would not be interesting in this context. For more details, please refer to [SvS23]. For the other agent types, it is interesting to compare waiting times at the intersection as an indicator for plausibility of the simulated behavior. When comparing the maximum time, an agent had to wait as first in line at the crossroads, EB agents spent less time waiting than PB agents. The longest period of waiting time for an PB agent was around 8 minutes when simulating 60 agents. The fact that an agent, after such an extended period of time, would still be yielding its right-of-way to another agent would seem highly implausible to an observer. Even in the case where 30 agents were simulated, at least one PB agent waited for ca. 2.5 minutes. For both traffic densities, maximum waiting times for EB agents were considerably lower at about 1.5 minutes resulting in much more plausible behavior. The maximum waiting times at the front of a line across all simulations are shown in Figure 7.4 (a). In Figure 7.4 (b), the average waiting times for all agents across all simulations are shown. For the latter figure, similar conclusions can be made. PB agents generally wait longer than EB agents. Interestingly, the variance of waiting times is much larger for PB agents than for EB agents. Considering both graphs, this variance is not only due to the agents' initial static personality profile, but also depend on their distribution within the road network and their routing decisions at the

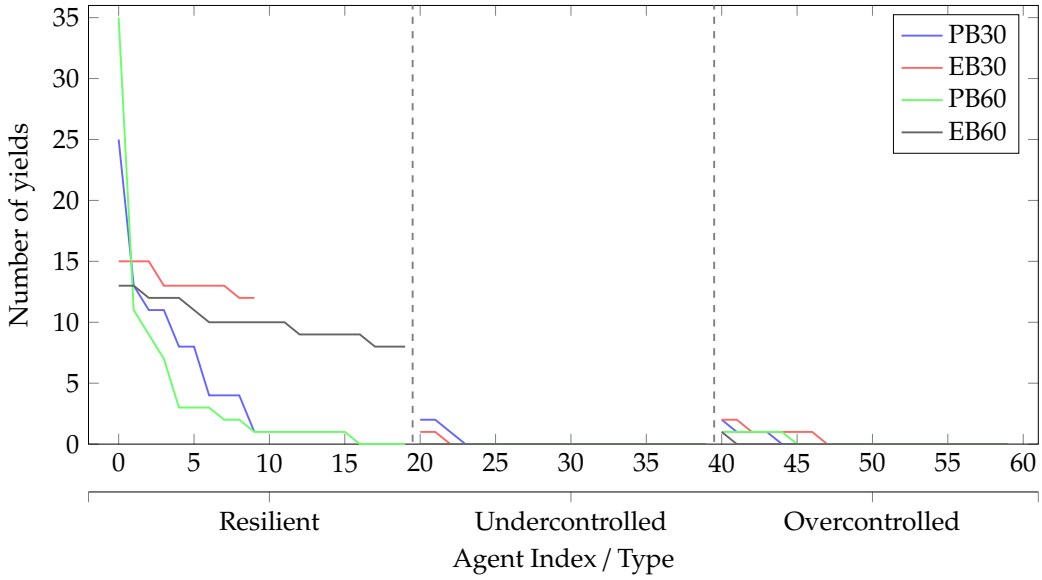


Figure 7.5: For each scenario (30 PB agents (PB30), 30 EB agents (EB30), 60 PB agents (PB60), 60 EB agents (EB60)), one representative sample test run is chosen. The plot shows the absolute number of yields each agent performed during the chosen test run, sorted by number of yields within each prototype group, in descending order. Each group consists of one third of the total population size, i.e., either 10 or 20 agents. Image based on [SvS12].

intersection. The box plot for the EB agent simulations also demonstrate the adaptability of EB agents as the number of agents in the simulation does not influence the maximum waiting times. Consequently, it can be argued that the slight increase of average waiting times is expected as the lines forming at the intersection will be longer.

Since agents that are not waiting at the front of a line cannot make a decision, waiting times are not the only indicator for plausibility of behavior. Another interesting aspect is the difference in number of yields performed by individual agents. Figure 7.5 shows how often each agent yielded its right-of-way during one test run, which was chosen as a representative example. For clarification, each time a deadlock occurs at the intersection, one agent waives its right-of-way to another agent resolving the deadlock. However, due to the definition of the scenario and the chosen traffic densities, the next agent in line will immediately cause the next deadlock, requiring a new yield decision. For each agent type scenario (PB30, EB30, PB60, and EB60) one representative run was chosen for this discussion.

The plot shows that overcontrolled and undercontrolled agents rarely yield to other drivers. Most yields are performed by resilient agents in all scenarios and test runs. These observations meet expectations from the interpretation of the chosen behavior study by Herzberg [Her09]. In the depicted sample test run, one resilient PB agent yields exceptionally often compared to all other agents within the same run. This difference is most obvious in

the PB60 scenario, where one agent performed 40% of all yields during the test run. The ratio is lower for the PB30 scenario, but one agent still performed 25% of all yields, which is significantly more than all other agents. In contrast, yields were better distributed among resilient agents in the EB scenarios. The agent that performed most of the yields, waived its right-of-way about 11% of the time in EB30 and 6% of the time in EB60. These results are close to a uniform distribution of yields among all resilient agents, which would be 10% and 6% for 30 and 60 EB agents, respectively. A uniform distribution represents an optimum result since all resilient agents perform the same number of yields. Considering the distribution of yields as a measure for plausibility, PB agents perform poorly despite the added benefit of showing individual behavior and being able to resolve the deadlock situation. Adding emotion to an agent's personality improves plausibility in these scenarios.

In addition to the total number of yields during a simulation, the number of consecutive yields of an agent is an intuitive factor to assess plausibility of behavior. Considering human drivers, one would expect them to yield their right-of-way in this situation only a few times in a row as they voluntarily increase their own waiting time with each yield. At some point, drivers will lack the patience to willingly wait any longer. For all test runs of both EB configurations, the highest number of consecutive yields was either 2 or 3. This number was implausibly high for the PB scenarios, where the highest number per run was between 5 and 7 for PB30 and between 8 and 19 for PB60. Depending on the situation, even 8 consecutive yields may still be plausible, but will likely be unusual to an observer. At the same time, consecutive yields are an additional cause for the longer waiting times of PB agents. The number of total and consecutive yields, and waiting times are improved by adding emotion leading to more plausible behavior overall.

7.2.2 Evaluating Personality-based Traffic Decisions in a Blocked-lane Scenario

In a second evaluation of the personality and emotion model realizations, agents must deal with a blocked road, which results in the two following, interesting scenarios. Both scenarios are shown in Figure 7.6 (a) and (b).

(1) Blocking one driving lane while keeping the other clear, German road traffic regulations StVO § 6 requires drivers who are blocked by the obstacle to allow traffic on the opposing lane to pass the obstacle before passing it themselves. Thus, agents waiting on the blocked road must wait for an appropriate gap to clear the obstacle or hope for an agent on the opposing lane to slow down, allowing the blocked agent to continue. Alternatively, an agent could start passing the obstacle and force oncoming agents to slow down for them. If the gap is large enough, a waiting agent can pass the obstacle without interfering or interacting with other agents. To provide a more interesting scenario, gap sizes are chosen such

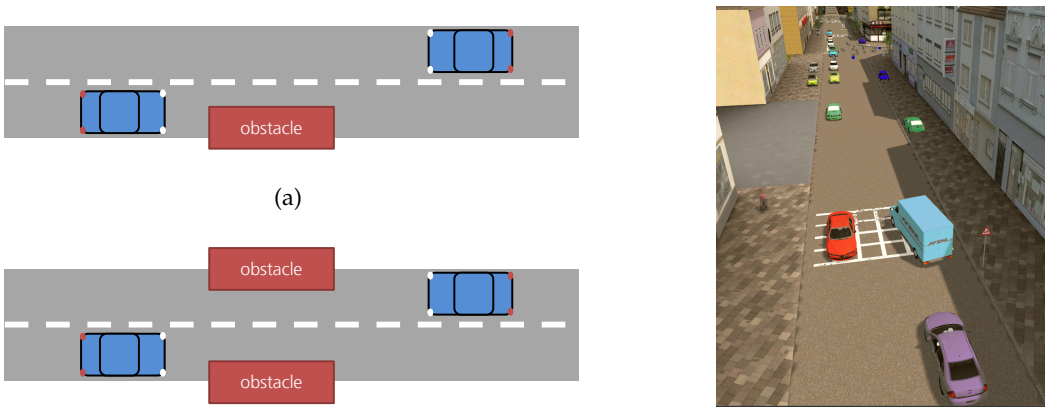


Figure 7.6: Depiction of the "Narrowed Road" evaluation scenario showing an obstacle blocking one lane (a) or both lanes (b). A realization of (a) within the game engine is shown in (c) where the delivery truck is the obstacle. Images (a) and (b) based on [IVC13], image (c)* based on [SvS14].
*Vehicle assets based on designs by "kilastaras" (waiting purple car, red, and gray cars) [kila, kilb], "Underground Lab" (yellow cars) [Und], Dosch Design (cyan cars at end of queue) [Dos]. Dark blue and white cars and light blue delivery truck by IVC.

that interaction is always required. A behavior that can often be observed in human drivers is a decrease of the accepted gap size during prolonged periods of waiting.

(2) Narrowing the road from both sides, an agent needs to use both lanes (the middle of the road) to pass the narrowing, requiring agents that drive into opposing directions to decide who passes first and who must wait. In this scenario, a deadlock situation can occur if waiting agents cannot reach a decision. This situation is similar to the scenario described in the previous section (7.2.1).

These scenarios were simulated using varying configurations regarding agent type, run time, traffic density and distribution, and different parameterizations of the personality and emotion models [SvS14, IVC13]. Since all simulations resulted in similar findings, one configuration for each scenario is reported to represent the results.

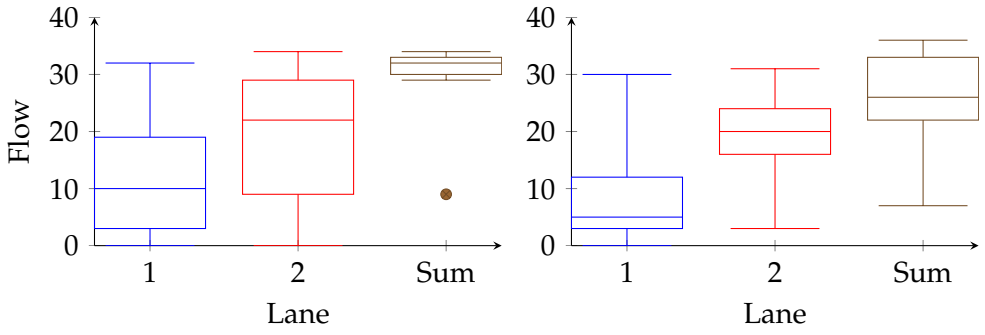
Within the selected configurations, the specified gap length between agents on the free lane in scenario (1) allow waiting agents to pass the obstacle only if they force agents on the free lane to decelerate. Thus, it is possible to utilize the generated gaps to pass the obstacle, but only overly aggressive agents do so. The road is a segment of 500 m in length. Agents are created at the start of both lanes of the road segment with a specified flow (*input flow*) and are deleted at the end of both lanes; measured as *outgoing flow*. The flow is measured to investigate emergent traffic behavior. The input flow on the free lane was 20 agents per minute, creating gaps of about 3 seconds between cars. To avoid congestion, only 12 agents per minute were generated on the blocked lane. In scenario (2), the input flow was 20 agents

per minute on both lanes, forcing agents to cope with heavy traffic by utilizing the deadlock solving mechanism described in the previous section. The test scenarios were set up such that only one agent can pass the narrowed part of the road at one time.

Three agent types were implemented and compared. Strictly rule-based (RB) agents make decisions based on predefined rules (applied traffic law) and can utilize the lane change model adapted from [KTH07] as described in Section 6.1. RB agents are given a static politeness of 0, i.e., they do not consider the disadvantages a lane change would cause for the oncoming agents and show the most aggressive behavior. Identically to the crossroads scenario, personality-based (PB) agents are assigned a static prototypical profile according to studies presented in [Her09] and the prototypes are connected to specific driving behavior. Each profile is generated randomly to fit one of the prototypes. The third agent type is emotion-based (EB), which incorporates all models described in Chapter 4. The following observations were expected when simulating both scenarios:

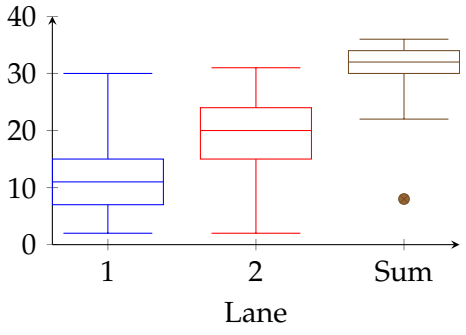
- In scenario (1), the free lane has priority and should have more output flow than the blocked lane.
- In scenario (2), both lanes are equal, and the observed output flow should reflect that.
- EB agents should exhibit the most plausible behavior in both scenarios.

As a straightforward implementation of agents, the behavior of RB agents forms a benchmark for the other two agent types. Figure 7.7 (a) shows the distribution of the outgoing flow values recorded for time intervals of one minute when simulating RB agents in scenario (1). The distribution for the narrowed lane, the free lane, and their combination are displayed. The flow was measured as the number of agents that left the defined road segment in one minute time spans. Flow on the blocked lane (lane 1) was less than on the free lane (lane 2). The median values of both lanes approximately correspond to their respective input flows (12 for lane 1, 20 for lane 2). The combined flow (sum) shows little deviation during the experiment, while the flows of both individual lanes vary noticeably. This effect is due to the fact that the traffic on both lanes alternates in passing the narrowed part. The alternation is a result of the agents' aggressive behavior and the gap sizes resulting from the input flow. While one lane is blocked and building up a queue of waiting agents, the other lane dissolves its queue completely. As soon as the congestion is cleared on the currently flowing lane, the gaps will grow to the size given by the input flow, allowing the waiting lane to start driving and blocking the other lane. The roles of the lanes switch even though one lane has priority over the other. Higher input flow on the free lane causes the queue to grow faster than on the blocked lane, blocking it longer and resulting in temporarily increased output flow.

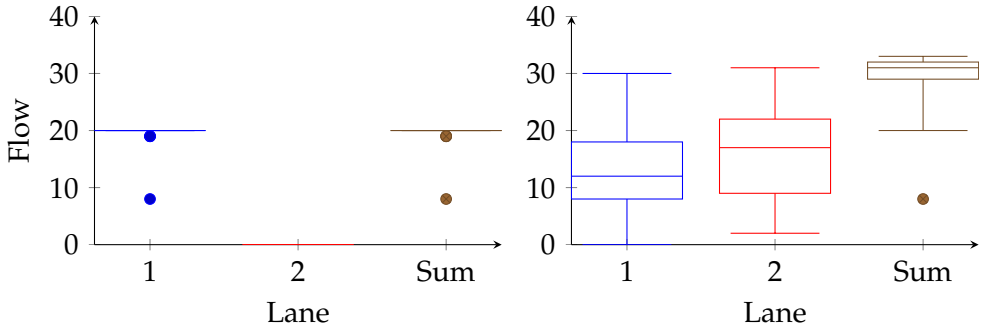


(a) Scenario (1), RB

(b) Scenario (1), PB

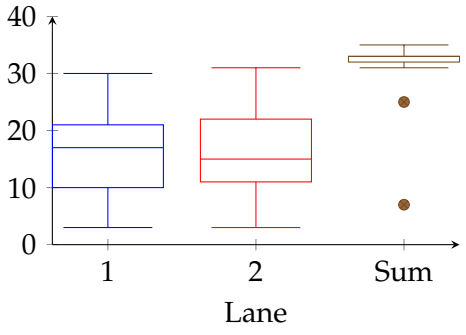


(c) Scenario (1), EB



(d) Scenario (2), RB

(e) Scenario (2), PB



(f) Scenario (2), EB

Figure 7.7: Outgoing traffic flow (agents/minute) for the narrowed road scenario measured during a 60-minute simulation. The figures show results for scenario (1) (a-c) and scenario (2) (d-f) for rule based (RB) agents, personality based (PB) agents, and emotion based (EB) agents.

In scenario (2) RB agents show the least interesting behavior. As shown in Figure 7.7 (d), one lane has a constant flow equal to its input, while the other has a constant flow of 0. This is the result of one agent getting to the narrowed part sooner, entering the bottleneck first and blocking the other lane. Due to a gap size of 3 seconds resulting from the input flow, the next agent enters the bottleneck before the previous one clears it. With the constant flow of the test scenario, this leads to one lane being blocked for the entire simulation. In scenario (1), traffic will not be blocked infinitely despite the same gap sizes on the free lane for the following reason: When passing the narrowed part in scenario (2), agents must decelerate to pass the obstacle decreasing the gap to the following agents. Without an obstacle in scenario (1), agents on the free lane can keep driving at the maximum allowed velocity, keeping the gap size at 3 seconds. This gap size is large enough for the extremely aggressive agents on the blocked lane to start driving. Although, RB agents show efficient behavior regarding output flow (i.e., output flow approximately matches input flow), they do so only by showing overly aggressive behavior. This behavior would not be expected of real-life traffic participants. In scenario (2), RB agents show implausible behavior and inefficient output flows, making this combination uninteresting for further investigations.

Figures 7.7 (b) and (e) show the same scenarios for PB agents. Figure 7.7 (b) shows a smaller median output flow for PB agents than for RB agents on the blocked lane. The median for the sum of both lanes is also lower for PB agents than for RB agents. While the variance is smaller for both lanes individually when compared to RB agents, it is much higher for the sum. The overall smaller output flow on lane 1 results from the fact that, due to their higher politeness, agents consider the gaps on the free lane as too small to pass the obstacle without interfering with oncoming traffic. Although there are also aggressive PB agents, they are not as aggressive as RB agents. Thus, PB agents must wait until an agent on the free lane allows the one waiting at the obstacle to pass. Once PB agents start passing the obstacle, they do not block the free lane until the congestion on the blocked lane is cleared. A polite driver in the queue will wait at the obstacle, allowing agents on the free lane to continue until another agent gives way again. This behavior is the reason for lower variance on the individual lanes: The lanes do not alternate; agents on the free lane always drive and only occasionally give agents on the blocked lane the chance to partly dissolve the congestion. Since the output flow on the narrowed lane is less than its input, the lane fills up until no more agents can be added and only dissolves as much as agents on the free lane allow. PB agents display a more plausible behavior, although it is less efficient regarding traffic flow. In scenario (2), PB agents show improved behavior compared to RB agents. An outgoing flow greater than 0 is observable for both lanes. Although both lanes have the same priority, one lane has a considerably larger output flow than the other. Over the course of the

simulation, the outgoing flow of one lane always dominates the other for extended periods of time (up to 10 minutes) and subsequently the roles of the lanes are reversed. Although PB agents in scenario (2) show more efficient behavior than RB agents, it still seems implausible.

Considering Figures 7.7 (c), the outgoing flow of EB agents combines the median value of RB agents with the variance of PB agents. EB agents start with behavior equal to PB agents, because of an initially neutral emotional state. Due to the modeled dynamics, the following occurs: First, agents will not always wait until being allowed to pass the obstacle by an agent on the opposing lane. Depending on their initial profile, agents might get impatient and force their way past the obstacle like RB agents. This behavior partially explains the almost identical median flow observed for RB agents and effectively models limited patience of human drivers. The resulting shorter waiting times are more plausible to an observer. Second, because agents waiting in the queue also get more impolite over time, those having waited for extended periods of time will more likely follow the first agent of the queue when passing the obstacle on the narrowed lane. However, not all queued agents will follow like in the case of the RB agents. Initially, polite agents with short waiting time will stop at the obstacle and let the free lane resume like PB agents do. Interestingly, this "tailgating" behavior, which can be observed frequently in real traffic, was not explicitly modeled. The behavior emerged from the application of the personality profile and emotion model concepts to this specific traffic scenario. EB agents combine behaviors of PB agents and RB agents resulting in efficient flow and more plausible behavior overall.

The behavior of EB agents in scenario (2) is comparable to the behavior of PB agents in the same scenario. However, by adapting their behavior, the time one lane continuously dominates the other is shorter. While agents on the currently submissive lane wait and become more impolite, agents that are currently driving keep their politeness. The overall effect of adaptation within this scenario can be seen in Figure 7.7 (f): Compared to PB agents in Figure 7.7 (e), the outgoing flows of both lanes are more balanced.

Adding a tool for deterministic decision-making based on an agents' attributes with personality profiles has shown that agents are able to cope with difficult situations that strictly rule-based agents are not able to handle satisfactorily. Making this additional attribute adaptive through emotions, improved plausibility even further. Figures 7.7 (a)-(f) show that personality profiles improve emergent traffic behavior (regarding output flow) compared to rule-based agents, which is improved further by adding emotions.

7.2.3 Summary

In this section, two traffic scenarios were designed to evaluate the addition of static trait-based personality profiles and dynamic emotional states. One scenario consists of a four-way

crossroads in which agents must frequently solve a deadlock situation. In another scenario, agents are prevented from driving by one or more obstacles on their lane creating a narrowed lane. To continue, agents must force their way around the obstacle by making traffic on the opposing lane slow down or stop, or by waiting for an agent on the opposing lane to allow the waiting agent to continue. Three agent types were simulated to compare their behavior: agents with added personality profiles, agents with personality profiles and an additional emotion model, and agents without personality or emotion. The personality is used to derive a politeness factor as a mechanism to resolve the problems arising within the scenarios. A psychology study by Herzberg and Roth was used to assign agents to either overcontrolled, undercontrolled, or resilient personality prototypes and to associate each prototype with certain driving behavior. Further, studies by Watson and Clark were used to define emotion incidents and correlate them with personality profiles. Negative emotion incidents were used to simulate impatience, which increases proportionally to waiting time.

Within the crossroads scenario, the time spent waiting at the intersection and the number of yields were used to judge plausibility of agent behavior. Since rule-based agents were not equipped with a deadlock solving mechanism, they are not able to show plausible behavior and will remain waiting for the entire simulation once the deadlock occurs. While agents with static personality profiles generally improve generated behavior, single "polite" agents will remain waiting at the intersection because they continuously yield their right-of-way to other agents. An observer experiencing this behavior would likely consider it implausible. Adding impatience by the means of an emotion model, waiting times are reduced overall and yields are evenly distributed across *resilient* agents, improving believability. Furthermore, emotional agents yielded at most three times in succession, which is more believable than up to 19 consecutive yields performed by a small subgroup of static personality agents. Additionally, looking at the emergent traffic behavior, emotional agents show that they can adapt to different traffic densities. Waiting times, the total number of yields per agents, and consecutive yields remained stable between the different trials.

In the narrowed road scenario, emergent behavior was measured using traffic flow as it indicates how well agents can deal with the situation. Gap sizes were chosen such that they are initially too small for most agents to pass the obstacle. Agents without personality or emotion were configured to only consider their own advantage while passing the obstacle, making them overly "aggressive". Only these agents utilize available gaps in oncoming traffic to pass the obstacle. While this aggressive behavior creates efficient traffic flows (outgoing flow is approximately equal to input flow on both lanes), the displayed behavior is implausible when observed in detail. When blocking the lane from both sides, agents on one of the two lanes never get to pass the obstacle, creating inefficient macroscopic

and implausible microscopic behavior. Agents with static personality profiles show more plausible microscopic traffic behavior for the scenario with only one obstacle. However, their behavior is less efficient than that shown by agents without personality. Personality-based agents also improve observable behavior when two obstacles are present, but individual agents wait up to 10 minutes at the obstacles, which still seems implausible. As is the case with the crossroads scenario, agents with personality and emotion generate the best behavior of the three agent types. When one lane is blocked, waiting times are shorter for all agents, due to the "impatience" modeled by emotion. Traffic flow is improved, and microscopic behavior is more plausible than that of agents without an emotion model for both narrowed road scenarios. Emotion-based agents also display behavior where agents, which had been waiting, pass the obstacle in small groups. This behavior, which can be observed frequently in real road traffic, was not modeled, but emerged from the interaction between agents.

For all considered scenarios and configurations, adding a personality-based communication mechanism improves behavior. Dynamically altering the personality profiles by emotions further improves plausibility at both macroscopic and microscopic scale.

7.3 Evaluating the Synthetic Perception Framework

The perception framework integrated into the CA^{2}RVE architecture consists of sensing, attention, and memory (see Chapter 5). To evaluate the utility of the framework, these components were realized as prototypes in three real-time game engine scenarios. First, the sensor interface is evaluated by providing different visual sensors and comparing accuracy against performance measurements. From the comparison results, consequences for application and sensor development are derived and discussed. Second, a proof-of-concept evaluation demonstrates whether the attention and memory components fulfill their intended usage. The integrated attention process, handling both bottom-up processing and task-oriented top-down factors, is investigated in specific scenarios. Additionally, different options regarding the combination of different sensor modalities and types are investigated to further evaluate the sensor interface. Third, the framework's applicability and its connection to decision-making processes is evaluated by realizing a traffic-related scenario. By successfully providing sample implementations of the perception components, the three evaluation approaches demonstrate possible solutions to **RT3.1**, **RT3.2**, and **RT4.1**.

7.3.1 Evaluation Sensor Accuracy Against Precision

While designing the synthetic perception framework, a key consideration was the balance between accuracy and computational performance. To investigate the feasibility of the

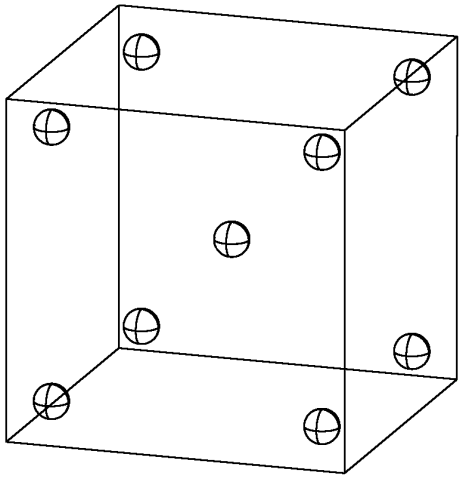


Figure 7.8: Mass points of a cube object can be used to improve simple geometric visibility checks. Image from [IVC8].

approach, a preliminary evaluation compares both aspects with each other. For this purpose, two visual sensors were realized and integrated into the framework and utilized within a defined evaluation scenario. Both sensors are described below, but more details are available in [SvS4, SvS5, IVC8].

7.3.1.1 Sample Implementations of Synthetic Sensors

Geometric Vision Sensor. This sensor realization is based on the geometry of 3D objects in the virtual environment. The overall process is aligned in a visual sensing pipeline. First, preselectors identify separate sets of interesting objects out of the set of all objects in a scene. Applying physical filters, the union of these sets is reduced to a subset representing the sensor's capabilities regarding the environmental stimulus. The exemplary implementation applies an elliptic field-of-view filter (cf. [RD08]). Every virtual object outside the ellipsoid around an agent is filtered from its visual perception. The ellipsoid, which is parameterized using view range and view angle, approximates the visual field of view of a human more closely than view cones, which are commonly used. The second filter is a raycast-based line-of-sight filter. All objects not filtered by the previous operation are checked for visibility by casting a ray from the agent's sensor origin to the tested object. If an intersection between the ray and another object exists, the tested object is occluded, and it is discarded. To counteract the effect of underrepresentation by checking only a single point for each object, multiple *mass points* are used to test visibility [TR95] (see Figure 7.8). During a sensing step only one ray is cast to one randomly chosen mass point, but objects are kept in short-term

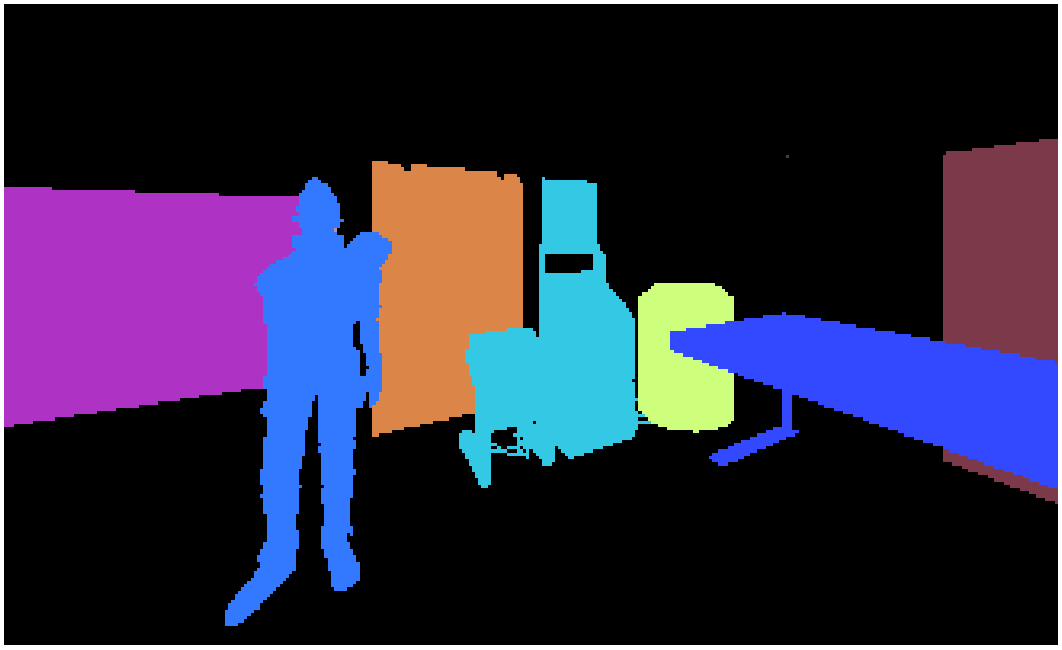
sensory storage longer to benefit from random sampling.

False-colored Vision Sensor. The other sensor used for the sensor evaluation is based on the false-coloring approach described by Noser et al. [NRTMT95, NT95]. Using a rasterization approach, each object is assigned a unique color and each agent's view is rendered into a low-resolution off-screen buffer using these unique colors without any lighting effects. An example is shown in Figure 7.9. Since this approach utilizes the regular rendering process, it automatically determines occlusions. In the resulting image, the unique colors can be used to identify the objects visible to an agent. As soon as a pixel is colored with a certain color, the object associated with the color-ID must be visible in the agent's current field of view. Once the visible objects are known, every additional information can be retrieved from the object itself or using appropriate semantics. In this sample implementation used for evaluation, the single occurrence of one color-ID is sufficient to perceive the associated object. If required, the number of all pixels of the same color could be used to apply additional heuristics, like a threshold. For example, a stimulus could be created only for objects that cover a certain number of pixels. This approach provides an intuitive way of emulating human vision without the need for complex object detection algorithms. It can be configured to be highly precise and could even utilize two cameras for stereo vision. However, the approaches' biggest advantage is also a disadvantage. Rendering separate views for each individual agent during each sensing step is a computationally expensive operation which does not scale well. Reducing the resolution for the image buffer improves performance but decreases accuracy. Using a stereo camera setup to counteract the loss of accuracy while using low resolution buffers would defeat the purpose of lowering resolution. Instead, a camera tremor – commonly called jittering – can be applied to the sensor's camera, which will add a small yaw angle α to the camera. During each sensing step, the rotation is altered between $0°, -\alpha$, or $+\alpha$, i.e., only one image is generated per step and agent. If visible parts of an object cover less than one pixel, only slight differences in viewing direction will change the resulting color of that pixel in most cases. Thus, by keeping stimuli in STSS for more than one sensing step, the tremor can provide a more accurate description of the scene. However, using a tremor, slightly occluded objects can become visible. At the same time, slightly visible objects may become occluded resulting in false positive and false negative results, respectively. If a small tremor is used, both effects are negligible.

7.3.1.2 Sensor Scenario Setup

The hypothesis is that the realized sensors can be used as base-line implementations: the geometric sensor for performance and the false-color sensor for accuracy. In the evaluation,

(a)



(b)

Figure 7.9: Example of an agent's view using normal rendering (a) and using the false-color approach (b). Images based on [SvS5].*

*Scene assets based on designs by Unity Technologies [Uni].

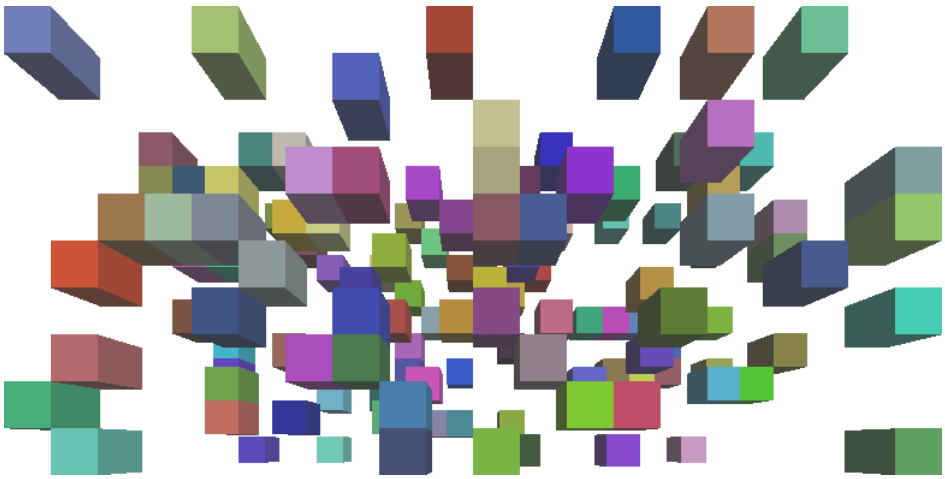


Figure 7.10: Setup of the sensor evaluation from an agent's point of view. 16 of 137 cubes are fully occluded, i.e., not visible to the agent. Image based on [SvS5].

both sensors are compared to each other in these two categories. An artificial evaluation scenario was generated that includes simple cube-shaped objects randomly distributed on a 3D grid for the accuracy evaluation. The distribution shown in Figure 7.10 was determined once and did not change between trial runs. During each trial, a stationary agent used one of the sensor configurations from Table 7.1 to sense the objects. The true number of visible cubes was determined manually, by providing discernible colors to each cube and counting the objects in a high-resolution render of the agent's view. More details about the scenario setup can be found in [SvS5] and [IVC8].

The geometric vision sensor was used in configurations A and B. In A, only the origin of the object (midpoint of the cube) was considered for the ray cast-based visibility check. In contrast, one of nine mass points (see Figure 7.10 (b)) was randomly chosen during each sensing step and checked for visibility in configuration B. In configurations C through K, the false-color-based sensor was used for sensing. Each pair of configurations (C and D, E and F, G and H, I and J, K and L) utilized textures of the same size, but either kept the sensor camera stationary or applied a tremor of 0.3° (see Section 7.3.1.1). To obtain stable results, every sensor configuration from Table 7.1 was simulated for 60 s during the accuracy evaluation. STSS retention was set to 0.5 s and sensing intervals were 0.1 s. The number of STSS entries was recorded after each sensing step, representing the number of sensed objects.

For the separate performance evaluation, the cubes were not randomly distributed across the scene but placed on a 2D grid (i.e., objects were never occluded). The number of cubes in the scene was gradually increased during each trial run. STSS retention rate and sensing step intervals were identical to the accuracy evaluation. The number of stimuli contained in the STSS was recorded after each sensing step. Configurations C through J were simulated

Table 7.1: Evaluated sensor configurations (view angle: 120°, view range: 100 *m*). Table based on [IVC8].

Type	ID	Configuration	ID	Configuration
Geometric	A	without mass points		
Geometric	B	with mass points		
False coloring	C	16^2 = 256 px, no tremor	D	16^2 = 256 px, tremor 0.3°
False coloring	E	32^2 = 1024 px, no tremor	F	32^2 = 1024 px, tremor 0.3°
False coloring	G	64^2 = 4096 px, no tremor	H	64^2 = 4096 px, tremor 0.3°
False coloring	I	128^2 = 16384 px, no tremor	J	128^2 = 16384 px, tremor 0.3°
False coloring	K	256^2 = 65536 px, no tremor	L	256^2 = 65536 px, tremor 0.3°

twice. In the second simulation, analysis of the texture colors was performed in parallel with support of the graphics hardware. For configurations K and L, only the parallel variant was simulated since a sequential analysis on the CPU was no longer feasible. The trials were run using an Intel Core i7-2600K consumer-level PC with 8GB RAM and a NVIDIA GeForce® GTX560.

7.3.1.3 Results and Analysis

Considering accuracy, no sensor type was "better" than the other because results depend on the sensor configuration, as indicated by Figure 7.11. Using configurations K and L, the false-coloring sensor was able to sense the scene at 100% accuracy, i.e., all 121 visible cubes were sensed. For configuration J, results were up to 100% accurate at times as well. Regarding configurations C and D, the false-coloring approach was outperformed by the geometric sensor in both configurations. Using configuration A, the geometric sensor was able to sense 91 objects. While simulating configuration B, between 108 and 119 objects resided in STSS at any given time.

As shown by the results, accuracy for the geometric sensor depends on the distribution of mass points. Testing only the center of the cube for visibility increases the probability of identifying visible objects as occluded in cluttered scenes. The probability of detecting an object can be improved by increasing the number of checked points per object (i.e., increasing the number of mass points), but still depends on scene layout and point distribution within the object itself. With the given layout, the geometric sensor never kept all 121 visible cubes in STSS at the same time. Since only one mass point was chosen randomly at each sensing step, configuration B showed better, but unstable results. Stability could be improved by checking more points per sense step or by increasing STSS retention rates.

While the geometric sensor was unable to achieve 100% accuracy using the tested con-

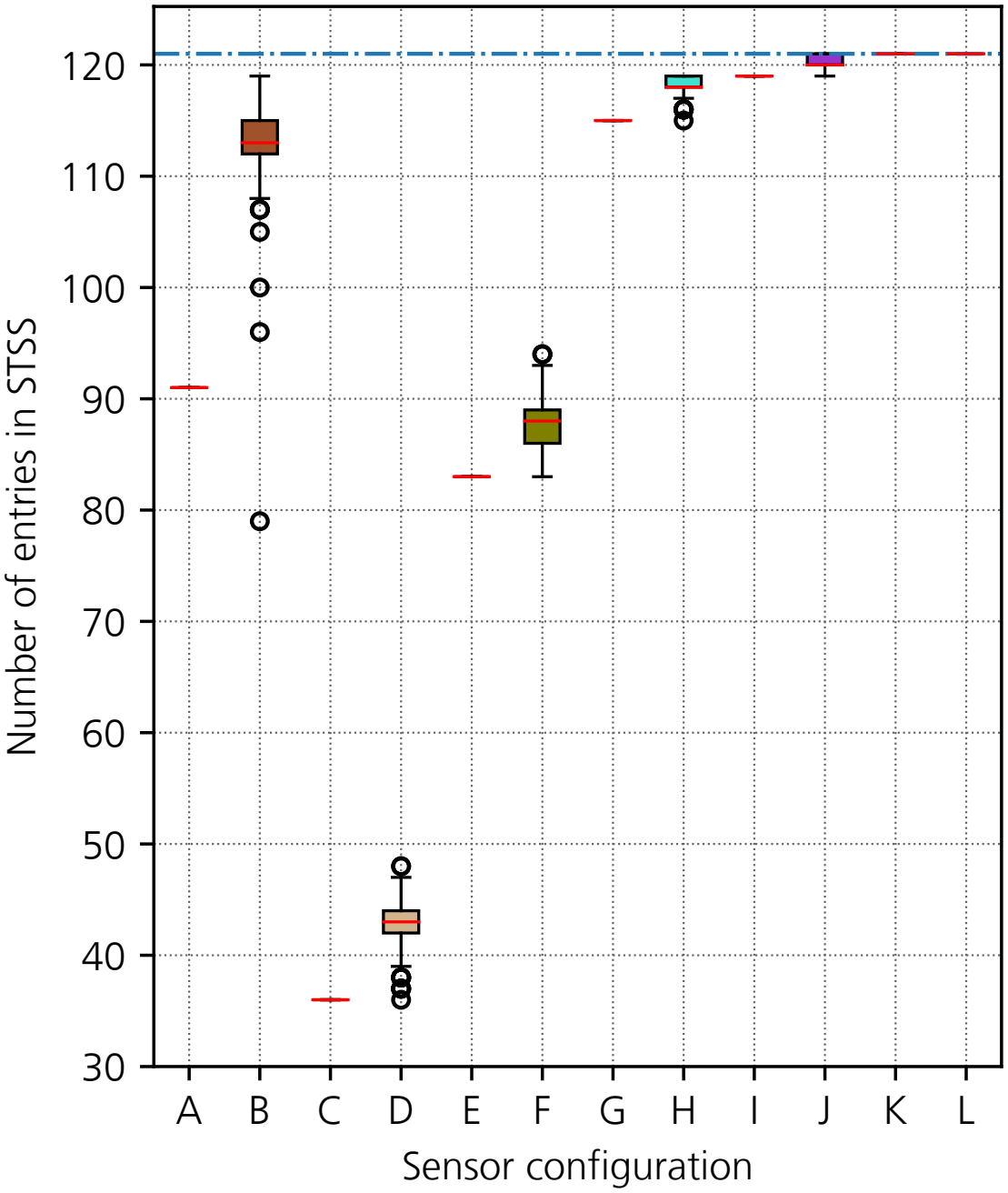


Figure 7.11: Results of the sensor accuracy evaluation. 121 of 137 cubes were not fully occluded (horizontal line). The sensor configurations from Table 7.1 (A–L) were applied. The number of entries in the STSS for each configuration is shown. Image based on [IVC8].

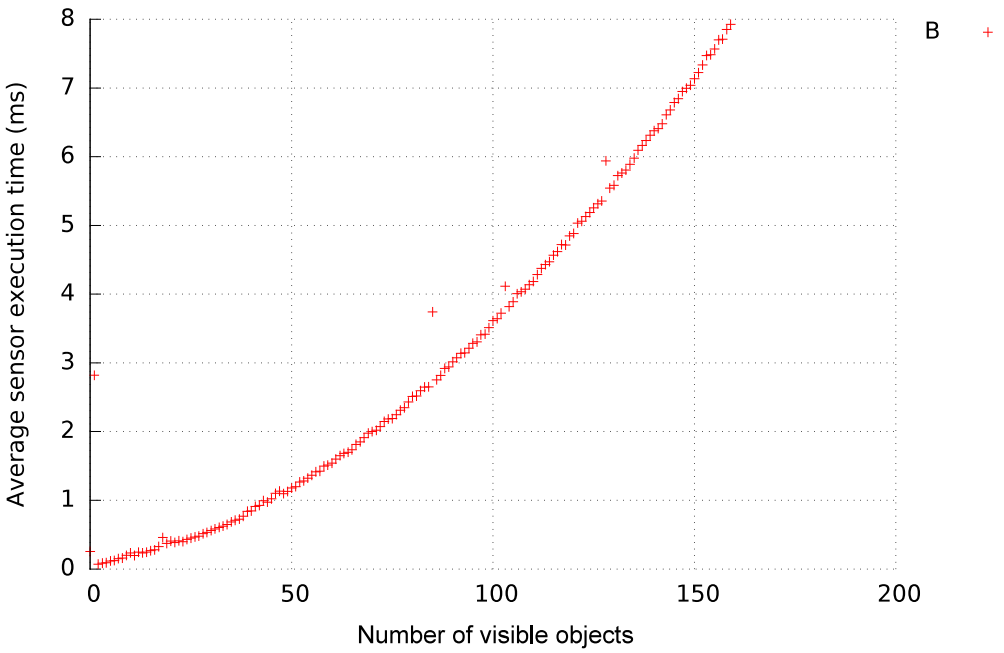
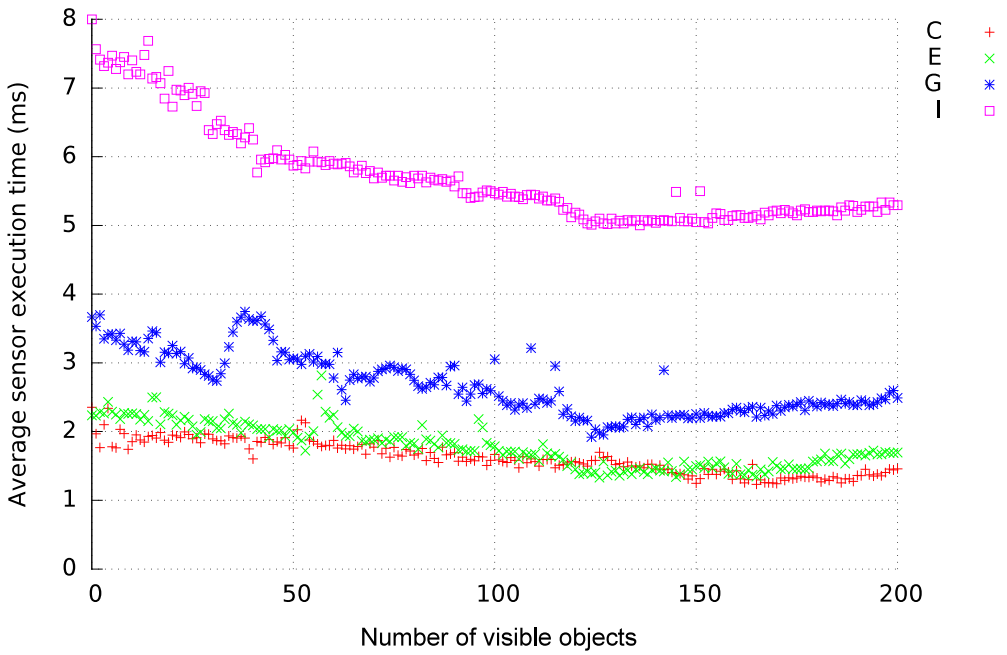


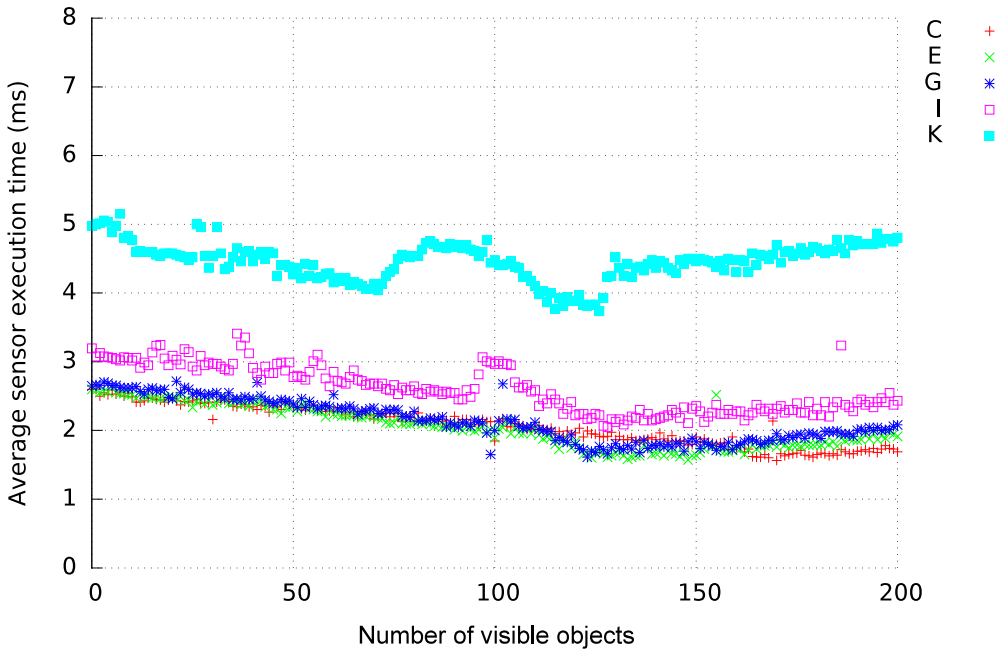
Figure 7.12: Average sensor execution times in ms for the geometric sensor in configuration B. Cubes were added every 5 *s* from 0 to 200. Cubes never occluded each other.

figurations, it produced predictable results at a level of accuracy that is likely acceptable for many applications. In contrast, the false-coloring sensor strongly varied in accuracy depending on the resolution of the used image buffer. Using a resolution of 64*x*64 or larger (G-L) the false-coloring sensor outperformed the geometric sensor regarding accuracy. Given a resolution of 128*x*128 (I, J) almost all 121 visible objects could be sensed. A resolution of 256*x*256 (K, L) was already enough to correctly identify all visible objects in this setup. If the image resolution for the color buffer is too small, the rasterization process performed by the render pipeline can cause pixels of visible objects to be overwritten by other colors. In this case, the trade-off for increased performance is decreased accuracy, such as in configurations C–F. Adding a tremor to the camera generally improves accuracies as it mitigates the rasterization problem. Consequently, if image resolution is sufficiently large, the tremor effect becomes unnecessary.

Execution times for the geometric sensor depend on the number of rays cast during each sense step. This number is influenced by the number of tested objects and how many mass points are checked per object. During the evaluation, only one mass point was chosen randomly per sense step. Therefore, the only influencing factor was the number of objects. Results are shown in Figure 7.12.



(a) False-coloring (serial)



(b) False-coloring (parallel)

Figure 7.13: Average sensor execution times in ms for the false-color sensors in relation to the number of perceivable cubes. In (a), the execution times for configurations C, E, G, and I are shown for the serial calculations, and in (b) for parallel calculations. Configuration K was only recorded in parallel execution. Cubes were added every 5 *s* from 0 to 200. Cubes never occluded each other.

Execution of the false-color sensor is theoretically influenced by two factors: the number of objects being rendered by one camera and the resolution of the image buffer. Figure 7.13 (a) and (b) show the results regarding the number of visible objects for all sensor configurations. Increasing the number of objects increases the number of triangles to be rendered by the graphics hardware, which increases render times. However, for modern graphics hardware this effect is negligible in this evaluation scenario. The number of triangles in the chosen setup is not enough to challenge the utilized hardware. The results reflect this observation.

In contrast, increased image resolution does increase execution times significantly. Obvious parts of this increase are longer render times as more pixels need to be filled and more data is transferred between GPU and CPU for the subsequent analysis step. Additionally, when the resulting false-colored image is analyzed sequentially, a hash set structure is used to eliminate duplicate color-IDs. This reduction operation is of $\mathcal{O}(n)$ complexity providing another explanation of the observed increase in execution times. For the GPU-based image analysis, a compute shader is realized and dispatched, which performs the ID reduction concurrently on the GPU writing the results to a 1D output buffer. As a result, the parallelized variant outperforms the serial implementation. The improved performance enables the utilization of the false-coloring approach for texture resolutions that are infeasible for the serial approach, e.g., 256×256 px (K and L). The increase in execution times between configurations is mostly explained by data transfers between the CPU and GPU. This explanation is consistent with the fact that the serial version is marginally faster for low resolution images (configurations C and E).

7.3.1.4 Comparison

As indicated by the results, both approaches are configurable such that any accuracy can be achieved. Either the number and distribution of mass points must be large enough for the geometric sensor, or the texture size must be sufficiently large in case of the false-color sensor. The maximum accuracy is only achievable in theory, as simulation should remain possible in real-time, i.e., with at least 30 frames per second. For a small number of objects (≤ 100), the geometric sensor provided better performance (see Figure 7.11) than the false-color sensor with the given configurations. With these numbers, checking each object separately outweighs the cost of rendering a separate image. Having to process a larger number of objects favors the false-color sensor as it profits from highly optimized graphics hardware. In conclusion, developing an application using the provided ideas, a designer can control accuracy and performance by configuring the sensors to the application's needs. At the same time, sensors can be combined or used situation dependent. For example, geometric

sensors could be used per default and swapped with a false-colored sensor if higher quality results are required, and performance permits it.

7.3.2 Proof-of-Concept Evaluation of the Attention Module

To evaluate the capabilities of the proposed attention approach, three different scenarios were defined. The objective was to recreate human behavior in typical every-day situations regarding attention. All scenarios explore the abilities and disabilities of virtual perception including physical and cognitive limitations. The focus was to show that the concept of the perception framework can be used as a proof of concept. First, a description of exemplary implementations of the framework's main elements (namely sensors, memory, and attention) is provided, followed by an explanation of the chosen scenarios including their intention and setup. Results have partially been presented in [SvS21] and [SvS22], but will be analyzed and discussed here in detail. Finally, the results are analyzed and discussed.

7.3.2.1 Sample Implementations of Perception and Attention Components

GPU-based Vision Sensor. A sensor based on the idea of false coloring as described in 7.3.1.1 provides visual information to an agent. Due to performance considerations, a parallel analysis of the result buffer is performed on graphics hardware (cf. Section 7.3.1.2). Nevertheless, simulating larger numbers of agents in a scenario will result in performance issues. As discussed Section 7.3.1, rendering individual views for each agent is a costly operation, especially for high resolution buffers.

Pre-computed Visual Importance. Precomputed radiance transfers or image-based lighting (IBL) is a technique used to achieve real-time global illumination in game engines. Light probes, which are distributed across a virtual scene, store light transport information, which is pre-computed using an offline process, i.e., before the game starts. At runtime, these probes can be sampled to efficiently retrieve global lighting information (e.g., see [ARM15]).

The idea of IBL is adapted to further improve scalability of the GPU-based vision sensor. *Saliency probes* store cubemaps that encode visibility information and saliency values of static virtual objects for a given scene. Both cubemaps are generated from the position of the probe. One cubemap is constructed using false-color rendering (see 7.3.1.1) and the other cubemap consists of saliency maps. Figure 7.14 demonstrates the generation process.

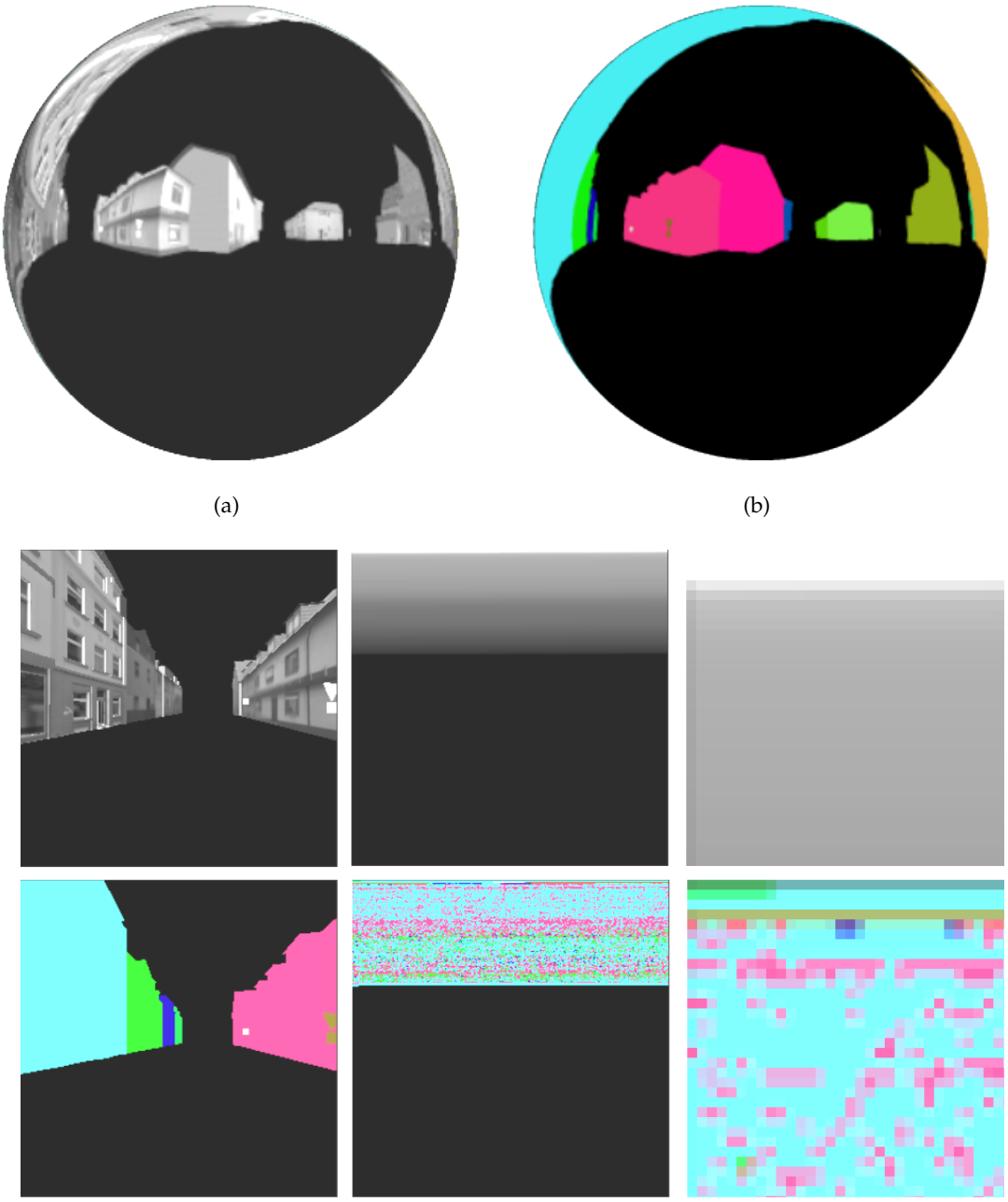
The complexity of the process that generates saliency maps depends on the application. One approach is presented by Itti et al. [IDP03], who strive to provide a solution that is biologically accurate. For demonstration purposes, the realization described here uses a straightforward approach where a pixel's color intensity describes the saliency of an object

at that pixel position. During construction, the false-colored cubemap and the saliency cubemap are sorted according to saliency values. By sorting both cubemaps in conjunction, the mapping from saliency to virtual object is preserved. During the simulation, probing the closest saliency probe provides approximated information about the most salient objects that are visible from the query position in descending order. Since cubemaps are pre-computed, the complexity of saliency calculations is irrelevant at run-time allowing application developers to focus on accuracy of the calculations. Additionally, all agents can query this information without an impact on performance. Obviously, as is the case with all offline computations, the approach only works for static information. Changing a scene invalidates all encoded information. To circumvent this problem, saliency probes store information only for static objects. Agents can still query the probes but must combine retrieved results with those of other sensor instances or types to include dynamic objects in their process.

Auditive Sensors. To provide a second sensor modality in the evaluation scenarios, an auditive sensor is attached to the agent. Similar to vision sensors, the complexity of sensors receiving audio signals depends on the application and what is to be achieved. For the proof-of-concept prototype a straightforward realization is chosen. All agents can potentially sense all objects with an attached audio property, but only signals above a specified volume threshold pass a pre-selection test. A proximity filter excludes all objects that are not within a certain distance to the agent. Obviously, the interaction between sound waves and auditive receptors as well as the physics of sound propagation are much more complex than the process described here, but a more detailed realization is not required for the proof-of-concept implementation. Currently, auditive stimuli are generated from all sound emitting objects that pass pre-selection and distance filtering. Volume and saliency are encoded within the stimulus using object semantics.

Application-Specific Sensors. While the prevalent motivation is to generate plausible behavior for virtual humans, run-time performance often surpasses the importance of accuracy. Due to the design of the proposed framework, sensor types and complexities can be tailored toward an application's specific needs. For example, a sensor can emulate a human sensory organ, but it could also work completely different to provide the same or a subset of information provided by the original. In the third of the evaluation scenarios described in the following section, indirect visual cues, i.e., reflections, are integrated into the sensing process using such an application-specific sensor.

To include reflections, the described GPU-based vision sensor could be extended in such a way that reflective surfaces in a camera view are not only represented by object IDs, but



(a) (b)

(c)

Figure 7.14: Visualization of a saliency probe, which is a combination of a saliency cubemap (a) and an according false-color cubemap visualizing object IDs (b). Saliency values are derived from pixel intensities in a rendered scene view at the probe's position. Each side of the cubemap textures is sorted according to saliency, exemplified for one side in (c): original saliency values (upper left), original object IDs (lower left), sorted saliency values (upper middle), object IDs sorted according to saliency values (lower middle). Images on the far-right show versions of the sorted maps zoomed in on the upper left corner. Images from [SvS22].*

*Building assets by IVC.

additionally reveal object IDs of all objects reflected by the surface. To avoid the additional complexity in design and run-time performance, a geometric vision sensor (cf. [PCR+11]) is used to provide additional visual information to the agent. The sensor uses proxy geometry within a scene to cast rays towards perceivable objects in the scene. If an emitted ray is not blocked by other scene geometry, the target object of the ray is considered visible as a reflection on the surface of the reflective object.

Other application-specific sensors could extract semantic information from a scene providing efficient access to predetermined knowledge. One example of a *semantic sensor* is discussed in Chapter 6. The sensor senses elements of a semantic road network layer, which encode traffic rules and other traffic related information.

Memory. Memory layout and information encoding is realized in accordance with the concept defined in Chapter 5. A short-term sensory storage (STSS) is used to store stimuli provided by all sensors. After being attended to, a stimulus is turned into a percept and moved to short-term memory (STM). Stimuli and percepts are stored in flat lists providing access to the associated virtual object and its perceivable properties. While not nearly representing human memory, the representation suffices within the given evaluation scenarios. If objects are not sensed again or moved to STM, they are removed from STSS after a memory decay time of 0.5 *s*. Percepts are retained in STM for 5 *s* from the last time they were queried by other processes. The decay times where chosen empirically for the test setups, but can be freely configured by users of the architecture. Since STM represents an agent's working memory, a LTM that encodes persistent and global information is not required in this evaluation and therefore omitted.

Attention. In accordance with the perception framework design, attention is used for selecting stimuli, simulating object recognition and information retrieval processes, and realizing inhibition of return. The selection process is based on saliency values assigned by sensors to stimuli during sensing. Possible sources for these values are object semantics, internal calculations by a sensor, or other sensors. For example, the described saliency probes can be used by a sensor to assign a pre-computed saliency value to a stimulus. At the same time, the probes can be queried by a sensor to retrieve only the object color-IDs, which are then used to look up visual saliency of the objects via semantics.

Saliency values for a single object are linearly integrated across all sensors to determine the object's multi-sensory value. The current object of interest is selected based on the sorting of saliency values as described in Section 5.5. The agent's actuators are requested to orient the primary visual sensor towards that object. Once aligned, agents will fixate the object for

a dwell-time period simulating object recognition and information retrieval. The duration of dwelling can be set according to the current application and could vary with object type to consider its complexity or with an object's saliency. In this sample realization, typical value ranges for dwell time as reported by Petersen et al. [PKB12] are used to randomly select dwell time duration between 0.2 and 0.5 seconds for the current stimulus. The stimulus is considered as attended to and stored as percept in STM after the agent has fixated the stimulus' source for the specified duration. After being attended to, the stimulus is inhibited in the subsequent sense step(s).

In the current realization, the saliency value of an attended stimulus is reduced to zero if the associated object's saliency has not increased since the last sense step. The latter is used to keep an object "interesting", e.g., due to constantly changing properties like color or position. Alternatively, instead of immediately setting saliency to zero, a function can be used to attenuate saliency values of attended objects over time. This will preserve the object of attention across multiple sense cycles if saliency remains higher than the saliency of other sensed stimuli. Additionally, the number of inhibited objects is determined by a parameter. If set to one, agents would switch between the two most salient objects in an unchanging scene. If the value is equal or larger than the total number of objects in the scene, agents will look at each object in their field of view in the order of their saliency values.

The purpose of top-down attention is to provide an agent with information that it is interested in, e.g., objects that are related to the agent's current task or goal. Agent architectures often realize this concept using subscription or filter interfaces. Agents can use these mechanisms to register their interest in specific objects, object classes, or events (e.g., see [BBT99, vOD11]). Using this approach, information of interest will always reach the agent's conscious, regardless of other stimuli that are present at that time. In the proposed realization described here, top-down attention is emulated by using an agent-centric bias to an object's saliency to simulate more realistic attention selection (see Section 5.5). Providing a task-based bias will guide attention towards specific objects or object types, but the approach ensures that highly salient objects will not be ignored in favor of task-relevant objects within the proposed attention mechanism.

Personality and Emotion. The aspects personality and emotion in perception and other cognitive processes were addressed throughout this thesis. However, their effect was not included in the proof-of-concept realization as it did not provide a benefit within the context of the evaluation scenarios that are described in the following section. For other scenarios task parameters could be defined that influence, e.g., memory decay, attention inhibition, sensor accuracy, or the number of objects that can be attended to. A task parameter's value

during a sense step is derived as a combination of traits from the current dynamic personality as exemplified in Chapter 4.

7.3.2.2 Attention Scenario Setup

To assess the proposed perception framework, the sample implementations described in the previous section were applied to three evaluation scenarios. The scenarios demonstrate that behavior generation can be driven by the realized attention process.

Scenario 1: Imagine yourself leaving an apartment, house, or office. At the door you try to remember whether you noticed your wallet on the way to the door. The wallet is placed in plain view and easily observable. However, if you were to be distracted by a blinking cell phone display while walking past the wallet, would you have perceived the wallet?

Scenario 2: This scenario is identical to scenario 1, but the distraction is caused by an auditive stimulus to investigate the interrelation of sensor modalities.

Scenario 3: For the third scenario imagine walking along a sidewalk. Shortly before reaching a corner, another pedestrian suddenly appears in front of you. Fortunately, you quickly notice the other person and you can evade them. But in another case, you may not look where you are going because you are distracted by a salient visual cue. As a result, you are unable to perceive the other pedestrian and you cannot prevent the inevitable collision. Because you are walking alongside a house, you can only gather information about the other pedestrian once it becomes available, i.e., as soon as you can see or hear them. However, this does not necessarily require a direct line of sight. Looking for reflective surfaces, e.g., could be a strategy to acquire additional information. The distraction could come from a bottom-up stimulus or be based on a top-down goal.

In all scenarios, a virtual human should show the behavior that is generally expected form a real human. Of course, this cannot be a universal statement, but needs to be viewed within the constraints of the current realization. To investigate whether the defined agent perception framework can be used to generate the described behavior, framework components were implemented alongside the evaluation scenarios using the Unity game engine[5]. An overview of the scenario realizations can be found in Figures 7.15 and 7.16.

In scenarios 1 and 2, an agent must navigate a small apartment structure. The agent's walking path towards the exit is depicted in Figure 7.15 by a red, dashed line. Two salient objects are placed on opposite sides of the path to the exit. The first is an object of interest ("target") and the second is a distraction. In both scenarios a cell phone is used as the distractor. In scenario 1 it produces a visual stimulus and in scenario 2 an auditive stimulus. In addition to these objects of interest, the virtual apartment includes multiple other objects

[5]Unity Technologies, `https://unity.com/`

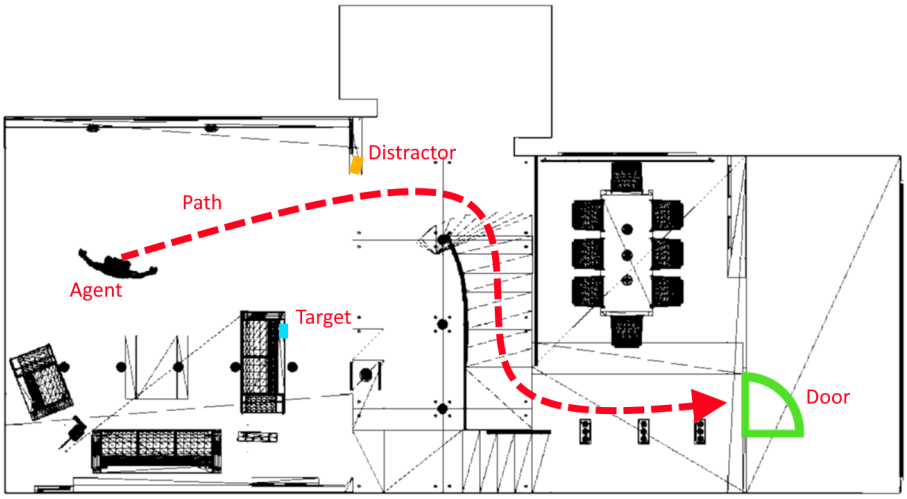


Figure 7.15: An overview of scenarios 1 and 2 used to evaluate the proposed attention approach. The agent walks along the indicated path to reach its target destination (the door). The salient target object (blue) is located on the back of the small sofa. The distractor object (yellow) is located on a structure on the opposite side of the walking path. In scenario 1 a visual cue distracts the agent and in scenario 2 an auditive cue is used.*

*Scene assets based on designs by Unity Technologies [Uni].

that are all perceivable. Both scenarios are simulated twice using two different specifications. The first specification includes only one object that produces a constant, highly salient stimulus s_0, which is the target object. The saliency here is due to a top-down prioritization, simulating that the object holds a certain value to the agent. For the second simulation, a visual (scenario 1) or auditive (scenario 2) stimulus with high saliency values (s_1,s_2) are generated at a specific point in time. The generated stimuli both have a higher saliency than the baseline saliency, i.e., $s_1, s_2 > s_0$. The results of running the simulations using both specifications are shown in Figure 7.17.

The third scenario includes agents A and B, both of which are following a specific path on the sidewalk. The buildings in the scenario prevent both agents from establishing a line of sight. However, across the street from agent A, a large store front allows the agent to gather additional knowledge by using an indirect line of sight. In this scenario, a moving car is used as a distractor and a traffic sign is used as the target object (which is always visible). To prevent any crossover effects, Agent B does not react to agent A, nor does it alter its actions. The paths of the agents are indicated by dashed lines in Figure 7.16. The path of the distractor is represented by the solid line. The reflective store front, agent A's target object, and the point of collision are also marked in the image.

The intention of defining the third scenario was to demonstrate several capabilities of

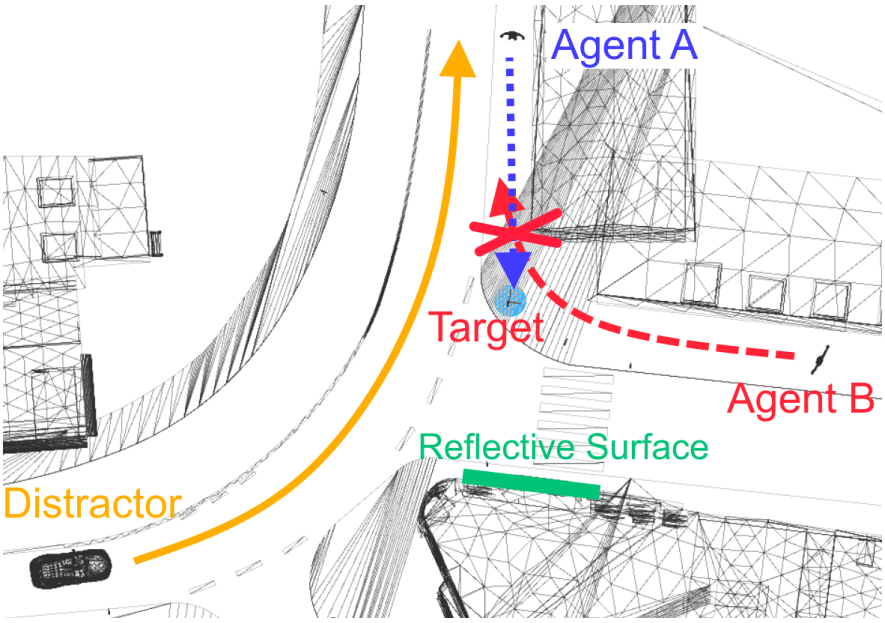


Figure 7.16: An overview of the third sample scenario used to evaluate the proposed approach. Two agents (A and B) follow predefined paths as indicated in the image. A situation of interest is created by intersecting both paths at the indicated position. A target is provided to Agent A for navigational and attention purposes. A passing car acts as distractor and a reflective surface (window) can provide additional visual information. Image from [SvS22].*

*Vehicle asset based on a design by "storque12" [sto], scene assets by IVC.

(a) (b)

Figure 7.17: A screenshot of the evaluation scenarios including the target object (on the back of the sofa) and a distractor object (on the wall at the top right). (a) While traversing the scene, the agent focuses and perceives the target object. (b) The scene is identical to the scene in (a), but at a certain point in time a distractor is activated, which captures the agent's attention. As a result, the agent cannot perceive the target object.*

*Scene assets based on designs by Unity Technologies [Uni].

(a) (b) (c)

Figure 7.18: Screenshots of the evaluation scene in scenario 3 including both agents and a distractor object (car). In (a) agent A traverses the scene and will have sufficient time to evade agent B coming around the corner. In (b) the scene is identical, except a visual distraction (passing car) will capture agent A's attention. Since the agent attends to the distractor, it is unable to avoid a collision with agent B. Parts of the image are enlarged to show agent A's gaze direction. In (c) the distractor is present again but using the reflective window agent A can perceive agent B in time to avoid the collision. The image part showing agent B's reflection is enlarged for clarity. Images from [SvS22].*
*Assets based on designs by Unity Technologies (agent) [Uni] and "storque12" (car) [sto]. Terrain by IVC.

the proposed perception framework. The most important aspect was to show how an attention model can contribute to changing the outcome of a scenario. For this purpose, the scenario was simulated once with a distractor and once without one. Another fundamental element is occlusion. An agent should only be able to perceive an object if line of sight can be established between the two. Additionally, knowing about occluding objects (and possibly the objects that are behind it) could also be turned into knowledge that higher-level processes can integrate. For example, an agent could decide to change its position to get a better view, it could select actions like peeking around corners, or it could slow down giving it more time to perceive additional information. Demonstrating the combination of different sensor types was another motivation. Here a computationally efficient false-color sensor was combined with a sensor that stored pre-computed saliency information of static virtual objects. Finally, the versatility of the sensing approach was to be shown by including an application-specific sensor that provides an additional source for acquiring stimuli. Therefore, in a third configuration of the scenario, agent A can gain additional information via reflections off a window across the street. Making information from the reflection sensor available to the agent also changed the scenario's outcome. The results from running all three configurations are presented in Figure 7.18.

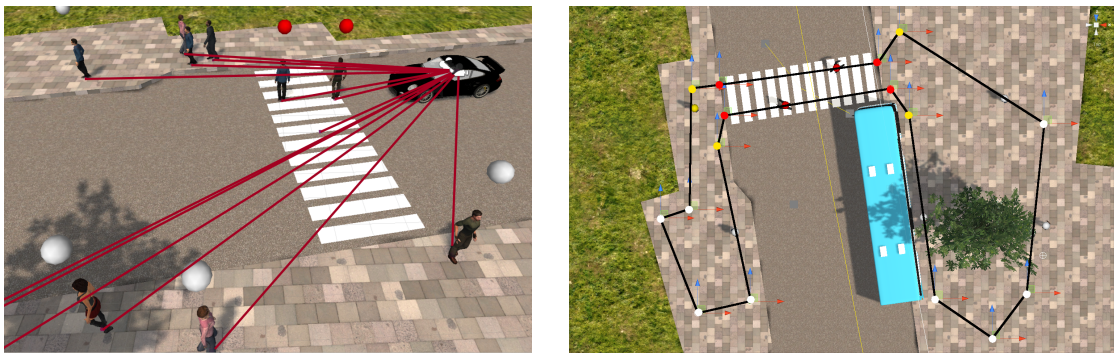
In the default configuration, enough time is available for agent A to focus on, attend to, and evade agent B. The outcome changes once agent A focused on the passing vehicle, which is used as a distractor to pull attention away from agent B (see 7.18 (b)). With its

visual sensors pointed at the vehicle, the agent cannot perceive agent B in time resulting in a collision. By making reflected stimuli available to agent A in the third configuration, the agent becomes aware of agent B before it is distracted by the passing vehicle. This allows the agent to adjust its path to avoid a collision.

7.3.2.3 Results and Discussion

With the presented sample realizations for several components of the described synthetic perception framework, it is possible to recreate the behavior included with each scenario definition. An agent senses its environment using multiple sensors, combines and extracts information from collected stimuli to build its current working memory, and uses additional information and rules to select appropriate actions. In addition to providing stimuli that can be sensed by an agent, the framework also considers the limitations of human sensors. For example, in scenario 3, agent A cannot perceive agent B if there is no line of sight, either due to occlusion or distraction. The latter is demonstrated in scenarios 1, 2, and 3 by introducing a salient object that draws attention away from the current target object. The saliency of the distracting stimulus can either originate from an object (bottom-up attention) or from an agent's higher-level processes (top-down attention). The exemplified stimulus selection based on saliency is combined with dwell time and stimuli inhibition to provide an attention mechanism. Parameters like dwell time or the number of inhibited objects can be adjusted to fit an application's specific needs. In scenario 3 it was also demonstrated how to use the sensor framework to provide a shortcut for higher-level reasoning. By providing a sensor that creates stimuli from reflective surfaces, the process of connecting the effigy on that surface with an object of interest can be efficiently simulated.

The additional cubemap approach is designed to accelerate the perception process, but they could potentially be used to provide multiple saliency values per object on a pixel-based level instead of just object-based saliency. An interesting problem remains the division of static and dynamic objects in the presented approach. Due to this separation it not easily possible to model center-surround features between both object groups. As these features are a central aspect of computational visual attention, the approach may not be sufficient for some applications. However, if performance is a negligible factor, these or similar features could be easily integrated into the presented framework. A further interesting aspect is that saliency probes can be used to encode other information that provides a utility to an agent. For example, in computer games or military simulations, probes could assign saliency based on how well an object is suited as cover. Using sensors in such ways allows tailoring agent behavior to an application.



(a) (b)

Figure 7.19: Perception evaluation scenario including a pedestrian crosswalk. (a): The rays cast from the vehicle (agent) indicate which objects were perceived. Image from [SvS4].* (b): Pedestrians follow the path indicated by the black lines. Waypoints do not only define said path, but also encode how pedestrians are perceived. Pedestrians are perceived as *crossing the road* whenever they are between red waypoints or as *approaching the crosswalk* when they are between a yellow and a red waypoint. Image from [IVC8].*

*Assets based on designs by Unity Technologies (pedestrians) [Uni] and "storque12" (car) [sto]. Terrain and bus assets by IVC.

7.3.3 Evaluating the Application of the Synthetic Perception Approach to Traffic Scenarios

The final evaluation demonstrates the application of the perception framework to the intended target domain by defining a typical traffic situation. A further objective was to investigate the limitations of the realized system. The result are presented in this section, but more details can be found in [SvS4, SvS9], and [IVC8].

7.3.3.1 Crosswalk Scenario Setup

The evaluation scenario includes a cognitive agent driving a car down a specified road section with a speed limit of 50 km/h. After a certain distance, a traffic sign reduces the speed limit to 30 km/h. Shortly thereafter, the agent encounters a crosswalk that is sporadically used by pedestrians to cross the road. Figure 7.19 shows the section of the scenario that includes the crosswalk. To properly pass the scenario, the agent must perceive and react to the traffic sign, subsequently reducing its velocity. After perceiving the crosswalk, the agent must decide whether it can pass it without endangering any of the pedestrians. To achieve these behaviors, two mechanisms were realized for this evaluation scenario that allow the agent to react to perceived information and select appropriate actions. These mechanisms pull information from the agent's STM in regular intervals. In Chapter 5 this is defined as a

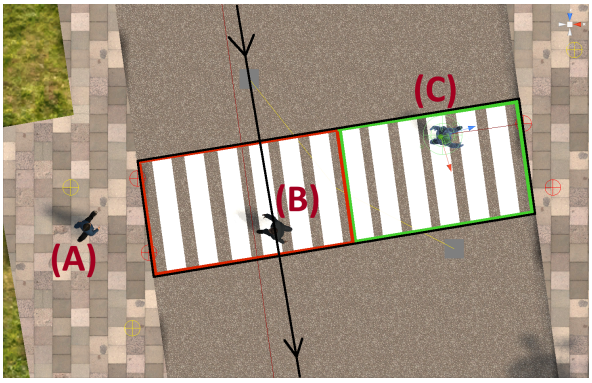


Figure 7.20: Visualization of the decision process for the driving agent. The agent approaches the crosswalk from above (indicated by black arrows). Pedestrian (A) is critical because it is currently navigating the part of the crosswalk that is passed by the agent in its vehicle. Pedestrian (B) is also critical because it is approaching and soon entering the crosswalk. Pedestrian (C) is not critical even though it is navigating the crosswalk. However, the pedestrian has already crossed the agent's driving lane and is continuing to cross the road on the other lane. Image from [IVC8].*
*Pedestrian assets based on designs by Unity Technologies [Uni]. Terrain assets by IVC.

synchronous request by a high-level process. One mechanism observes the memory module for speed limit signs and whenever a new sign is detected, the agent adapts its desired velocity to the speed limit associated with the sign. The other mechanism checks the STM for crosswalks, after perceiving one, all nearby pedestrians are queried from STM. If one of the perceived pedestrians is approaching the crosswalk or currently crossing the road, the agent stops at the crosswalk. Figure 7.20 demonstrates the processing of pedestrians and their significance towards the agent's decision.

7.3.3.2 Limitations of the Applied Perception System

To evaluate the limits of the perception system, the scenario was modified by various factors, which are described in the following sections.

Velocity Variation

The first modification was the variation of velocities for all involved entities, i.e., the agent and the pedestrians. The default velocity for pedestrians was between 0.7 m/s and 1.1 m/s. At velocities greater or equal to 3 m/s, the agent regularly collides with a pedestrian. The reason is how the perception of critical pedestrians is set up in this sample scenario. The agent perceives pedestrians as *approaching the crosswalk* only after they reach a certain waypoint. If the pedestrians are moving too fast, the time span they need to start crossing the road after switching to the approaching state is too short for the agent to stop in time resulting in a collision. Decreasing the pedestrians' velocities results in increased waiting times for

the agent for the same reason. Pedestrians change their state to *approaching*, but because they move so slowly, the agent spends more time waiting at the crosswalk, although it may have had the opportunity to pass the crosswalk before a pedestrian entered it. To avoid both collisions and increased waiting times, the time of changing the state to *approaching* would have to be based on a pedestrian's velocity at the time. Since all pedestrians walk at different velocities, this state change would have to be within the agent's memory and performed for each perceived pedestrian. A mechanism for *predictive coding* could be a solution for the agent to make an *a priori* estimation of the point in time at which a pedestrian requires the agent to stop.

To change the velocity of the agent, the speed limit sign was changed to a different velocity. Despite high velocities of more than 100 km/h, the agent can perceive pedestrians early enough to decelerate in time to stop in front of the crosswalk.

Visual Occlusion by an Obstacle

An additional modification is an object that occludes certain areas of the scenario to investigate the resulting agent behavior. Since occlusion is already part of the realized concept, no further modifications are necessary other than placing additional objects. At first, a tree was placed to occlude the traffic sign. As expected, the occlusion resulted in the sign being perceived late or not at all, causing a delayed or absent adaption of the velocity. However, even at high velocities the agent can stop in front of the crosswalk if pedestrians are crossing the road because the pedestrians are not occluded and are perceived by the agent with enough time to stop.

A second alteration is a bus, which is stopped at the side of the road occluding the approach to the crosswalk. Additionally, a stationary pedestrian is placed within this approach area. As soon as the pedestrian is perceived, it is classified as critical, which prompts the agent to stop. Due to the setup, the pedestrian is perceived shortly before the agent reaches the crosswalk. Expectedly, whether the agent can stop in time depends on its velocity and when a line of sight can be established to the pedestrian.

While the observed results were expected, one particular aspect is not considered. A human driver would realize that the bus possibly occludes pedestrians that want to cross the road. In response, a careful driver would reduce its speed to have more time to react to emerging pedestrians. Consequently, an agent should also perceive that the approach to the crosswalk is occluded and reduce its velocity. To achieve this behavior, agents would have to be able to realize that certain areas are not perceivable. So far, the perception concept does not consider this case. A possible solution within the current concept could be to model the crosswalk approach areas as perceivable objects. If there is no line of sight to these

Figure 7.21: Visualization of the crosswalk evaluation scenario with visual range reduced to 50 m by fog. The crosswalk and pedestrians are visible in the background. Image from [IVC8].*
*Assets based on designs by Unity Technologies (pedestrians) [Uni] and "storque12" (car) [sto]. Terrain and bus assets by IVC.

objects, the agent could react accordingly. However, in this case, the mass point approach for raycast-based sensors, described in Chapter 5, would have to be altered. Instead of one mass point, all mass points must not be visible to ensure that the area is indeed not visible.

Limiting Visual Range

The range of the human visual system can be altered by many factors. One aspect are environmental influences, which can be introduced by changing visual representations within the virtual scene or by altering world semantics (see Section 5.7). For visual perception, these changes concern mostly visibility and lighting conditions, e.g., rain, snow, fog, or darkness.

In the evaluation scenario, a parameter to simulate fog was added to the scene's world semantics. The parameter was used to reduce the visual range to a certain distance around an agent based on the thickness of the fog. Objects beyond this range are not sensed by an agent. Once the distance to an object is below this range, the object is regularly sensed, perceived, and considered for action selection by an agent. Reducing visual range by fog, the perception of relevant situations within the crosswalk scenario is delayed, which can cause sudden deceleration or the agent to miss an object. Figure 7.21 depicts a section of the evaluation scenario in which visual range is reduced by fog to 50 m. The representation in world semantics was combined with a rendering effect to visualize the limitation.

Using the realized perception system, limitation of the visual range can be simulated by

environmental and individual effects. By adjusting the appropriate parameters, the outcome of the scenario can be influenced. The agent either stopped in time for crossing pedestrians or it collided with pedestrians if perception occurred too late or not at all.

7.3.4 Summary

During the evaluation of the perception framework, the aspects of plausibility, real-time capability, and controllability were considered, which are the content of **RT3.2**. Specifically, run time performance of example sensor implementations was measured to evaluate the relationship between accuracy and performance. In multiple evaluation scenarios several functionalities of the framework were demonstrated and a set of implementation examples for sensors, memory, and attention was provided. Furthermore, it was shown how sensors can be used to provide additional information to an agent using semantics and scene knowledge. Since the sample implementations were performed using a real-time game engine, one possible solution to **RT4.1** is implied.

A crosswalk scenario was used as an additional evaluation to show that the system can simulate several perception processes in a real-time road traffic application (**RT4.2**). An agent driving a car can perceive and react to a traffic sign, a crosswalk, and pedestrians crossing the street using the crosswalk. To observe different outcomes of the defined scenario, only the perception process was modified while the decision process remained unaltered. While modifying the scenario, observed outcomes matched expectations. The range of possible velocity variations mainly depends on the realization of decision processes. Occlusion can be simulated, but the precision depends on the accuracy of provided sensors. While it was also shown that visual range can be limited within the proposed system, the range of possible limitations could be explored in more depth.

Assuming a preconfigured set of sensors, application developers do not have to be concerned with the process of acquiring information from the virtual environment. The perception module accumulates the "correct" data and makes it available to high-level processes. If and when an agent perceives certain objects is based on the abilities and limitations of its sensors. Developers can focus on realizing decision mechanisms appropriate to a given application, which simplifies the development process. Due to a defined interface, additional sensors can be integrated to cover specific needs.

In general, the evaluations also showed how knowledge about the environment can be efficiently provided to agents (see **RT3.2**). For example, visual range limitations by world semantics or relevance of perceived pedestrians to the decision process. In these cases, *efficiently* refers to the fact that information can be perceived and must not be deduced from available sensor information by each agent. However, at the same time, efficiency can refer to

the development process, as setting up a scenario and changing its outcome can be achieved by simple decision processes and without any additions to the perception process.

7.4 Evaluating Scalability

The intention of the mesoscopic queuing model is to allow the implementation of interactive experiences while keeping invisible agents in the system (see Section 6.3.1). To verify that the concept fulfills this intention, also as valid solution to **RT4.2**, three separate performance evaluations are conducted. First, the mesoscopic system is investigated in isolation to evaluate how well the system scales (cf. [SvS3, SvS26], and [IVC5]). However, since a mesoscopic simulation layer is only meant as a support for the microscopic simulation, three combinations of both cognitive agents and mesoscopic agents are simulated within the same virtual environment (cf. [SvS9]). Finally, the level-of-detail approach explained in Section 6.3 is evaluated by combining cognitive agents, simplified agents, and mesoscopic agents in a single scenario (cf. [SvS9]). As mentioned in Chapter 6, the FIVIS bicycle simulator is used to facilitate a realization of the concept. Therefore, the scenario implemented within the FIVIS project is utilized for the evaluations explained here. The following sections provide more detail about the three evaluations.

7.4.1 Evaluating the Scalability of the Mesoscopic Simulation System

To investigate the performance of the realization of the mesoscopic simulation concept, a test scenario is created, which is based on a scalable traffic network. The targeted framerate for the simulations was 30 frames per second (FPS), which is a typical benchmark for interactive experiences. To match this target frame rate, calculating a frame can take no more than 33.33 ms/f. Besides the framerate benchmark, the time required to calculate one frame of the simulation is expected to increase linearly for both network size and agent count. Additionally, to allow a more generalized usage, e.g., in open world virtual environments, the system should be able to handle reasonable network sizes. For example, the game Marvel's Spider-Man[6] included the entire road traffic network of Manhattan, NYC containing ca. 4500 intersections.

The traffic network inside the simulation can have any form, but for the evaluation, it was limited to a $n \times n$ grid layout (see Figure 7.22). Each node is connected by a road with its four immediate neighbors with the exceptions of the intersections at the fringes and corners, which have three or two neighbors, respectively. Since each road consists of two

[6]Sony Interactive Entertainment, 2018

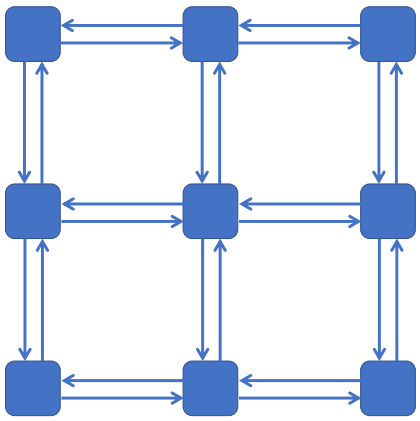


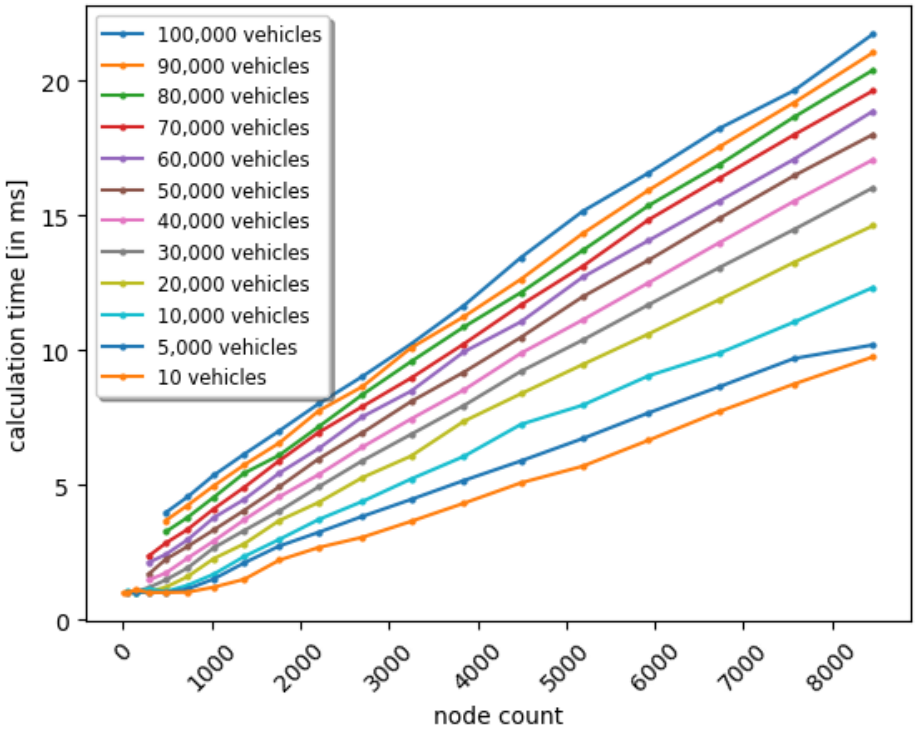
Figure 7.22: Depiction of a generated $n \times n$ traffic network of 9 nodes and 24 edges for evaluating scalability of the mesoscopic traffic simulation. Image based on [SvS3].

lanes (edges), the number of edges in each evaluation network is given by:

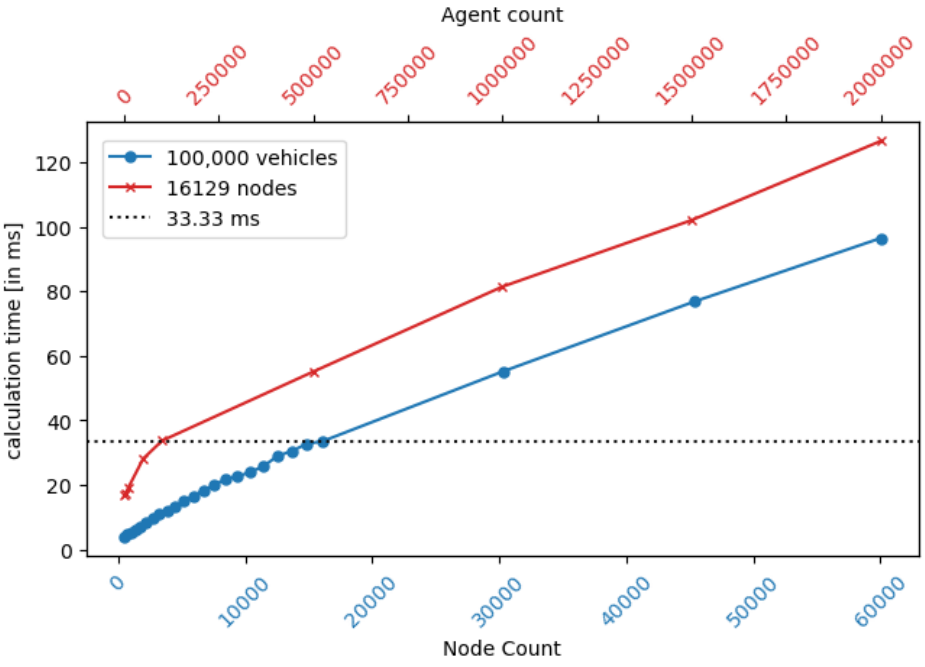
$$2n(n-1) + 2n(n-1) = 4(n^2 - n) \tag{7.3}$$

All edges are assigned a speed limit of 50 km/h and defined to be 500 m in length. Assuming an average vehicle length of 7.5 m, to account for different vehicle types, each edge has a maximum capacity of 66 vehicles. In the mesoscopic queuing model, calculations are performed for each network element and each simulated agent. Therefore, varying numbers of mesoscopic agents are simulated within increasingly large networks. Starting at a network grid of size $n = 2$ (4 nodes), the agent population is systematically increased from 10 to 100000 agents. The maximum number of agents that can be simulated for a given network is thereby determined by the capacity and number of edges. If an agent population does not fit a network size, the configuration is not simulated and no data is acquired. For example, for the 2×2 grid, the included $4 \cdot (2^2 - 2) = 8$ edges can be populated by a maximum of $8 \cdot 66 = 528$ agents and all larger populations are skipped for this network. To simulate, e.g., 20000 agents, a network must have at least 360 edges ($n = 12$). After simulating the maximum agent population, n is increased in increments of 5 and the sequence of agent counts is repeated until $n = 92$ (8464 nodes, 33488 edges). Each simulation is run for 120 seconds to collect a sufficient amount of performance data, which is logged every second. All simulations were run on a standard PC (Intel® i7-4790 3.6GHz CPU, 16GB RAM, NVIDIA GeForce GTX 780 Ti).

Each graph in Figure 7.23 (a) shows the effect of network size on the average calculation times for a constant number of agents. After initial irregularities, all graphs show linear growth with increasing traffic network size, as was expected when setting up the experiments. However, as indicated by the graphs, increasing the agent population size dominates



(a)



(b)

Figure 7.23: Median frame calculation times for the mesoscopic agent evaluation. (a): The median calculation times for different agent counts in traffic networks of increasing size. The graphs show that overall, the calculation times grow linearly, and all configurations can be simulated interactively. (b): Median calculation times for 100000 agents in networks of increasing size (blue line) and an increasing number of agents in a network of 16129 nodes (red line). The graph shows that calculation times grow linearly, although for smaller agent counts the slope is steeper. Images from [SvS26].

the increase in calculation times until a threshold is reached. After reaching the threshold the calculation times increase linearly. This effect can also be observed for the node graph in Figure 7.23 (b) and would have to be considered when designing a specific scenario.

To observe the growth behavior for traffic networks of even larger sizes, simulations were run with 100000 agents and a traffic network size of 30276 nodes, 45369 nodes, and 60025 nodes, respectively (n = 174, 213, and 245). As demonstrated in Fig. 7.23 (b), even for very large traffic networks, the calculation time increases linearly. To investigate the development of the calculation time with increasing numbers of agents, additional simulations were performed using a network size of 16129 nodes (n = 127) and larger agent counts. Again, after a certain initial agent count, a linear growth of the calculation time can be observed (see Figure 7.23 (b)).

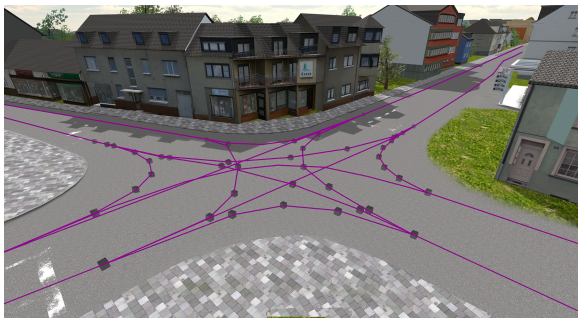
As shown in Figure 7.23 (a), all configurations up to 8000 nodes and up to 100000 agents can be simulated below the targeted calculation times. These results are sufficient for the system's original application within FIVIS. However, for larger networks and agent counts the target framerates cannot be achieved at certain points depending on the configuration. In the case of 100000 simulated agents, that threshold is reached at about 15000 nodes (see Figure 7.23 (b)). Thus, a city the size of Cologne, Germany (13206 nodes) could be simulated, but a metropolis like New York City, USA (51506 nodes) would be too large for interactive experiences when simulating persistent agents using the described system[7]. Additional results regarding the mesoscopic simulation can be found in Appendix B.

7.4.2 Evaluating the Combination of Microscopic and Mesoscopic Systems

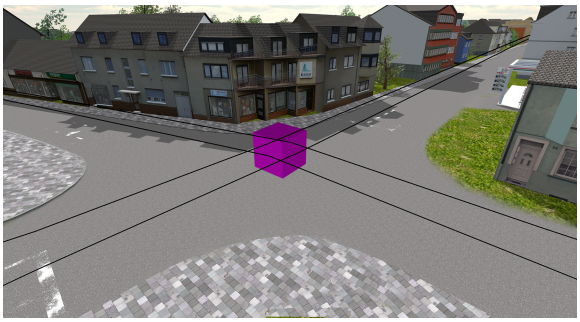
In the previous section, it was shown that 100000 mesoscopic agents can be simulated at interactive framerates in a traffic network of more than 8000 intersections. However, the queuing model is meant only as a support for applying cognitive agents to interactive experiences. Therefore, a second evaluation combines cognitive agents in a microscopic simulation with the mesoscopic queuing model. The environment used for the evaluation is the same as the main scene within the FIVIS bicycle simulator project. The scene represents a section of the city of Siegburg, Germany, with an area of approximately 0.32 km^2. The total length of the road network within the scene is about 10 km, consisting of mostly secondary roads and one main road, which are connected by 24 intersections. The entire road network consists of about 500 waypoints, which are used by the microscopic simulation (cf. Section 6.2). In comparison, the mesoscopic simulation graph consists of 24 nodes connected by 64 edges. A screen capture of the scene as well as a representation of the simulation elements

[7]Road network sizes were determined using the OSMnx [Boe17] by extracting the number of intersections from OpenStreetMap data (`https://www.openstreetmap.org/`).

(a)



(b)



(c)

Figure 7.24: Screenshot of one intersection in the Siegburg city scene used within the FIVIS bicycle simulator (a)* and its road network elements for the microscopic (b)** and the mesoscopic simulations (c)**. For the microscopic simulation, 45 waypoints represent paths across the intersection. In the mesoscopic simulation graph, the intersection is represented by one node, which is connected to its neighbors by one edge per lane. Images based on [SvS3].

*Assets based on designs by "Underground Lab" [Und] (&), "3DJunior" [3DJ] ($), and IVC (§ and terrain).

**Terrain assets by IVC.

are shown for one of the larger intersections in Figure 7.24. Since the average vehicle length is estimated at 7.5 m, the maximum capacity for the road network is 1333 vehicles. Due to the lack of available traffic data for the simulated environment, data provided by the United Kingdom's Department of Transport[8] is used as a realistic initial guess for populating the scene. According to this data, a usage of 40% of the maximum capacity represents rush hour traffic, which means 500 agents for the evaluation scenario. Since congested roads during a rush hour scenario are typically not interesting for an application like FIVIS, 200 agents are estimated as a suitable number for plausible traffic flows.

To acquire referential data, the scene is simulated without any agents (A). The resulting measurements represent the fastest possible frame times when simulating the Siegburg scene in Fivis. The majority of computational effort is spent on rendering but Unity's entire engine framework and several Fivis components, e.g., responsible for input monitoring, logging, screen definition, and more, also require processing resources. For a direct comparison, the scenario is simulated once with 200 microscopic agents (B) and once with 200 mesoscopic agents (C). The fourth configuration simulates the estimated maximum of 500 agents using the mesoscopic approach (D). Finally, a hybrid simulation using 20 microscopic agents and either 180 (E) or 480 mesoscopic agents (F) is investigated. Twenty agents are estimated to be visible on average to a user based on the scene population as well as the size of the environment and the traffic network. All configurations are simulated for 10 minutes and run fifteen times to compensate for background processes running on the same machine. To simulate a user navigating the scene, the scene's camera is set up to follow one of the simulated agents. The targeted frame rate is again 30 FPS, i.e., 33.33 ms/f.

Figure 7.25 summarizes the results for all six configurations. The reference scenario is simulated without any agents and is equal for all configurations, i.e., frames cannot be calculated faster than 2.9 *ms* on average, which is equivalent to 345 FPS. While simulating configuration B, the average and median calculation time of 333.33 *ms* per frame (3 FPS) is clearly above the targeted time of 33.33 *ms*. Therefore, simulating 200 microscopic agents is not an option for interactive systems. In comparison, simulating the same number of mesoscopic agents, average and median calculation times are equivalent to the reference scenario (2.9 *ms*). This result is expected since mesoscopic agents require neither a visual representation that must be rendered, physics components that must be simulated, nor a simulation of cognitive functions and interactions. Calculation times also do not change when simulating the estimated maximum of 500 mesoscopic agents within the FIVIS scene. Considering the results reported in the previous section, these result meet expectations. Finally, both hybrid configurations produce significantly longer calculation times than con-

[8] `https://www.gov.uk/government/organisations/department-for-transport`, [online: May 2, 2023]

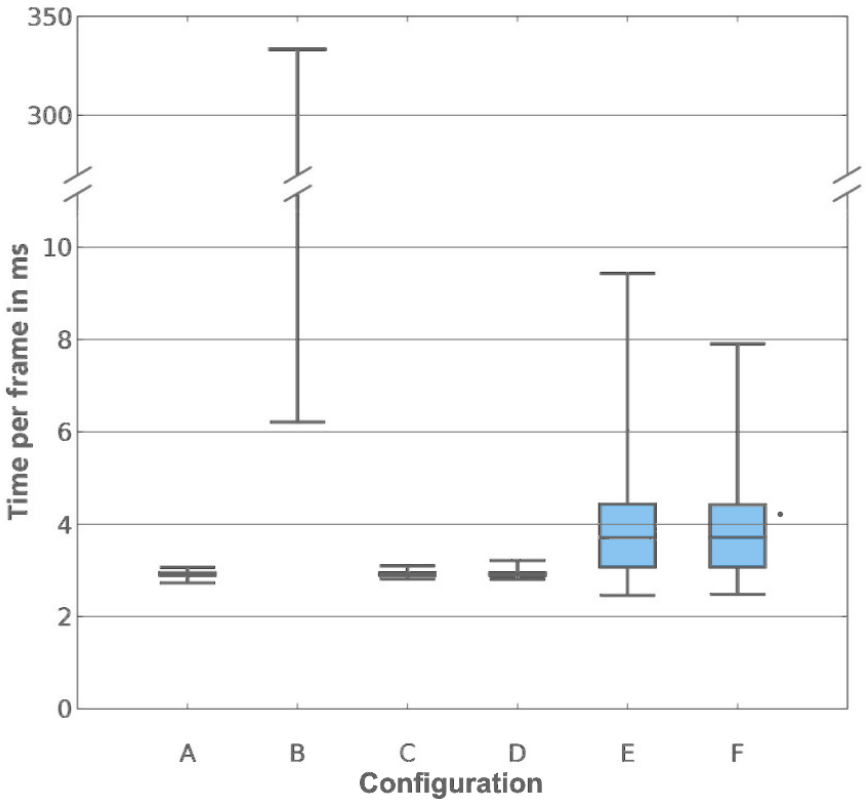


Figure 7.25: Calculation times for individual frames in milliseconds from different test scenarios. A: Reference scenario without agents (median: 2.9 *ms*). B: Simulation of 200 microscopic agents (median: 333.33 *ms*). C: Simulation of 200 mesoscopic agents (median: 2.9 *ms*). D: Simulation of 500 mesoscopic agents (median: 2.9 *ms*). E: Simulation of 20 microscopic and 180 mesoscopic agents (median: 3.69 *ms*). F: Simulation of 20 microscopic and 480 mesoscopic agents (median: 3.71 *ms*). Image based on [SvS9].

figurations A, C, and D, which also show more fluctuation. Depending on the distribution of the microscopic agents throughout the road network, the amount of required computational resources varies, which explains the fluctuations. However, both median values are essentially equal (E: 3.69 *ms*, F: 3.71 *ms*), showing that the mesoscopic simulation is negligible for the considered road network and agent population. Using a hybrid simulation and an estimate for the number of visible microscopic agents, interactive frame rates were achieved.

7.4.3 Evaluating the Level-of-detail Approach to Simulation

After evaluating the isolated effects of mesoscopic and microscopic simulations on framerates, the level-of-detail approach introduced in Section 6.3 is evaluated. For this purpose,

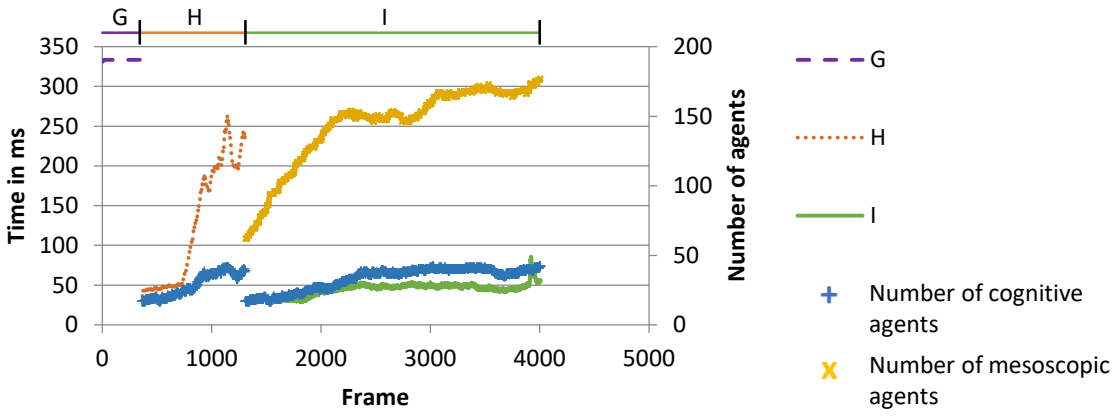


Figure 7.26: Calculation times for individual frames in milliseconds and the number of agents simulated in each simulation layer: Only cognitive agents (G), cognitive and simplified agents (H), and all levels of detail, i.e., microscopic, simplified, and mesoscopic agents (I). Cognitive agents are microscopic agents simulated at the highest level of detail. The number of simplified agents is omitted since it can be deduced from the given values. Image from [SvS9].

three additional configurations are simulated and the calculation times of individual frames are recorded. The Siegburg scene is used for this evaluation as well and the camera is statically placed at a busy intersection. The total number of simulated agents is always 200. In configuration G, 200 microscopic (cognitive) agents are simulated at the maximum level of detail and layer transfers were disabled. In configuration H, agents are transferred to the simplified layer as soon as they leave the vicinity around the camera. Finally, transfers between all three layers are simulated in configuration I. Agents within the camera's vicinity are simulated at the maximum level of detail, agents leaving the vicinity are simulated in a simplified manner, but visually represented, and agents not visible from the camera's position are simulated by the macroscopic layer. Figure 7.26 shows the calculation times and the number of simulated cognitive and mesoscopic agents for one representative test run. The distribution of the measured frame times for the same test run are depicted in Figure 7.27.

Equivalent to the previous evaluation, it is not possible to achieve interactive framerates using only the microscopic layer. Frames continuously require 333.33 *ms* to be calculated in configuration G. The distribution of agents at the beginning of a simulation is determined once and used for all simulations. Due to this distribution, 17 cognitive agents are situated within the camera's vicinity at the beginning of the sample scenario. The remaining 183 agents are simulated in simplified form (i.e., without physics and cognitive processes). This measure already reduces calculation time at the beginning of the simulation to 36 *ms*, which almost meets the target time of 33.33 *ms*. In the presented test run, the camera is placed at a large intersection regulated by traffic lights. For this reason, the number of

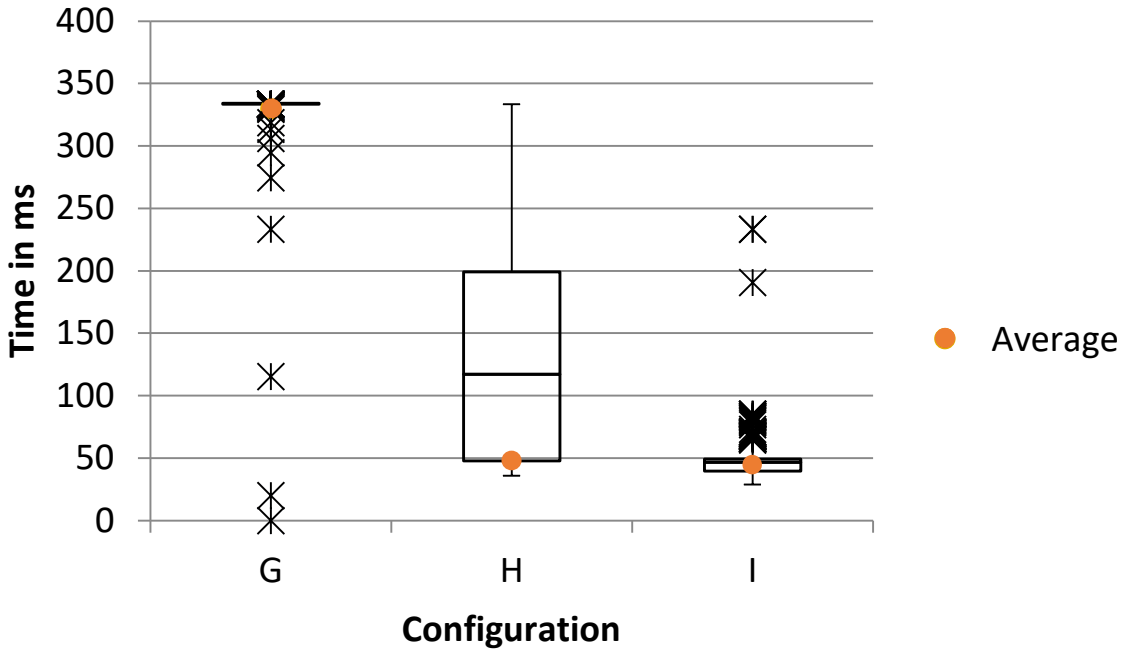


Figure 7.27: Distribution of calculation times for the test run from Figure 7.26. Image from [SvS9].

microscopic agents increases to 41 during the simulation. Figure 7.26 indicates how the increase in microscopic agents directly influences calculation times. When considering the median of 117.3 *ms* in configuration H, the calculation times decreases by 65% compared to configuration G. The average time of 48.5 *ms* per frame could already be considered an interactive simulation, although the targeted time is still not achieved. Again, 17 cognitive agents are simulated at the beginning of the simulation. A total of 138 microscopic agents (17 cognitive agents, 121 simplified agents) are simulated next to 62 mesoscopic agents resulting in an additional decrease of the calculation time to 29 *ms* for the first frame. The median time is 47 *ms* and the average is 44 *ms* for the entire simulation. While the targeted times can still not be achieved, Figure 7.26 shows that an increasing number of cognitive agents influences calculation times less and that calculation times vary less than in previous configurations.

Finally, to investigate the performance of the realized system in a less challenging scenario, all three configurations are run with the camera placed at a smaller intersection with less traffic. For this situation, it is possible to achieve calculation times below the target value of 33.33 *ms*. Figure 7.28 shows the results for this simulation. The average time for configuration I is 19.8 *ms* and the median time is 20.24 *ms*.

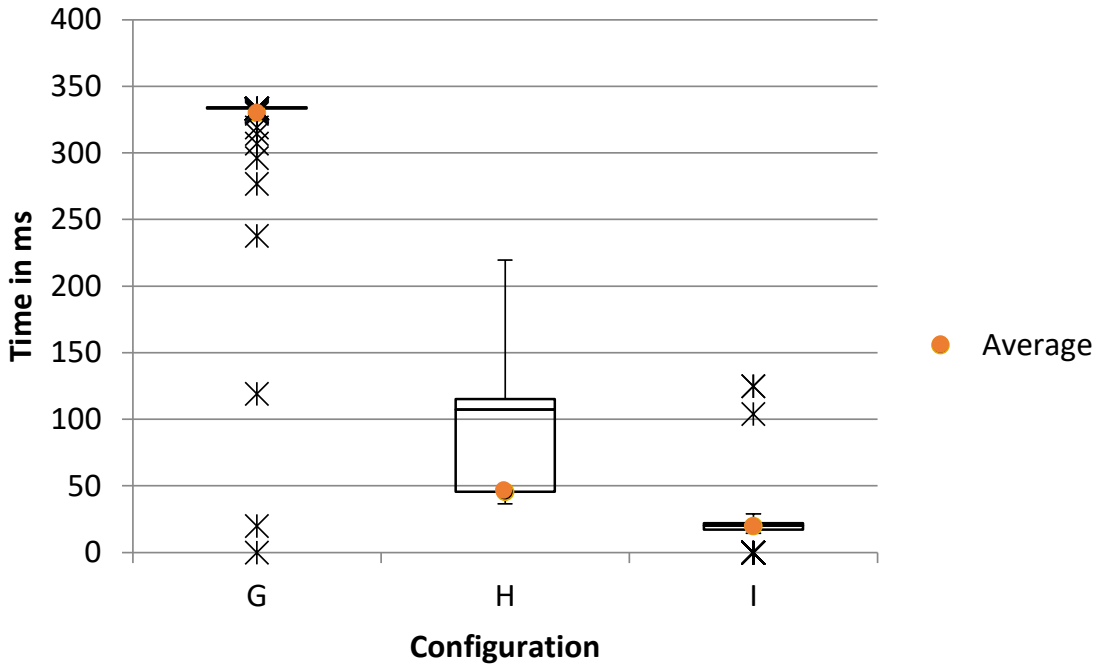


Figure 7.28: Distribution of calculation times from a test run with the scene camera situated at a less frequented intersection. Combining all level of detail layers (I), calculation times below 33.33 *ms* (30 FPS) could be achieved. Image from [SvS9].

7.4.4 Summary

While realizing the cognitive traffic agent concept, it was predictable that achieving interactive framerates would not be possible when simulating enough agents for the designated scenario. The main reasons being the computational demands of all cognitive processes, the visual representation, and the physics simulation. To counteract this issue, Section 6.3 presented a level-of-detail approach consisting of multiple simulation layers, one of which simulates traffic agents mesoscopically in a queue-based system. To investigate the effect of the mesoscopic simulation, its overhead and scalability was evaluated using an artificial road network graph. Graph size and agent population were successively increased, and computational costs were recorded for each frame. The simulations show that computational resources increase linearly with graph size and agent population. Furthermore, 100000 agents can be simulated within an 8000-node network below the target of 33.33 ms/f.

Both agent population and network size are far beyond what is necessary for the intended target application The results also show that the system could theoretically be applied to current open world environments, like the one in Marvel's Spider-Man. The size of the road

traffic network of New York City's borough Manhattan (4493 nodes), which is the setting for the game, is well below the 15000-node threshold for interactive framerates. However, the navigation graph used in the open world of ancient Egypt in the game Assassin's Creed: Origin[9] consisted of 480000 nodes[10] [Lef18]. Interactively simulating a network of this magnitude would not be possible using the described system without optimization. Additionally, in a digital game, this mesoscopic simulation would only be one part of a much larger system. Thus, it would have to operate within a certain frame budget to leave enough resources for other sub-systems (rendering, physics, animation, behavior, etc.). In this case, the maximum size of an interactive traffic network would be significantly smaller, depending on the actual frame budget. Considering these constraints, it would be challenging to apply the mesoscopic agent system to current open world scenarios without additional performance improvements. However, due to the linear growth in calculation times, it is reasonable to assume that optimizing the system could resolve this situation.

Additionally, it was shown that the presented system is not capable of simulating very large traffic networks, like those required for a metropolis or a small country (e.g., Switzerland with 203742 nodes). However, since the system was not originally developed for traffic networks of these dimensions in mind, there are many opportunities for improvement. Further work on optimization could decrease the growth and therefore might allow larger networks to be interactively simulated. For example, as indicated by others (e.g., [SCA19]), the sequential realization would lend itself well to parallelization.

To further evaluate all layers of the level-of-detail system in combination, a virtual representation of a part of the city of Siegburg was used, which is the main scenario for the FIVIS bicycle simulator. Multiple configurations were run using this scene, including different combinations of microscopic (cognitive and simplified agents) and mesoscopic agents. The evaluation confirmed the anticipated drop below interactive framerates when simulating only microscopic agents. The evaluation showed that the mesoscopic layer introduces negligible overhead and confirmed the anticipation that the simulation of microscopic agents is the major factor for the computational requirements. Despite the utilization of the level-of-detail approach, the targeted median time per frame of 33.33 *ms* (30 FPS) could not be achieved for the application scenario. However, the recorded time of 47 *ms* is close to interactive framerates and the maximum time as well as variance could be significantly decreased compared to configurations without mesoscopic agents. Additional simulations for a less demanding setup showed that it is possible to achieve interactive framerates by using the level-of-detail approach. To ensure interactivity for a broader set of cases, additional opti-

[9] Ubisoft, 2017

[10] Note that Assassin's Creed: Origin's navigation graph consisted of 520000 links, which is significantly less than the 1.9 million edges that would have been contained in a test network grid of 480000 ($n = 692$) nodes.

mizations would be beneficial. For example, the processing of individual cognitive agents could be distributed across multiple frames or the user's vicinity, in which agents are simulated at the maximum level of detail, could be dynamically resized to keep the number of cognitive agents at a constant value. Realizing these additional optimizations was not within the scope of this thesis project.

The intent of **RQ4** was to evaluate whether the developed concepts could be realized as an interactive application. Realizing the cognitive agent concept by itself, did not provide the desired result within the designated application. However, by extending the concept with a level-of-detail solution, an interactive simulation could be achieved without compromising the intended effect of cognitive agents on a user or trainee. Note that other applications, requiring only a small number of cognitive agents, could achieve interactive framerates without level-of-detail adjustments. This assumption has not been evaluated but could be the subject of a future project.

8

Conclusions and Discussion

"Every accomplishment starts with the decision to try."

- John F. Kennedy

Intelligent agents can be a valuable asset for virtual training environments. To facilitate user immersion, agents must portray believable and individualized behavior, which must also be controllable for an application designer to support the training objectives. At the same time, a solution for generating agent behavior, must be scalable to consider real-time constraints of virtual environments. So far, existing agent architectures do not meet all requirements simultaneously, in particular regarding the use cases envisioned within this thesis project. To address this issue, four major research contributions are provided. A general design of a lightweight cognitive agent architecture has been developed (**RQ1**). Within this architecture design, two specific aspects were highlighted. First, the combination of a trait-based personality model and an emotional state represents a component to systematically generate individual, dynamic agent behavior (**RQ2**). Second, a perception and attention module has been designed and integrated to demonstrate its role in the behavior generation process for cognitive agents (**RQ3**). Finally, the scalability of the proposed architecture has been shown by applying it to a real-time, multi-agent system (**RQ4**).

The approach to achieving believable agent behavior in this thesis project was to emulate human cognitive capabilities and characteristics constrained by real-time requirements. A lightweight cognitive agent architecture concept has been developed, evaluated, and deployed. The concept includes multiple identified key features, such as perception, decision-making, memory, internal simulation, learning, emotion, mood, and feelings. This concept represents a methodological framework for solving the proposed research questions, which integrates the mentioned cognitive capabilities. Due to the complexity of every individual

component and the constraints of the intended application domain, the refined concept focuses on four core elements: perception (including memory and attention), personality, emotion, and decision-making. The resulting CA^{2}RVE architecture defines a blueprint for applying attentive cognitive agents to real-time virtual environments (**RT1**).

The generation of individual behavior patterns has been realized by integrating psychological personality profiles as a foundation for all cognitive processes (**RT2.1**) and evaluated. It has been shown that personality profiles can be individualized, such that every agent in a simulation is assigned a unique profile. While it was shown that distinguishable and consistent behavior can be generated by linking task specific behavior to personality prototypes, variety and quality is limited by multiple factors, e.g., the number of prototypes or the variety of actions available to an agent. Since providing a diverse set of actions requires substantial effort and resources, the focus was on realizing actions required for the evaluation scenarios, e.g., *wait*, *yield*, and *start driving* for a blocked-lane scenario.

Linking individualized personality to actions has been achieved using task parameters, which are applied to model cognitive capabilities and filter actions. Thus, depending on an agent's personality, certain actions may be prioritized. It has been demonstrated how results from a psychological study about driving behavior are transferred to the agent architecture by means of a developed mapping process. Additionally, since the mapping process is generalized, it is possible to transfer results of other psychological studies to agent behavior as well. This option presents interesting opportunities for further research and applications. For example, results of psychological studies may be validated or re-evaluated in virtual settings using the proposed agent architecture.

Extending the concept of individualized behavior, a two-dimensional emotion model has been realized to achieve behavior pattern alteration (**RT2.2**). Using this approach, it has been shown that a problematic situation can be resolved (e.g., the exemplified deadlock at an intersection). By enabling agents to react to their circumstances, it has been demonstrated that emotions can improve the plausibility of observable agent behavior.

Based on human perception, a synthetic perception framework has been designed that contains three key aspects: sensing, memory, and attention (**RT3.1**). The design has been realized and evaluated using specific perception-based scenarios. In these scenarios, it has been shown that the saliency-based interface allows a flexible combination of different sensor modalities. Due to this flexibility, users of the architecture can prioritize either performance, robustness, or accuracy of the perception process (**RT3.2**). The ability to customize the solution allows a wide variety of potential applications, e.g., training environments, digital games, perception research, education, and simulation. The sample realization includes configurable memory and attention mechanisms to provide additional controllability of

generated behavior. The developed attention module was used to demonstrate the intended filtering of stimuli in short-term sensory storage based on saliency. In addition, the modular design allows integrating other attention approaches as well.

The perception module has been realized such that acquisition, storage, and availability of information is transparent to high-level processes, like decision-making. This approach enables the design of processes that obtain information from the perception module without requiring detailed knowledge of the internals of the system. Developers using the framework can simply query the perception module and assume that the currently available information always complies with the agent's perception capabilities. This frees developers from managing the acquisition of information from the virtual environment themselves.

Due to the way sensor integration is designed, multiple modalities can be considered for stimuli acquisition. Within this thesis project, visual sensors were investigated, and the integration of an auditory sensor was demonstrated as well. Furthermore, two examples for application-specific sensors were presented, in which the perception framework is used to efficiently provide agents with knowledge about their environment (**RT4.1**). One example is the integration of objects visible on reflective surfaces. The other example is the perception of road network elements. By implementing application-specific sensors, valuable information becomes available to agents without the need for complex computations, which improves run-time performance. At the same time, agents can choose to utilize this information individually, which allows managing the perception process' plausibility.

The proof-of-concept aspect of this thesis project consisted of applying the developed concepts to real-time, multi-agent systems demonstrating the architecture's scalability (**RT4.2**). Parts of this aspect have been addressed above regarding **RT4.1**, i.e., the ability to focus on performance when configuring sensors. Furthermore, a solution to **RT4.2** has been demonstrated by providing a realization of the concepts as an interactive, virtual user experience. CA^2RVE agents were utilized as traffic participants within the FIVIS bicycle simulator context. Using consumer hardware, up to 20 traffic agents can be simulated in real-time, i.e., keeping framerates above 30 FPS, without optimizations. While these numbers fulfill the requirement of a real-time, multi-agent environment, especially the small agent population degrades the credibility in the considered scenarios and limits the variety of imaginable applications. To counteract this problem, a level-of-detail system has been designed and realized to balance performance load against agent population size. The solution included dealing with application-related issues like visibility constraints and unnoticeable transfers between level-of-detail layers. Using these techniques, 200 agents were simulated with 50 FPS for a representative traffic scene. Further tests indicated that rendering and physics

exert more influence on performance than the agent system as a constant number of 20 microscopic and 480 mesoscopic agents were simulated at about 270 FPS.

In conclusion, solutions to all investigated research subtasks were presented and the contributions of this work can be summarized as follows. A methodological contribution was put forth by devising a new lightweight cognitive agent architecture considering the scientific gap regarding the simultaneous coverage of believability, controllability, and scalability. The cognitive agent architecture concept from the conceptual stage has been refined to the CA^{2}RVE architecture, which has been applied to a realistic evaluation scenario. It was then successfully demonstrated that the generalized concept can be realized and customized such that the intended use case could be solved: populating a real-time traffic simulation with cognitive agents. The generic trait-based personality profile does not constrain the architecture to a specific personality model. Once a personality model has been selected, the proposed mapping process allows transferring results from psychology studies to task-specific agent behavior. It was shown how task parameters can be used to relate agent behavior to personality. Finally, combining the personality profile with an emotional state facilitates adaptation of generated behavior in dynamic scenes and virtual environments.

Future Work

Being able to successfully realize the developed concepts within the target domain proves that the objectives of this thesis project are achieved by the provided architecture. To further investigate the presented approach, it would be interesting if other researchers would utilize the architecture to add more applications and application areas.

One aspect that could be addressed in the future is the configuration of perception components using an agent's personality and emotional state. It was discussed how this could be achieved by extending the existing task parameter concept to include perception as well. The presented efforts to improve scalability and real-time applicability of the proposed architecture could be expanded by exploring additional optimizations, which are not directly related to the architecture itself. Examples might be: Optimizing the render process, load balancing and scheduling of agent updates, batching or instantiation techniques, or source code optimization. Regarding decision-making, another meaningful extension to the proposed architecture could be reasoning approaches that achieve more behavioral variety. It would be interesting to explore reasoning approaches that combine with perception, personality, and emotion to generate "true" alternative behavior, i.e., actions that are not pre-determined at design time. However, currently it is unclear whether this level of autonomy can be achieved at all. Resulting high-level processes could also be more tightly integrated into the

developed perception framework. By these means, decision processes could be modeled such that they are more involved in virtual perception, e.g., deciding how long an attended object must be observed until it is perceived. Other considerations could be: How long will the agent be able to remember having seen a particular object? How does the type of object or interaction influence an agent's ability to recall an object in the future? How does distraction influence the perception process? These are just a few interesting perception-related questions that could be investigated in future work.

Lastly, during this thesis project, the field of *deep learning* (DL) experienced a revolution caused by advances in processing power and the availability of a vast amount of training data [Sej18]. Going forward, the combination of cognitive real-time agents and DL would be an interesting research topic. For example, DL image processing techniques could be used to realize image-based vision sensors or action-selection methods could be based on DL. These examples indicate how cognitive agents could be improved by utilizing DL. However, it would also be possible for DL approaches to benefit from an agent simulation. Considering the presented application scenario of road traffic simulation, cognitive agents could be used to generate large amounts of observable, virtual data for training DL mechanisms of self-driving cars. This method would mitigate the need for acquiring sensor data from the real world and having to manually annotate it.

Although multiple directions for future research exist, the presented architecture concept as well as its realization represent a fundamentally new framework for attentive cognitive agents in real-time virtual environments. Furthermore, the provided descriptions of background information, design choices, and examples can certainly support other researchers or application developers in their own endeavors of trying to simulate more human-like software agents in real-time virtual environments.

Bibliography

[3DJ] 3DJunior. asset design: https://archive3d.net/?a=download&id=964df051, original design retrived in 2012, last accessed 2023-01-15.

[Act14] Mike Acton. Data-Oriented Design and C++. In *CppCon*, 2014. https://youtu.be/rX0ItVEVjHc. Online, accessed: 2021-07-26.

[AEO05] Charles H. Anderson, David C. Van Essen, and Bruno A. Olshausen. Directed Visual Attention and the Dynamic Control of Information Flow. In Laurent Itti, Geraint Rees, and John K. Tsotsos, editors, *Neurobiology of Attention*, chapter 3, pages 11–17. Academic Press, 2005.

[AKG+00] Elisabeth André, Martin Klesen, Patrick Gebhard, Steve Allen, and Thomas Rist. Integrating Models of Personality and Emotions into Lifelike Characters. In Ana Paiva, editor, *Affective Interactions: Towards a New Generation of Computer Interfaces*, pages 150–165. Springer Berlin Heidelberg, 2000.

[ALD11] Christopher Steven Applegate, Stephen David Laycock, and Andrew Day. A Sketch-Based System for Highway Design. In *Proc. 8th Eurographics Sym. Sketch-Based Interfaces and Modeling*, SBIM '11, pages 55–62, 2011.

[ALS09] Raúl Arrabales, Agapito Ledezma, and Araceli Sanchis. Towards Consciousness-like Behavior in Computer Game Characters. In *IEEE Conf. Computational Intelligence and Games*, CIG '09, pages 217–224, 2009.

[AML+12] Raúl Arrabales, Jorge Muñoz, Agapito Ledezma, German Gutierrez, and Araceli Sanchis. A Machine Consciousness Approach to the Design of Human-Like Bots. In Philip Hingston, editor, *Believable Bots: Can Computers Play Like People?*, pages 171–191. Springer Berlin Heidelberg, 2012.

[APMG12] Sean Andrist, Tomislav Pejsa, Bilge Mutlu, and Michael Gleicher. Designing Effective Gaze Mechanisms for Virtual Agents. In *Proc. SIGCHI Conf. Human Factors in Computing Systems*, CHI '12, pages 705–714. ACM, 2012.

[ARM15] ARM. ARM Guide for Unity Developers: Real-Time 3D Art Best Practices: Lighting – Light Probes. https://developer.arm.com/documentation/102109/0100/Light-probes, 2015. Online, accessed: 2022-05-24.

[AS68] Richard C. Atkinson and Richard M. Shiffrin. Human Memory: A Proposed System and its Control Processes. *The Psychology of Learning and Motivation*, 2:89–195, 1968.

[AS16] Richard C. Atkinson and Richard M. Shijfrin. Human Memory: A Proposed System and Its Control Processes. *Scientists Making a Difference: One Hundred Eminent Behavioral and Brain Scientists Talk About Their Most Important Contributions*, page 115, 2016.

[ASA21] ASAM e.V. ASAM OpenDRIVE®. `https://www.asam.net/standards/detail/opendrive/`, 2021. Online, accessed: 2022-05-22.

[BA13] Tim Balint and Jan M. Allbeck. What's Going on? Multi-sense Attention for Virtual Agents. In Ruth Aylett, Brigitte Krenn, Catherine Pelachaud, and Hiroshi Shimodaira, editors, *Proc. 13th ACM Int. Conf. Intelligent Virtual Agents*, IVA '13, pages 349–357. Springer Berlin Heidelberg, 2013.

[Bad03] Alan D. Baddeley. *Human Memory: Theory and Practice*. Psychology Press Ltd., 2 edition, 2003.

[Bar02] Christoph Bartneck. Integrating the OCC Model of Emotions in Embodied Characters. In *Workshop on Virtual Conversational Characters: Applications, Methods, and Research Challenges*, VCC '02, 2002.

[Bar10] Jaume Barceló. *Fundamentals of Traffic Simulation*. Springer, New York, NY, 2010.

[BBT99] Christophe Bordeux, Ronan Boulic, and Daniel Thalmann. An Efficient and Flexible Perception Pipeline for Autonomous Agents. *Computer Graphics Forum*, 18:23–30, 1999.

[BdSP+10] Elisabetta Bevacqua, Etienne de Sevin, Catherine Pelachaud, Margaret McRorie, and Ian Sneddon. Building Credible Agents: Behaviour Influenced by Personality and Emotional Traits. In Carole Bouchard, Améziane Aoussat, Pierre Lévy, and Toshimasa Yamanaka, editors, *Proc. 2nd Int. Conf. Kansei Engineering and Emotion Research*, KEER '10, pages 1071–1080, 2010.

[BGG19] Shakir Belle, Curtis Gittens, and Nicholas Graham. Programming with Affect: How Behaviour Trees and a Lightweight Cognitive Architecture Enable the Development of Non-Player Characters with Emotions. In *IEEE Games, Entertainment, Media Conf.*, pages 1–8, 2019.

[BI13] Ali Borji and Laurent Itti. State-of-the-Art in Visual Attention Modeling. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 35(1):185–207, 1 2013.

[Bia02] Gari Biasillo. Representing a Racetrack for the AI. In Steve Rabin, editor, *AI Game Programming Wisdom*, pages 439–443. Charles River Media, 2002.

[BKA+05a] Selmer Bringsjord, Sangeet Khemlani, Konstantine Arkoudas, Chris McEvoy, Marc Destefano, and Matthew Daigle. Advanced Synthetic Characters, Evil, and E. In *Int. Conf. Intelligent Games and Simulation*, Game-On '05, 2005.

[BKA05b] Wilco Burghout, Haris Koutsopoulos, and Ingmar Andréasson. Hybrid Mesoscopic-Microscopic Traffic Simulation. *Transportation Research Record*, 1934:218–255, 1 2005.

[BL06] Bradley J. Best and Christian Lebiere. Cognitive Agents Interacting in Real and Virtual Worlds. In Ron Sun, editor, *Cognition and Multi-Agent Interaction*, pages 186–218. Cambridge University Press, 2006.

[Blu97] Bruce Mitchell Blumberg. *Old Tricks, New Dogs: Ethology and Interactive Creatures*. PhD thesis, Massachusetts Institute of Technology, 1997.

[Boe17] Geoff Boeing. OSMnx: New Methods for Acquiring, Constructing, Analyzing, and Visualizing Complex Street Networks. *Computers, Environment and Urban Systems*, 65:126–139, 2017.

[Bor64] Edgar F. Borgatta. The Structure of Personality Characteristics. *Behavioral Science*, 9:8–17, 1964.

[BP17] Roxanne Blouin-Payer. Helping it all Emerge - Managing Crowd AI in Watch Dogs 2. In *Proc. Game Developers Conference*, GDC '17, 2017. `https://youtu.be/LHEcpy4DjNc`. Online, accessed: 2021-06-28.

[BSFL+12] Martin Brunnhuber, Helmut Schrom-Feiertag, Christian Luksch, Thomas Matyus, and Gerd Hesina. Bridging the Gap Between Visual Exploration and Agent-based Pedestrian Simulation in a Virtual Environment. In *Proc. 18th ACM Sym. VR Software and Technology*, VRST '12, pages 9–16. ACM, 2012.

[BSGS13] Tobias Brosch, Klaus R. Scherer, Didier Grandjean, and David Sander. The Impact of emotion on perception, attention, memory, and decision-making. *Swiss Medical Weekly*, 143:w13786, 2013.

[BT09] Neil D. B. Bruce and John K. Tsotsos. Saliency, Attention, and Visual Search: An Information Theoretic Approach. *Vision*, 9(3):1–24, 3 2009.

[BTD14] Daniel A. Briley and Elliot M. Tucker-Drob. Genetic and Environmental Continuity in Personality Development: A Meta-Analysis. *Psychological Bulletin*, 140:1303–1331, 2014.

[BYMS06] Jeremy N. Bailenson, Nick Yee, Dan Merget, and Ralph Schroeder. The Effect of Behavioral Realism and Form Realism of Real-Time Avatar Faces on Verbal Disclosure, Nonverbal Disclosure, Emotion Recognition, and Copresence in Dyadic Interaction. *Presence: Teleoperators and Virtual Environments*, 15(4):359–372, 2006.

[Byr07] Michael D. Byrne. Cognitive Architecture. In Andrew Sears and Julie A. Jacko, editors, *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications*, chapter 5, pages 93–114. CRC Press, 2 edition, 2007.

[BZBP09] Hans-Joachim Bungartz, Stefan Zimmer, Martin Buchholz, and Dirk Pflüger. *Modellbildung und Simulation: Eine anwendungsorientierte Einführung*. Springer, 2009.

[Cat46] Raymond B. Cattell. *Description and Measurement of Personality*. Yonkers-on-Hudson, N.Y., World Book Company, 1946.

[Cat57] Raymond B. Cattell. *Personality and Motivation Structure and Measurement*. Yonkers-on-Hudson, N.Y., World Book Company, 1957.

[CBBP93] Gian Vittorio Caprara, Claudio Barbaranelli, Laura Borgogni, and Marco Perugini. The "Big Five Questionnaire". *Personality and Individual Differences*, 15(3):281 – 288, 1993.

[CBN03] Nurhan Cetin, Adrian Burri, and Kai Nagel. A Large-scale Agent-based Traffic Microsimulation Based on Queue Model. In *Proc. Swiss Transport Research Conf.*, STRC '03, 2003.

[CC76] James P. Curran and Raymond Bernard Cattell. *Manual of the Eight State Questionnaire*, 1976.

[CDB+02] Karen Craig, Jeff Doyal, Bryan Brett, C. Lebiere, E. Biefeld, and E. Martin. Development of a hybrid model of tactical fighter pilot behavior using IMPRINT task network model and ACT-R. In *Proc 11th Conf. Computer Generated Forces and Behavior Representation*, 2002.

[CEW+08] Guoning Chen, Gregory Esch, Peter Wonka, Pascal Müller, and Eugene Zhang. Interactive Procedural Street Modeling. *ACM Trans. Graphics*, 27:103:1–103:10, 2008.

[Cha07] Alex J. Champandard. Behavior Trees for Next-gen Game AI. In *Proc. Game Developers Conference*, GDC '07, 2007.

[CHLC18] Susana Castillo, Philipp Hahn, Katharina Legde, and Douglas W. Cunningham. Personality Analysis of Embodied Conversational Agents. In *Proc. 18th ACM Int. Conf. Intelligent Virtual Agents*, IVA '18, pages 227–232. Association for Computing Machinery, 2018.

[CKH+15] Lawrence Cavedon, Christian Kroos, Damith Herath, Denis Burnham, Laura Bishop, Yvonne Leung, and Catherine J. Stevens. "C'Mon dude!": Users adapt their behavior to a robotic agent with an attention model. *Int. J. Human-Computer Studies*, 80(C):14–23, 8 2015.

[CKN+07] Dongkyu Choi, Tolga Könik, Negin Nejati, Chunki Park, and Pat Langley. A Believable Agent for First-Person Shooter Games. In *Proc. 3rd AAAI Conf. Artificial Intelligence and Interactive Digital Entertainment*, AIIDE'07, pages 71–73, 2007.

[CL97] Michael Cremer and M. Landenfeld. A Mesoscopic Model for Saturated Urban Road Networks. *Traffic and Granular Flow*, pages 169–180, 1997.

[CM92] Paul T. Costa and Robert R. MacCrae. *Revised NEO Personality Inventory (NEO PI-R) and NEO Five-Factor Inventory (NEO-FFI): Professional Manual*. Psychological Assessment Resources, Inc., 1992.

[CM09] Philip J. Corr and Gerald Matthews. *The Cambridge Handbook of Personality Psychology*. Cambridge University Press, 2009.

[Cou15] Francois Cournoyer. Massive Crowd on Assassin's Creed Unity: AI Recycling. In *Proc. Game Developers Conference*, GDC '15, 2015. `https://youtu.be/Rz2cNWVLncI`. Online, accessed: 2021-06-28.

[CS09] Maria Cutumisu and Duane Szafron. An Architecture for Game Behavior AI: Behavior Multi-queues. In *Proc. 5th AAAI Conf. Artificial Intelligence and Interactive Digital Entertainment*, AIIDE'09, pages 20–27. AAAI Press, 2009.

[CSPC00] Justine Cassell, Joseph Sullivan, Scott Prevost, and Elizabeth F. Chruchill. *Embodied Conversational Agents*. MIT Press, 2000.

[CT04] Toni Conde and Daniel Thalmann. An Artificial Life Environment for Autonomous Virtual Agents with Multi-Sensorial and Multi-Perceptive Features. *Computer Animation and Virtual Worlds*, 15:311–318, 2004.

[CT06] Toni Conde and Daniel Thalmann. An Integrated Perception for Autonomous Virtual Agents: Active and Predictive Perception. *Computer Animation and Virtual Worlds*, 17:457–468, 2006.

[CTHG10] Julien Chaplier, Thomas Nguyen That, Marcus Hewatt, and Gilles Gallée. Toward a Standard: RoadXML, The Road Network Database Format. *Actes INRETS*, pages 211–220, 2010.

[CVB+12] Angelo Cafaro, Hannes Högni Vilhjálmsson, Timothy Bickmore, Dirk Heylen, Kamilla Rún Jóhannsdóttir, and Gunnar Steinn Valgarðsson. First Impressions: Users' Judgments of Virtual Agents' Personality and Interpersonal Attitude in First Encounters. In Yukiko Nakano, Michael Neff, Ana Paiva, and Marilyn Walker, editors, *Proc. 12th ACM Int. Conf. Intelligent Virtual Agents*, IVA '12, pages 67–80. Springer Berlin Heidelberg, 2012.

[Dig90] John M. Digman. Personality Structure: Emergence of the Five-Factor Model. *Annual Rev. of Psychology*, 41(1):417–440, 1990.

[DOP08] Włodzisław Duch, Richard J. Oentaryo, and Michel Pasquier. Cognitive Architectures: Where Do We Go from Here? In *Proc. 2008 Conf. Artificial General Intelligence 2008*, pages 122–136. IOS Press, 2008.

[Dos] Dosch Design Kommunikationsagentur GmbH. `https://trian3dbuilder.de/`, original design retrieved on 2013-10-30, original no longer available.

[DR10] M. Brent Donnellan and Richard W. Robins. Resilient, Overcontrolled, and Undercontrolled Personality Types: Issues and Controversies. *Social and Personality Psychology Compass*, 4(11):1070–1083, 2010.

[DSG10] Marius Dupuis, Martin Strobl, and Hans Grezlikowski. OpenDRIVE 2010 and Beyond – Status and Future of the de facto Standard for the Description of Road Networks. In *Proc. Driving Simulaton Conf.*, DSC '10, 2010.

[dSVA19] Felipe de Souza, Omer Verbas, and Joshua Auld. Mesoscopic Traffic Flow Model for Agent-based Simulation. *Proc. Computer Science*, 151:858–863, 1 2019.

[Enn04] James T. Enns. *The Thinking Eye, the Seeing Brain*. W. W. Norton & Company, 2004.

[Eys47] Hans Jürgen Eysenck. *Dimensions of Personality*. Kegan Paul and Trench and Trubner & Co., Ltd., 1947.

[Eys70] Hans Jürgen Eysenck. *The Structure of Human Personality*. Methuen's Manuals of Modern Psychology. Methuen, 1970.

[Fis49] Donald W. Fiske. Consistency of the Factorial Structures of Personality Ratings from Different Sources. *Abnormal and Social Psychology*, 44(3):329–344, 1949.

[Fri06] Simone Frintrop. *VOCUS: A Visual Attention System For Object Detection And Goal-directed Search*, volume 3899 of *Lecture Notes in Artificial Intelligence (LNAI)*. Springer Berlin Heidelberg, 2006.

[Fri11] Simone Frintrop. Computational Visual Attention. In Albert Ali Salah and Theo Gevers, editors, *Computer Analysis of Human Behavior*. Springer, 2011.

[FV10] Martin Fellendorf and Peter Vortisch. Microscopic Traffic Flow Simulator VISSIM. In *Fundamentals of Traffic Simulation*. Springer New York, 2010.

[Gaw98a] Christian Gawron. An Iterative Algorithm to Determine the Dynamic User Equilibrium in a Traffic Simulation Model. *Modern Physics C*, 9(3):393–407, 1998.

[Gaw98b] Christian Gawron. *Simulation-based Traffic Assignment*. PhD thesis, University of Cologne, 1998.

[GB16] Bruce Goldstein and James Brockmole. *Sensation and Perception*. Cengage Learning, 10 edition, 2016.

[GBK+10] Jakub Gemrot, Cyril Brom, Rudolf Kadlec, Michal Bída, Ondřej Burkert, Michal Zemčák, Radek Píbil, and Tomáš Plch. Pogamut 3 – Virtual Humans Made Simple. In Samia Nefti-Meziani and John Gray, editors, *Advances in Cognitive Systems*, pages 211–243. The Institution Of Engineering And Technology, 2010.

[Geb05] Patrick Gebhard. ALMA: A Layered Model of Affect. In *Proc. Int. Joint Conf. Autonomous Agents and Multiagent Systems*, AAMAS '05, pages 29–36. Association for Computing Machinery, 2005.

[Ger09] Antony P. Gerdelan. Driving Intelligence: A New Architecture and Novel Hybrid Algorithm for Next-Generation Urban Traffic Simulation. Technical report, Massey University, 2009.

[GKB+09] Jakub Gemrot, Rudolf Kadlec, Michal Bída, Ondřej Burkert, Radek Píbil, Jan Havlíček, Lukáš Zemčák, Juraj Šimlovič, Radim Vansa, Michal Štolba, Tomáš Plch, and Cyril Brom. Pogamut 3 Can Assist Developers in Building AI (Not Only) for Their Videogame Agents. In Frank Dignum, Jeff Bradshaw, Barry Silverman, and Willem van Doesburg, editors, *Agents for Games and Simulations: Trends in Techniques, Concepts and Design*, pages 1–15. Springer Berlin Heidelberg, 2009.

[GLBV08] Francisco Grimaldo, Miguel Lozano, Fernando Barber, and Guillermo Vigueras. Simulating Socially Intelligent Agents in Semantic Virtual Environments. *The Knowledge Engineering Review*, 23(4):369–388, 2008.

[GNN12] Dominik Grether, Andreas Neumann, and Kai Nagel. Simulation of Urban Traffic Control: A Queue Model Approach. *Proc. Computer Science*, 10:808–814, 2012.

[GPMG10] Eric Galin, Adrien Peytavie, Nicolas Maréchal, and Eric Guérin. Procedural Generation of Roads. *Computer Graphics Forum*, 29:429–438, 2010.

[GRS03] Samuel D. Gosling, Peter J. Rentfrow, and William B. Swann. A Very Brief Measure of the Big-Five Personality Domains. *Research in Personality*, 37(6):504 – 528, 2003.

[GSBS01] Maia Garau, Mel Slater, Simon Bee, and Martina Angela Sasse. The Impact of Eye Gaze on Communication Using Humanoid Avatars. In *Proc. SIGCHI Conf. Human Factors in Computing Systems*, CHI '01, pages 309–316. ACM New York, 2001.

[GSV+03] Maia Garau, Mel Slater, Vinoba Vinayagamoorthy, Andrea Brogni, Anthony Steed, and Angela M. Sasse. The Impact of Avatar Realism and Eye Gaze Control on Perceived Quality of Communication in a Shared Immersive Virtual Environment. In *Proc. SIGCHI Conf. Human Factors in Computing Systems*, CHI '03, pages 529–536, 2003.

[Han89] Peter A. Hancock. A Dynamic Model of Stress and Sustained Attention. *Human Factors and Ergonomics Society*, 31(5):519–537, 1989.

[Her09] Philipp Yorck Herzberg. Beyond "Accident-proneness": Using Five-Factor Model Prototypes to Predict Driving Behavior. *Research in Personality*, 43(6):1096 – 1100, 2009.

[HGH+18] Jack Harmer, Linus Gisslén, Henrik Holst, Joakim Bergdahl, Tom Olsson, Kristoffer Sjöö, and Magnus Nordin. Imitation Learning with Concurrent Actions in 3D Games. *Computing Research Repository*, abs/1803.05402, 2018.

[HHST02] Dirk Helbing, A. Hennecke, Vladimir Shvetsov, and Martin Treiber. Micro- and Macro-Simulation of Freeway Traffic. *Mathematical and Computer Modelling*, 35:517–547, 3 2002.

[Hin09] Philip Hingston. The 2K BotPrize. In *IEEE Conf. Computational Intelligence and Games*, CIG '09, 2009.

[HKRS95] Rainer Herpers, Holger Kattner, Holm Rodax, and Gerald Sommer. GAZE: An Attentive Processing Strategy to Detect and Analyze the Prominent Facial Regions. In Martin Bichsel, editor, *Proc. Int. Workshop Automatic Face- and Gesture-Recognition*, IWAFGR '95, pages 214–220. University of Zurich, 1995.

[HL15] Kaveh Hassani and Won-Sook Lee. An Intelligent Architecture for Autonomous Virtual Agents Inspired by Onboard Autonomy. In Plamen Angelov, Krassimir T. Atanassov, Lyubka Doukovska, Mincho Hadjiski, Vladimir Jotsov, Janusz Kacprzyk, Nikola K. Kasabov, Sotir Sotirov, Eulalia Szmidt, and Sławomir Zadrożny, editors, *Intelligent Systems*, IS '15, pages 391–402. Springer International Publishing, 2015.

[HR95] Barbara Hayes-Roth. An Architecture for Adaptive Intelligent Systems. *Artificial Intelligence*, 72(1):329 – 365, 1995.

[HR06] P. Y. Herzberg and M. Roth. Beyond Resilients, Undercontrollers, And Overcontrollers? An Extension of Personality Prototype Research. *Personality*, 20(1):5–28, 1 2006.

[HR15] Nader Hanna and Deborah Richards. The Influence of Users' Personality on the Perception of Intelligent Virtual Agents' Personality and the Trust Within a Collaborative Context. In Fernando Koch, Christian Guttmann, and Didac Busquets, editors, *Advances in Social Computing and Multiagent Systems*, MFSC '15, pages 31–47. Springer International Publishing, 2015.

[HSK+10] Rainer Herpers, David Scherfgen, Michael Kutz, Ulrich Hartmann, Oliver Schulzyk, Dietmar Reinert, and Holger Steiner. FIVIS – A Bicycle Simulation System. In Olaf Dössel and Wolfgang C. Schlegel, editors, *World Congress on Medical Physics and Biomedical Engineering*, pages 2132–2135. Springer Berlin Heidelberg, 2010.

[HSK+12] Rainer Herpers, David Scherfgen, Michael Kutz, Jens Bongartz, Ulrich Hartmann, Oliver Schulzyk, Sandra Boronas, Timur Saitov, Holger Steiner, and Dietmar Reinert. *Multimedia Sensory Cue Processing in the FIVIS Simulation Environment*, pages 217–233. IGI Global, 2012.

[IDP03] L. Itti, N. Dhavale, and F. Pighin. Realistic Avatar Eye and Head Animation Using a Neurobiological Model of Visual Attention. In B. Bosacchi, D. B. Fogel, and J. C. Bezdek, editors, *Proc. SPIE 48th Annual Int. Sym. Optical Science and Technology*, volume 5200, pages 64–78. SPIE Press, 8 2003.

[Isl05] Damian Isla. Handling Complexity in the Halo 2 AI. In *Proc. Game Developers Conference*, GDC '05, 2005.

[JBV+18] Arthur Juliani, Vincent-Pierre Berges, Esh Vckay, Yuan Gao, Hunter Henry, Marwan Mattar, and Danny Lange. Unity: A General Platform for Intelligent Agents. *arXiv e-prints*, 9 2018.

[JD12] Anja Johansson and Pierangelo Dell'Acqua. Emotional Behavior Trees. In *IEEE Conf. Computational Intelligence and Games*, CIG '12, pages 355–362, 2012.

[KBM18] Paulo Knob, Marcio Balotin, and Soraia Raupp Musse. Simulating Crowds with OCEAN Personality Traits. In *Proc. 18th Int. Conf. Intelligent Virtual Agents*, IVA '18, pages 233–238. Association for Computing Machinery, 2018.

[KH08] M. Kutz and R. Herpers. Urban Traffic Simulation for Games. In *Proc. Conf. Future Play: Research, Play, Share*, 2008.

[KHW+11] Andrey Kiselev, Benjamin Alexander Hacker, Thomas Wankerl, Niyaz Abdikeev, and Toyoaki Nishida. Toward Incorporating Emotions with Rationality into a Communicative Virtual Agent. *AI & SOCIETY*, 26(3):275–289, 2011.

[kila] kilastaras. profile: `https://free3d.com/user/kilastaras`, asset design: `https://free3d.com/3d-model/chevrolet-impala-44468.html`, original design retrieved in 2012, last accessed 2023-01-12.

[kilb] kilastaras. profile: `https://free3d.com/user/kilastaras`, asset design: `https://free3d.com/3d-model/alfa-romeo-159-ti-39437.html`, original desing retrieved in 2013, last accessed 2023-01-12.

[Kim06] Yougjun Kim. *A Computational Model of Dynamic Perceptual Attention for Virtual Humans*. PhD thesis, University of Southern California, 2006.

[KMT08] Zerrin Kasap and Nadia Magnenat-Thalmann. Intelligent Virtual Humans with Autonomy and Personality: State-of-the-Art. In Nadia Magnenat-Thalmann, Lakhmi C. Jain, and Nikhil Ichalkaranje, editors, *New Advances in Virtual Humans: Artificial Intelligence Environment*, pages 43–84. Springer Berlin Heidelberg, 2008.

[Kra10] Jan Kratochvíl. Living City in Mafia II. In *Proc. Game Developers Conf. Europe*, GDC Europe '10, 2010.

[Kra12] Jan Kratochvíl. Creating Living City for Openworld Game. Vienna Game/AI Conf., 2012.

[KS16] Hanneke Kersjes and Pieter Spronck. Modeling Believable Game Characters. In *IEEE Conf. Computational Intelligence in Games*, CIG '16, pages 193–201. IEEE Press, 2016.

[KT99] Marcelo Kallmann and Daniel Thalmann. Modeling Objects for Interaction Tasks. In Bruno Arnaldi and Gérard Hégron, editors, *Computer Animation and Simulation '98: Proc. Eurographics Workshop*, pages 73–86. Springer Vienna, 1999.

[KT18] Iuliia Kotseruba and John K. Tsotsos. 40 Years of Cognitive Architectures: Core Cognitive Abilities and Practical Applications. *Artificial Intelligence Review*, 7 2018.

[KTH07] Arne Kesting, Martin Treiber, and Dirk Helbing. General Lane-Changing Model MOBIL for Car-Following Models. *Transportation Research Record*, 1999(1):86–94, 2007.

[KU87] Christof Koch and Shimon Ullman. Shifts in Selective Visual Attention: Towards the Underlying Neural Circuitry. In Lucia M. Vaina, editor, *Matters of Intelligence: Conceptual Structures in Cognitive Neuroscience*, pages 115–141. Springer Netherlands, 1987.

[Kut09] Michael Kutz. Verkehrssimulation für virtuelle Umgebungen. Master's thesis, Bonn-Rhein-Sieg University of Applied Sciences, 2009.

[KvVH05] Youngjun Kim, Martin van Velsen, and Randall W. Hill. Modeling Dynamic Perceptual Attention in Complex Virtual Environments. In Themis Panayiotopoulos, Jonathan Gratch, Ruth Aylett, Daniel Ballin, Patrick Olivier, and Thomas Rist, editors, *Proc. 5th ACM Int. Conf. Intelligent Virtual Agents*, IVA '05, pages 266–277. Springer Berlin Heidelberg, 2005.

[KW11] Dane Kuiper and Rym Z. Wenkstern. Virtual Agent Perception in Large Scale Multi-Agent Based Simulation Systems. In *Proc. 10th Int. Conf. Autonomous Agents and Multiagent Systems*, AAMAS '11, pages 1235–1236. Int. Foundation for Autonomous Agents and Multiagent Systems, 2011.

[KW13] Dane M. Kuiper and Rym Z. Wenkstern. Virtual Agent Perception Combination in Multi Agent Based Systems. In *Proc. Int. Conf. Autonomous Agents and Multi-agent Systems*, AAMAS '13. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[KW15] Dane M. Kuiper and Rym Z. Wenkstern. Agent Vision in Multi-agent Based Simulation Systems. *Autonomous Agents and Multi-agent Systems*, 29(2):161–191, 3 2015.

[LA00] Michael Luck and Ruth Aylett. Applying Artificial Intelligence to Virtual Reality: Intelligent Virtual Environments. *Applied Artificial Intelligence*, 14:3–32, 2000.

[Lai01] John E. Laird. It Knows What You're Going To Do: Adding Anticipation to a Quakebot. In *Proc. 5th Int. Conf. Autonomous Agents*, AGENTS '01, pages 385–392. ACM, 2001.

[Lai08] John E. Laird. Extending the Soar Cognitive Architecture. In *Proc. Conf. Artificial General Intelligence*, AGI '08, pages 224–235. IOS Press, 2008.

[Lai12] John E. Laird. *The Soar Cognitive Architecture*. MIT Press, 2012.

[LB19] Joan Llobera and Ronan Boulic. A Tool to Design Interactive Characters Based on Embodied Cognition. *IEEE Trans. Games*, 11(4):311–319, 2019.

[LBB02] Sooha Park Lee, Jeremy B. Badler, and Norman I. Badler. Eyes Alive. *ACM Trans. Graphics*, 21(3):637–644, 2002.

[LC06] Pat Langley and Dongkyu Choi. A Unified Cognitive Architecture for Physical Agents. In *Proc. National Conf. Artificial Intelligence*, volume 21 number 2 of *AAAI '06*, pages 1469–1474, 2006.

[Lef18] Charles Lefebvre. Virtual Insanity: Meta AI on Assassin's Creed: Origins. In *Proc. Game Developers Conference*, GDC '18, 2018. `https://youtu.be/a09vnDjmY_E`. Online, accessed: 2021-06-28.

[Leo03] Tom Leonard. Building an AI Sensory System: Examining the Design of Thief: The Dark Project. `http://www.gamasutra.com/view/feature/131297`, 2003. Online, accessed: 2021-06-28.

[Li15] Jamy Li. The Benefit of Being Physically Present: A Survey of Experimental Works Comparing Copresent Robots, Telepresent Robots and Virtual Agents. *Human-Computer Studies*, 77:23–37, 2015.

[Lid04] Lars Lidén. Artificial Stupidity: The Art of Intentional Mistakes. In Steve Rabin, editor, *AI Game Programming Wisdom 2*. Charles River Media, 2004.

[Liv06] Daniel Livingstone. Turing's Test and Believable AI in Games. *Computers in Entertainment*, 4(1):6–13, 1 2006.

[LJ98] John E. Laird and Randolph M. Jones. Building Advanced Autonomous AI Systems for Large Scale Real Time Simulations. In *Computer Games Development Conference*, 1998.

[LL01] Marco Lauriola and Irwin P. Levin. Personality Traits and Risky Decision-making in a Controlled Experimental Task: An Exploratory Study. *Personality and Individual Differences*, 31(2):215 – 226, 2001.

[LL07] Scott D. Lathrop and John E. Laird. Towards Incorporating Visual Imagery Into a Cognitive Architecture. In *Proc 8th Int. Conf. Cognitive Modeling*, ICCM '07, page 25, 2007.

[LSC+09] Nan Li, David J. Stracuzzi, Gary Cleveland, Pat Langley, Tolga Konik, Dan Shapiro, Kamal Ali, Matthew Molineaux, and David W. Aha. Learning Hierarchical Skills for Game Agents from Video of Human Behavior. Technical report, KNEXUS RESEARCH Corp, 2009.

[LTC+01] James C. Lester, Stuart G. Towns, Charles B. Callaway, Jennifer L. Voerman, and Patrick J. FitzGerald. Deictic and Emotive Communication in Animated Pedagogical Agents. In Justine Cassell, Joseph Sullivan, Scott Prevost, and Elizabeth F. Churchill, editors, *Embodied Conversational Agents*, pages 123–154. MIT Press, 2001.

[Mat18] Gerald Matthews. Cognitive-Adaptive Trait Theory: A Shift in Perspective on Personality. *Personality*, 86(1):69–82, 2018.

[MB10] Rachel McDonnell and Martin Breidt. Face Reality: Investigating the Uncanny Valley for Virtual Faces. In *ACM SIGGRAPH ASIA 2010 Sketches*, SA '10, pages 41:1–41:2. ACM New York, 2010.

[MC08] Robert R. McCrae and Paul Costa. *Empirical and Theoretical Status of the Five-Factor Model of Personality Traits*, pages 273–294. SAGE Publications, Inc., 1 2008.

[MDWS00] Gerald Matthews, D. Roy Davies, Stephen J. Westerman, and Rob B. Stammers. Individual Differences: Personality and Mood. In *Human Performance: Cognition, Stress, and Individual Differences*, pages 265–286. Psychology Press, 2000.

[Meh96] Albert Mehrabian. Pleasure-Arousal-Dominance: A General Framework for Describing and Measuring Individual Differences in Temperament. *Current Psychology*, 14(4):261–292, 1996.

[MJ92] Robert R. McCrae and Oliver P. John. An Introduction to the Five-Factor Model and its Applications. *Personality*, 60:175–215, 1992.

[MS09] James McCrae and Karan Singh. Sketch-Based Path Design. In *Proc. Graphics Interface*, GI '09, pages 95–102, 2009.

[MS11] Torsten Metzler and Kristina Shea. Taxonomy of Cognitive Functions. In *Proc. 18th Int. Conf. on Engineering Design*, ICED '11, pages 330–341, 2011.

[Naj98] Lawrence J. Najjar. Principles of Educational Multimedia User Interface Design. *Human Factors*, 40(2):311–323, 1998.

[NFdSS10] Bressane Neto, Ary Fagundes, Corrêa da Silva, and Flávio Soares. On the Construction of Synthetic Characters with Personality and Emotion. In Antônio Carlos da Rocha Costa, Rosa Maria Vicari, and Flavio Tonidandel, editors, *Advances in Artificial Intelligence*, SBIA '10, pages 102–111. Springer Berlin Heidelberg, 2010.

[Nor63] Warren T. Norman. Toward an Adequate Taxonomy of Personality Attributes: Replicated Factor Structure in Peer Nomination Personality Ratings. *Abnormal and Social Psychology*, 66:574–583, 1963.

[NRTMT95] Hansrudi Noser, Olivier Renault, Daniel Thalmann, and Nadia Magnenat-Thalmann. Navigation for Digital Actors Based on Synthetic Vision, Memory, and Learning. *Computers & Graphics*, 19:7–19, 1995.

[NT95] Hansrudi Noser and Daniel Thalmann. Synthetic Vision and Audition for Digital Actors. *Computer Graphics Forum*, 14:325–336, 1995.

[OAVE93] Bruno A. Olshausen, Charles H. Anderson, and David C. Van Essen. A Neurobiological Model of Visual Attention and Invariant Pattern Recognition Based on Dynamic Routing of Information. *Neuroscience*, 13(11):4700–4719, 1993.

[OCC88] Andrew Ortony, Gerald L. Clore, and Allan Collins. *The Cognitive Structure of Emotions*. Cambridge University Press, 1988.

[OPOD10] Jan Ondřej, Julien Pettré, Anne-Hélène Olivier, and Stéphane Donikian. A Synthetic-vision Based Steering Approach for Crowd Simulation. In *ACM SIGGRAPH 2010 Papers*, SIGGRAPH '10, pages 123:1–123:9. ACM, 2010.

[Ort03] Anthony Ortony. On Making Believable Emotional Agents Believable. In Robert Trappl, Paolo Petta, and Sabine Payr, editors, *Emotions In Humans and Artifacts*. MIT Press, 2003.

[PAGM15] Tomislav Pejsa, Sean Andrist, Michael Gleicher, and Bilge Mutlu. Gaze and Attention Management for Embodied Conversational Agents. *ACM Trans. Interactive Intelligent Systems*, 5(1):3:1–3:34, 2015.

[Pal18] Jorge Palacios. *Unity 2018 Artificial Intelligence Cookbook Second Edition*. Packt Publishing, 2 edition, 2018.

[PCR+11] Christopher Edward Peters, Ginvera Castellano, Matthias Rehm, Elisabeth Andre, Amaryllis Raouzaiou, Kostas Rapantzikos, Kostas Karpouzis, Gaultiero Volpe, Antonio Camurri, and Asimina Vasalou. Fundamentals of Agent Perception and Attention Modelling. In *Emotion-Oriented Systems*. Springer, 2011.

[PD05] Sakda Panwai and Hussein Dia. Comparative Evaluation of Microscopic Car-following Behavior. *IEEE Trans. Intelligent Transportation Systems*, 6(3):314–325, 2005.

[PKB12] Anders Petersen, Søren Kyllingsbæk, and Claus Bundesen. Measuring and Modeling Attentional Dwell Time. *Psychonomic Bulletin & Review*, 19(6):1029–1046, 2012.

[PLI07] Rolf Pfeifer, Max Lungarella, and Fumiya Iida. Self-Organization, Embodiment, and Biologically Inspired Robotics. *Science*, 318(5853):1088–1093, 2007.

[PM01] Yoav I. H. Parish and Pascal Müller. Procedural Modeling of Cities. In *Proc. 28th Conf. Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 301–308. Association for Computing Machinery, 2001.

[PO02] Christopher Edward Peters and Carol O'Sullivan. Synthetic Vision and Memory for Autonomous Virtual Humans. *Computer Graphics Forum*, 21:743–752, 2002.

[Pom06] Marc Pomplun. Saccadic Selectivity in Complex Visual Search Displays. *Vision Research*, 46(12):1886–1900, 2006.

[POS03] Christopher Edward Peters and Carol O' Sullivan. Bottom-Up Visual Attention for Virtual Human Animation. In *Proc. 16th Int. Conf. on Computer Animation and Social Agents*, CASA '03. IEEE Computer Society, 2003.

[PT97] Paolo Petta and Robert Trappl. Why to Create Personalities for Synthetic Actors. In Paolo Petta and Robert Trappl, editors, *Creating Personalities for Synthetic Actors: Towards Autonomous Personality Agents*. Springer, 1997.

[RD08] Steve Rabin and Michael Delp. Designing a Realistic and Unified Agent-Sensing Model. In Scott Jacobs, editor, *Game Programming Gems 7*. Course Technology, 2008.

[Rey87] Craig W. Reynolds. Flocks, Herds and Schools: A Distributed Behavioral Model. *SIGGRAPH Computer Graphics*, 21(4):25–34, 8 1987.

[RGA+09] Michael Rushforth, Sudeep Gandhe, Ron Artstein, Antonio Roque, Sarrah Ali, Nicolle Whitman, and David Traum. Varying Personality in Spoken Dialogue with a Virtual Human. In Zsófia Ruttkay, Michael Kipp, Anton Nijholt, and Hannes Högni Vilhjálmsson, editors, *Proc. 9th ACM Int. Conf. Intelligent Virtual Agents*, IVA '09, pages 541–542. Springer Berlin Heidelberg, 2009.

[RJ00] Jeff Rickel and W. Lewis Johnson. Task-Oriented Collaboration with Embodied Agents in Virtual Worlds. In Justine Cassell, Joseph Sullivan, Scott Prevost, and Elizabeth F. Churchill, editors, *Embodied Conversational Agents*, pages 95–122. MIT Press, 2000.

[RJSL10] David Reitter, Ion Juvina, Andrea Stocco, and Christian Lebiere. Resistance is Futile: Winning Lemonade Market Share Through Metacognitive Reasoning in a Three-Agent Cooperative Game. In *Proc. 19th Conf. Behavior Representation in Modeling and Simulation 2010*, BRiMS '10, pages 120–127, 2010.

[RL97] Cheryl L. Rusting and Randy J. Larsen. Extraversion, Neuroticism, and Susceptibility to Positive and Negative Affect: A Test of Two Theoretical Models. *Personality and Individual Differences*, 22(5):607 – 612, 1997.

[RN10] Stuart Russell and Peter Norvig. *Artificial Intelligence*, volume 3. Pearson Education, Inc., 2010.

[RP12] Stefan Rank and Paolo Petta. Backstory Authoring for Affective Agents. In *Interactive Storytelling*, volume 7648 of *LNCS '12*, pages 144–149. Springer Berlin Heidelberg, 2012.

[RPA+15] Kerstin Ruhland, Christopher Edward Peters, Sean Andrist, Jeremy B. Badler, Norm I. Badler, Michael Gleicher, Bilge Mutlu, and Rachel McDonnell. A Review of Eye Gaze in Virtual Agents, Social Robotics and HCI: Behaviour Generation, User Interaction and Perception. *Computer Graphics Forum*, 34(6):299–326, 2015.

[RTO19] Frank E. Ritter, Farnaz Tehranchi, and Jacob D. Oury. ACT-R: A cognitive architecture for modeling cognition. *WIREs Cognitive Science*, 10(3):e1488, 2019.

[RVP13] Tiago Ribeiro, Marco Vala, and Ana Paiva. Censys: A Model for Distributed Embodied Cognition. In Ruth Aylett, Brigitte Krenn, Catherine Pelachaud, and Hiroshi Shimodaira, editors, *Proc. 13th ACM Int. Conf. Intelligent Virtual Agents*, IVA '13, pages 58–67. Springer Berlin Heidelberg, 2013.

[SCA19] Aleksandr Saprykin, Ndaona Chokani, and Reza S. Abhari. Large-scale Multi-agent Mobility Simulations on a GPU: Towards High Performance and Scalability. *Proc. Computer Science*, 151:733–738, 2019.

[Sej18] Terrence J. Sejnowski. *The Deep Learning Revolution*. The MIT Press, 2018.

[SEN99] Patrice M. Simon, Jörg Esser, and Kai Nagel. Simple Queueing Model Applied to the City of Portland. *Modern Physics C*, 10(5):941–960, 1999.

[SFC+10] Alberto Signoretti, Antonio Feitosa, André M. Campos, Anne M. Canuto, and Sergio V. Fialho. Increasing the Efficiency of NPCs Using a Focus of Attention Based on Emotions and Personality. In *2010 Brazilian Sym. Games and Digital Entertainment*, SBGames '10, pages 171–181, 11 2010.

[SFC+11] Alberto Signoretti, Antonino Feitosa, Andre M. Campos, Anne M. Canuto, Joao C. Xavier-Junior, and Sergio V. Fialho. Using an Affective Attention Focus for Improving the Reasoning Process and Behavior of Intelligent Agents. In *Proc. 2011 IEEE/WIC/ACM Int. Conf. Web Intelligence and Intelligent Agent Technology*, WI-IAT '11, pages 97–100. IEEE Computer Society, 2011.

[SGW12] Daniel L. Schacter, Daniel Todd Gilbert, and Daniel M. Wegner. *Psychology*. Palgrave Macmillan, 2012.

[SKW10a] Travis Steel, Dane M. Kuiper, and Rym Z. Wenkstern. Context-Aware Virtual Agents in Open Environments. In *Proc. 6th Int. Conf. Autonomic and Autonomous Systems*, ICAS '10, pages 90–96. IEEE Computer Society, 2010.

[SKW10b] Travis Steel, Dane M. Kuiper, and Rym Z. Wenkstern. Virtual Agent Perception in Multi-agent Based Simulation Systems. In *Proc. IEEE/WIC/ACM Int. Conf. Web Intelligence and Intelligent Agent Technology*, WI-IAT '10, pages 453–456. IEEE Computer Society, 2010.

[SL05] Johannes Strassner and Marion Langer. Virtual Humans with Personalized Perception and Dynamic Levels of Knowledge. *Computer Animation and Virtual Worlds*, 16:331–342, 2005.

[Smi67] Gene M. Smith. Usefulness of Peer Ratings of Personality in Educational Research. *Educational and Psychological Measurement*, 27:967–984, 1967.

[SOS10] William Steptoe, Oyewole Oyekoya, and Anthony Steed. Eyelid Kinematics for Virtual Characters. *Computer Animation and Virtual Worlds*, 21(3-4):161–171, 2010.

[SSSS16] Paul Richard Smart, Tom Scutt, Katia Sycara, and Nigel R. Shadbolt. Integrating ACT-R Cognitive Models with the Unity Game Engine. In Jeremy Owen Turner, Michael Nixon, Ulysses Bernardet, and Steve DiPaola, editors, *Integrating Cognitive Architectures into Virtual Character Design*, pages 35–64. IGI Global, 2016.

[sto] storque12. profile: https://free3d.com/user/storque12, asset design: https://free3d.com/3d-model/porsche-911-gt-43465.html, original design retrieved in 2015, last accessed 2023-01-12.

[TBSK08] Tim Tutenel, Rafael Bidarra, Ruben M. Smelik, and Klaas Jan De Kraker. The Role of Semantics in Games and Simulations. *Computers in Entertainment*, 6(4):57:1–57:35, 12 2008.

[TC92] Ernest C. Tupes and Raymond E. Christal. Recurrent Personality Factors Based on Trait Ratings. *Personality*, 60:225–251, 6 1992.

[TCW+95] John K. Tsotsos, Scan M. Culhane, Winky Yan Kei Wai, Yuzhong Lai, Neal Davis, and Fernando Nuflo. Modeling Visual Attention Via Selective Tuning. *Artificial Intelligence*, 78(1-2):507–545, 1995.

[Tha86] Robert E. Thayer. Activation-Deactivation Adjective Check List: Current Overview and Structural Analysis. *Psychological Reports*, 58:607–614, 1986.

[Tha89] R. E. Thayer. *Modern Perspectives on Mood*. Oxford University Press, 1989.

[THH00] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested Traffic States in Empirical Observations and Microscopic Simulations. *Physical Review E*, 62:1805–1824, 2000.

[TK10] Martin Treiber and Arne Kesting. *Verkehrsdynamik und -simulation*. Springer, Berlin, 2010.

[TR95] Demetri Terzopoulos and Tamer F. Rabie. Animat Vision: Active Vision in Artificial Animals. In *Proc. 5th Int. Conf. Computer Vision*, ICCV '95, 1995.

[Tri] TrianGraphics GmbH. Trian3DBuilder, https://trian3dbuilder.de/, accessed 2023-01-20.

[tSKT+20] Silke ter Stal, Lean Leonie Kramer, Monique Tabak, Harm op den Akker, and Hermie Hermens. Design Features of Embodied Conversational Agents in eHealth: a Literature Review. *Human-Computer Studies*, 138:102409, 2020.

[UMA] UMA Steering Group. UMA - Unity Multipurpose Avatar https://assetstore.unity.com/packages/3d/characters/uma-2-unity-multipurpose-avatar-35611, last accessed 2023-03-28.

[Und] Underground Lab. profile: https://sketchfab.com/xaverius0404, asset design: https://sketchfab.com/3d-models/lowpoly-honda-jazz-38ee278a018945538c7b34371699c3a0, original design retrieved in 2012, last accessed 2023-01-12.

[Uni] Unity Technologies. asset design: https://unity.com/, original designs ("Standard Asset Packages") retrieved in 2015, original no longer available.

[vO14] Joost van Oijen. *Cognitive Agents in Virtual Worlds: A Middleware Design Approach*. PhD thesis, Utrecht University, 2014.

[vOD11] Joost van Oijen and Frank Dignum. Scalable Perception for BDI-Agents Embodied in Virtual Environments. In *Proc. IEEE/WIC/ACM Int. Conf. Web Intelligence and Intelligent Agent Technology*, WI-IAT '11, pages 46–53. IEEE Computer Society, 2011.

[vOVD12] Joost van Oijen, Lois Vanhee, and Frank Dignum. CIGA: A Middleware for Intelligent Agents in Virtual Environments. In *Agents for Educational Games and Simulations*. Springer, 2012.

[VSB+20] Sarah Theres Völkel, Ramona Schödel, Daniel Buschek, Clemens Stachl, Verena Winterhalter, Markus Bühner, and Heinrich Hussmann. Developing a Personality Model for Speech-Based Conversational Agents Using the Psycholexical Approach. In *Proc. 2020 CHI Conf. Human Factors in Computing Systems*, CHI '20, pages 1–14. Association for Computing Machinery, 2020.

[VvWV00] Nico Vandaele, Tom van Woensel, and Aviel Verbruggen. A Queueing Based Traffic Flow Model. *Transportation Research Part D: Transport and Environment*, 5(2):121–135, 2000.

[vWV07] Tom van Woensel and Nico Vandaele. Modeling Traffic Flows with Queueing Models: A Review. *Asia-Pacific J. Operational Research*, 24(4):435–461, 2007.

[WC91] David Watson and Lee Anna Clark. Self- versus Peer Ratings of Specific Emotional Traits: Evidence of Convergent and Discriminant Validity. *Personality and Social Psychology*, 60(6):927–940, 1991.

[WC92a] D. Watson and L. A. Clark. Affects Separable and Inseparable: On the Hierarchical Arrangement of the Negative Affects. *Personality and Social Psychology*, 62(3):489–505, 1992.

[WC92b] David Watson and Lee Anna Clark. On Traits and Temperament: General and Specific Factors of Emotional Experience and Their Relation to the Five-Factor Model. *Personality*, 60(2):441–476, 1992.

[WCT88] David Watson, Lee Anna Clark, and Auke Tellegen. Development and Validation of Brief Measures of Positive and Negative Affect: The PANAS scales. *Personality and Social Psychology*, 54(6):1063–70, 6 1988.

[YR01] Michelle S. M. Yik and James A. Russell. Predicting the Big Two of Affect from the Big Five of Personality. *Research in Personality*, 35(3):247 – 277, 2001.

[YT15] Georgios N. Yannakakis and Julian Togelius. A Panorama of Artificial and Computational Intelligence in Games. *IEEE Trans. Computational Intelligence and AI in Games*, 7(4):317–335, 2015.

[YT18] Georgios N. Yannakakis and Julian Togelius. *Artificial Intelligence and Games*. Springer, 2018.

[ZMLY19] Michelle X. Zhou, Gloria Mark, Jingyi Li, and Huahai Yang. Trusting Virtual Agents: The Effect of Personality. *ACM Trans. Interactive Intelligent Systems*, 9(2–3):1–36, 3 2019.

Author's Publications

[SvS1] Helmut Buhler, Jonas Schild, Sven Seele, and Rainer Herpers. Integration von Panorama-Bilddaten in eine Echtzeit-Game Engine für Virtual Reality Szenen. In *13. Workshop Virtuelle Realität und Augmented Reality der GI-Fachgruppe VR/AR* (2016).

[SvS2] Thomas Dettmar, Sven Seele, Rainer Herpers, and Peter Becker. Multi-Level Traffic Simulations for Virtual Environments. In *Tagungsband Sommertreffen Verkehrssimulation 2012* (2012).

[SvS3] Thomas Dettmar, Sven Seele, Rainer Herpers, Peter Becker, and Christian Bauckhage. Efficient Mesoscopic Simulations for Persistent Agents in 3D-Applications and Games. In *Proc. 5th Int. Conf. Games and Virtual Worlds for Serious Applications (VS-Games)* (2013).

[SvS4] Tobias Haubrich, Sven Seele, Rainer Herpers, Christian Bauckhage, and Peter Becker. Synthetic Perception for Intelligent Virtual Agents. In *Proc. ACM SIGCHI Annual Sym. Computer-human Interaction in Play* (2014), CHI PLAY '14, doi:10.1145/2658537.2661302, pp. 421–422.

[SvS5] Tobias Haubrich, Sven Seele, Rainer Herpers, Christian Bauckhage, and Peter Becker. *Modeling Sensation for an Intelligent Virtual Agent's Perception Process*. IVA '15. Springer International Publishing, 2015, pp. 87–97.

[SvS6] Tobias Haubrich, Sven Seele, Rainer Herpers, and Peter Becker. Integration of Road Network Logics into Virtual Environments. In *Proc. IEEE Virtual Reality* (3 2014), IEEE VR '14, pp. 79–80.

[SvS7] Tobias Haubrich, Sven Seele, Rainer Herpers, Martin E. Müller, and Peter Becker. Semantic Road Network Models for Rapid 3D Traffic Scenario Generation. In *Tagungsband ASIM/GI-Fachgruppentreffen STS/GMMS, Workshop Simulation technischer Systeme - Grundlagen und Methoden in Modellbildung und Simulation* (2013).

[SvS8] Tobias Haubrich, Sven Seele, Rainer Herpers, Martin E. Müller, and Peter Becker. A Semantic Road Network Model for Traffic Simulations in Virtual Environments: Generation and Integration. In *7th Workshop on Software Engineering and Architectures for Realtime Interactive Systems* (2014), SEARIS '14.

[SvS9] Rainer Herpers, Peter Becker, Sven Seele, David Scherfgen, and Timur Saitov. Agentenbasierte Verkehrssimulation mit psychologischen Persönlichkeitsprofilen. Technical report, Bonn-Rhein-Sieg University of Applied Sciences, Department of Computer Science, Institute of Visual Computing, 2015.

[SvS10] Alina Ibbeken, Fenja Zell, Christina Hagen, Sven Seele, Ulrike Grzyska, Alex Frydrychowicz, Armin Steffen, and Thorsten M. Buzug. Deformation Measurements With a Flexible Pharyngeal Phantom. *Biomedical Engineering/Biomedizinische Technik* (2021).

[SvS11] Steffen Kampmann, Sven Seele, Rainer Herpers, Christian Bauckhage, and Peter Becker. Automatic Mapping of Human Behavior to Personality Model Parameter for Traffic Simulations in Virtual Environments. In *IEEE Conf. Computational Intelligence and Games* (2015), CIG '15.

[SvS12] Fabian Krueger, Sven Seele, Rainer Herpers, Christian Bauckhage, and Peter Becker. Dynamic Emotional States Based on Personality Profiles for Adaptive Agent Behavior Patterns. In *11. Workshop Virtuelle Realität und Augmented Reality der GI-Fachgruppe VR/AR* (2014).

[SvS13] Fabian Krueger, Sven Seele, Rainer Herpers, and Peter Becker. Dynamic Emotional States based on Personality Profiles for Adaptive Agent Behavior Patterns. Tech. rep., Bonn-Rhein-Sieg University of Applied Sciences, Department of Computer Science, Institute of Visual Computing, 2013. ISSN 1869-5272, online: http://opus.bib.hochschule-bonn-rhein-sieg.de/opus-3.3/volltexte/2013/16/.

[SvS14] Fabian Krueger, Sven Seele, Rainer Herpers, Peter Becker, and Christian Bauckhage. Adaptive Decision Making in Microsimulations of Urban Traffic in Virtual Environments. In *Entertainment Computing* (2014), ICEC '14, Springer Berlin, Heidelberg.

[SvS15] Jonas Schild, Sven Seele, Jonas Fischer, and Maic Masuch. Multi-pass Rendering of Stereoscopic Video on Consumer Graphics Cards. In *Proc. 2011 Sym. Interactive 3D Graphics and Games* (2011), i3D '11.

[SvS16] Jonas Schild, Sven Seele, and Maic Masuch. Integrating Stereoscopic Video in 3D Games. In *Entertainment Computing* (2011), ICEC '11.

[SvS17] Jonas Schild, Sven Seele, and Maic Masuch. YouDash3D – Exploring Depth-based Game Mechanics and Stereoscopic Video in S3D Gaming. In *Proc. 8th Int. Conf. Advances in Computer Entertainment Technology* (2011), ACE '11. Poster – Best Late Breaking Result Bronze Award.

[SvS18] Jonas Schild, Sven Seele, and Maic Masuch. YouDash3D: Exploring Stereoscopic 3D Gaming for 3D Movie Theaters. In *Proc. Stereoscopic Displays and Applications XXIII* (2012), SPIE '12.

[SvS19] Sven Seele, Thomas Dettmar, Rainer Herpers, Christian Bauckhage, and Peter Becker. Cognitive Aspects of Traffic Simulations in Virtual Environments. In *Tagungsband ASIM/GI-Fachgruppentreffen STS/GMMS, Workshop Simulation technischer Systeme - Grundlagen und Methoden in Modellbildung und Simulation* (2012).

[SvS20] Sven Seele, Thomas Dettmar, Rainer Herpers, Christian Bauckhage, and Peter Becker. Cognitive Aspects of Traffic Simulations in Virtual Environments. *Simulation Notes Europe (SNE) Special Issue: Simulation of Traffic Systems – Technical Systems 22*, 2 (8 2012), 83–88.

[SvS21] Sven Seele, Tobias Haubrich, Tim Metzler, Rainer Herpers, Jonas Schild, and Marcin Grzegorzek. Integration of Multi-modal Cues in Synthetic Attention Processes to Drive Virtual Agent Behavior. In *Proc. 17th ACM Int. Conf. Intelligent Virtual Agents* (2017), IVA '17.

[SvS22] Sven Seele, Tobias Haubrich, Jonas Schild, Rainer Herpers, and Marcin Grzegorzek. Augmenting Cognitive Processes and Behavior of Intelligent Virtual Agents by Modeling Synthetic Perception. In *Thematic Workshops'17 of ACM Multimedia 2017 Proceedings* (2017), doi:10.1145/3126686.3126752.

[SvS23] Sven Seele, Rainer Herpers, and Christian Bauckhage. Cognitive Agents for Microscopic Traffic Simulations in Virtual Environments. In *Entertainment Computing* (2012), ICEC '12.

[SvS24] Sven Seele, Rainer Herpers, and Christian Bauckhage. Cognitive Agents with Psychological Personality Profiles for Traffic Simulations in Virtual Environments. In *Tagungsband Sommertreffen Verkehrssimulation 2012* (2012).

[SvS25] Sven Seele, Sebastian Misztal, Helmut Buhler, Rainer Herpers, and Jonas Schild. Here's Looking At You Anyway! How Important is Realistic Gaze Behavior in Co-located Social Virtual Reality Games? In *Proc. 2nd ACM SIGCHI Sym. Computer-human Interaction in Play* (2017), CHI PLAY '17, doi:10.1145/3116595.3116619.

[SvS26] Sven Seele, Luisa Pätzold, and Rainer Herpers. Gone But Not Forgotten: Evaluating Performance and Scalability of Real-Time Mesoscopic Agents. In *Proc. 20th ACM Int. Conf. Intelligent Virtual Agents* (2020), IVA '20, doi:10.1145/3383652.3423891.

[SvS27] Suzannah Smith, Sven Seele, and Rainer Herpers. Modeling Synthetic Perception for Virtual Agents. In *Science Atlantic Conference, Mathematics, Statistics, and Computer Science Conf.* (2017).

[SvS28] Konstantin Wegner, Sven Seele, Helmut Buhler, Sebastian Misztal, Rainer Herpers, and Jonas Schild. Comparison of two inventory design concepts in a collaborative virtual reality serious game. In *Extended Abstracts 2nd ACM SIGCHI Sym. on Computer-human Interaction in Play* (2017), CHI PLAY '17, pp. 323 – 329.

Theses and Projects Supervised by the Author

[IVC1] Francis Beauchemin. Realization of a Road Network Editing Tool in the Unity Game Enginge, 2014.

[IVC2] Björn Buhr. Workflow zur Integration Realistischer Fußgängeranimationen in das AVeSi-Projekt, 2014.

[IVC3] Xinyi Chen. Realization of Right-of-way Priorities in the Semantic Road Network Representation SeRoNet, 2013.

[IVC4] Thomas Dettmar. Macroscopic Traffic Simulations for Virtual Environments, 2012.

[IVC5] Thomas Dettmar. Queuing Models for Traffic Simulations in Virtual Environments, 2013.

[IVC6] Tim Grundmanns. Modelling Behavior for Personalized Virtual Agents, 2021.

[IVC7] Tobias Haubrich. Konzeption einer Verkehrsnetzrepräsentation für kognitive Agenten in virtuellen Umgebungen, 2013.

[IVC8] Tobias Haubrich. Entwicklung eines synthetischen Perzeptionsprozesses für intelligente virtuelle Agenten, 2014.

[IVC9] Norbert Jörgensen. Entwicklung eines Physikalisch Basierten Fahrzeugmodells für Virtuelle Umgebungen, 2014.

[IVC10] Steffen Kampmann. Machine Learning Based Driver Behavior in Traffic Simulations for Virtual Environments, 2013.

[IVC11] Steffen Kampmann. Inference-based Model Analysis for Traffic Simulations in Virtual Environments, 2014.

[IVC12] Marco Koppenol, and Timo Rothdeutsch. viPerEVAL: Framework zur Datenerhebung und Evaluation für Perzeptionsmodelle intelligenter virtueller Agenten, 2016.

[IVC13] Fabian Krueger. Persönlichkeitsprofile für Mikrosimulationen des Straßenverkehrs in Virtuellen Umgebungen, 2013.

[IVC14] Luisa Pätzold. Evaluating Performance and Scalability of Real-Time Mesoscopic Agents, 2020.

[IVC15] Suzannah Smith. Heat Map Visualization of Eye-tracking Data in VR, 2017.

A

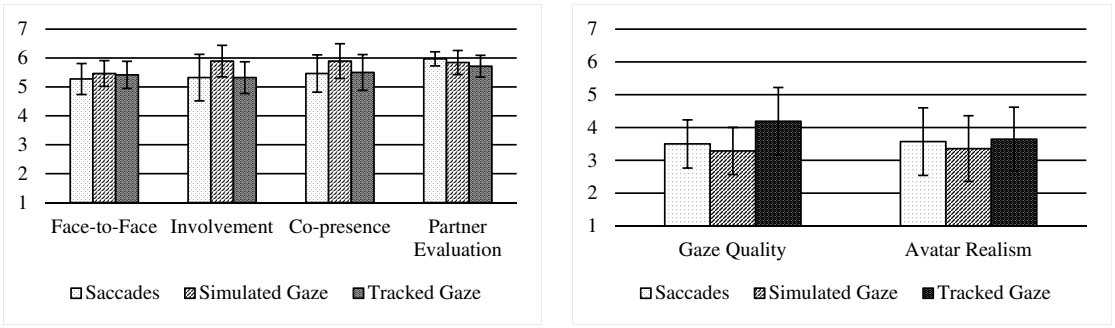
Beyond Traffic Agents

In Chapter 6, the realization of the proposed cognitive agent architecture is discussed in the context of a traffic simulation for road safety education. One aspect that has not been addressed explicitly is directly communicating agent perception to a user during interaction. In the example of traffic simulators, eye gaze and head orientation are important indicators about what an agent is focusing. However, these movements become even more important when users interact with agents more closely (e.g., see [BYMS06, GSV+03, LBB02, RPA+15]). The significance is further increased when engaging with conversational agents or user avatars in VR. To investigate this claim in a social VE setting, a study was conducted that compared off-the-shelf gaze behavior software to eye movements tracked from users using a VR headset [SvS25].

The study was based on studies performed by Garau et al. [GSBS01, GSV+03] with additional elements from McDonnell and Breidt [MB10], and Steptoe et al. [SOS10]. Since the focus of this study was the effect of gaze behavior on the perception of and interaction with virtual characters, the use of agents in the study was ruled out to avoid any side effects caused by generated agent behavior. Instead, two human subjects engaged in a social interaction through virtual avatars as shown in Figure A.1. Both subjects entered a co-located virtual space using a collaborative VR framework and were seated at a table to

(a) (b)

Figure A.1: Experimental setup of the gaze behavior study with two human subjects sitting across each other at table (a) and the same scene in VR (b)*. Images from [SvS25].
*Scene assets designed by IVC and characters generated using Unity Multipurpose Avatar [UMA].



(a) (b)

Figure A.2: Results of the gaze behavior study showing mean scores for (a) quality of communication and (b) gaze quality and avatar realism. 95% confidence intervals are depicted by error bars. Images from [SvS25].

minimize fatigue. The same scene was virtually replicated to match perceptions from the real and virtual world. Hands were controlled using HTC Vive controllers and head motion was driven by VR HMDs. One subject was wearing an HTC Vive headset and the other an Oculus Rift DK2 with integrated eye tracker. 42 subjects with mean age of 27.5 years and very low VR expertise (median of 1.0 on a scale from 1 to 7, with 7 being the highest score) were tested in pairs. Each test subject was assigned to one of three gaze behavior conditions that the subject could observe on its counterpart:

1. No gaze behavior: The eyes of other participants did not exhibit any gaze shifts, but off-the-shelf software was used to generate saccades[1] and blinks.
2. Simulated gaze behavior: In addition to saccades and blinks, fixations are simulated by generating a saliency map for the current view based on depth information only. A trained neural network predicts what a human would look at in a scene based on human training data with specific attention to human faces.
3. Eye-tracked gaze behavior: Eye movements and blinks are recorded by the Oculus HMD and transferred to the subject's avatar.

Each condition was assigned 14 times. Subjects were asked to engage in an unstructured conversation and play two co-operative word games. These pseudo tasks were meant to encourage participants to look at each other without revealing the intent of the study.

Using questionnaires, the study measured the influence of gaze behavior on perceived quality of communication and perceived realism of gaze behavior and avatar representation. One of the hypotheses was that condition 3 would have a significant impact on the quality of the experience. The implied consequence would have been that similar means are required to improve the perception of agent-controlled virtual humans, i.e., providing an intricate attention model for head movement and gaze shifts on top of saccadic eye movements. However, as shown by Figure A.2 no significant difference could be found between the three conditions. The quality of communication was reported to be high across all four

[1] Saccades are (mostly automatic) rapid eye movements used to scan a scene.

measured dimensions and across all gaze behavior conditions with mean scores ranging between 5.27 and 5.97. Gaze quality and avatar realism were not rated as favorably, but again no significant differences could be found between the three conditions.

Consequently, detailed gaze behavior models may not be as important for high quality communication in co-located VR environments as initially thought. It is reasonable to assume the same is true for interactions between a user and one or multiple agents, although this may have to be investigated separately. Despite the results, the findings are not sufficient to disprove the mentioned hypothesis as the subjects' lack of VR experience may have prevented them from noticing the subtle differences in gaze behavior. The novelty factor of VR may have outweighed all other effects. A similar assumption was derived from another performed VR study that compared a metaphoric with an abstract VR inventory [SvS28]. Additionally, the overall level of realism may have subverted user expectations, e.g., due to the lack of realistic facial expressions or mouth movements, users may not have expected to observe realistic eye movements and as a result may have not paid attention to it. Finally, other pseudo tasks may be better fitted to investigate the impact of gaze behavior. Further studies would have to systematically eliminate these possible interference factors, before being able to dismiss the hypothesis.

B

Level of Detail Evaluation Scenario Results

To investigate the mesoscopic simulation system (see Section 6.3), an evaluation based on a quadratic traffic network grid is discussed in Section 7.4.1. The evaluation method is described, and the results show that 100000 mesoscopic agents can be simulated in large networks of 15000 nodes and about 60000 edges with interactive framerates above 30 FPS. A similar experiment was performed previously and reported in [SvS3]. This previous evaluation tested a limited number of agents and network sizes due to the intended application to the FIVIS bicycle simulator [HSK+10, HSK+12], which required a network of 24 nodes and a population of 200 traffic agents.

The original publication presented the evaluation results using frames per second (FPS). With this measurement it is difficult to observe the actual changes caused by adding or removing nodes or agents. This difficulty is due to the frame rate decreasing and asymptotically approaching 0 FPS. Furthermore, by measuring FPS it is difficult to evaluate the mesoscopic system's integration into a larger system, like a simulator or game. Thus, the results from [SvS3] are transformed here to show how much time it takes to calculate one frame (milliseconds per frame). This transformation prevents the mentioned issues and simplifies drawing conclusions from the results.

Network graphs for $n \in \{10, \ldots, 25\}$ (i.e., 100 to 625 nodes) were simulated using an agent population of 1000, 10000, 50000, and 100000. The results depicted in Figure B.1 (a) show a more noticeable change in slope when increasing the agent population size, but otherwise support the observations stated in Section 7.4.1. The calculation times grow linearly with either the number of agents or the network size, and all configurations could be simulated in less than half of the targeted frame calculation time of 33.33 *ms*. The highest measured average calculation time per frame is 10.94 *ms*.

Figure B.1 (b) shows the effect of increasing the number of agents in a network of constant size for 1000, 5000, 10000, 50000, and 100000 agents in a 25×25 network (625 nodes, 2400 edges). This depiction supports the linear correlation between the number of agents and the calculation time. Additionally, the minimum and maximum ranges for each simulation are plotted to show that the average calculation times closely correspond with the minimum calculation times. While most frames are calculated around the presented average times, there are several frames in each simulation that take more time to be completed. These

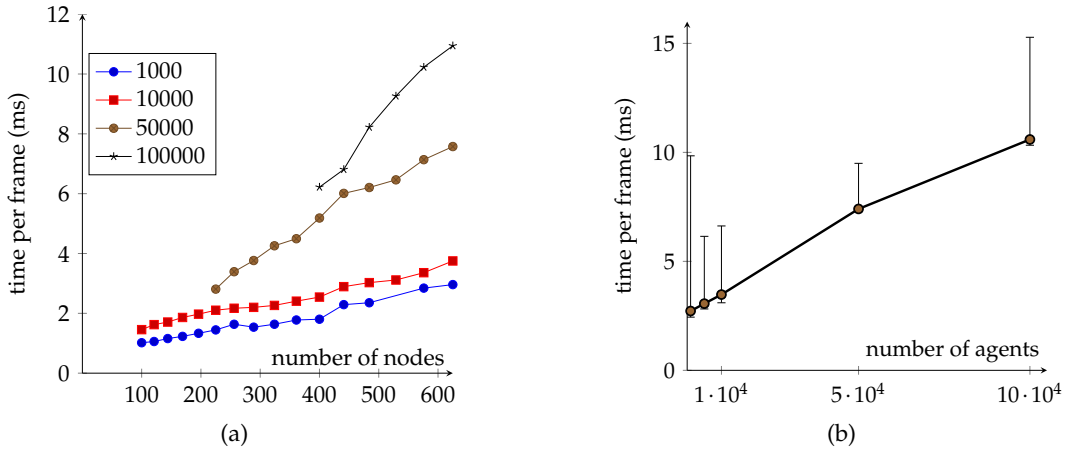


Figure B.1: Average times in milliseconds required to calculate each frame when increasing the size of a traffic network (a) and when increasing the number of agents simulated within a queuing model network graph of 25×25 intersections (b). Figure (b) also shows maximum and minimum calculation times. Average and maximum calculation times were below the target value of 33.33 *ms* per frame for all simulated configurations.

maxima also increase proportionally to the average calculation times with the highest single frame time being 15.49 *ms*. Both the highest single frame time and the highest average frame time (10.94 *ms*) were measured for the largest network ($n = 25$) simulating the largest number of agents (100000). This network size and the number of agents are sufficient for the system's original application within the FIVIS bicycle simulator.

C

Evaluating the Generation of Road Network Semantics for Cognitive Traffic Agents

As an answer to **RT4.1**, how knowledge can be provided to agents efficiently, the concept of road network semantics was introduced in Section 6.2. Besides efficiently providing knowledge to agents interacting in road traffic, the road network definition can be used to efficiently generate traffic networks, which was also mentioned in Section 6.2.

Three scenarios were used to evaluate the generation process: the Siegburg scene used in the FIVIS bicycle simulator, a part of the city of Sankt Augustin (the location of the Bonn-Rhein-Sieg University of Applied Sciences), and a larger area of the association of municipalities Hachenburg. For each scenario, a virtual scene was created using the workflow mentioned in Section 6.2 and detailed in [SvS7]. The workflow includes the generation of 3D meshes and according OpenDRIVE®[1] descriptions of the road network using Trian3D Builder[2]. The Unity game engine can import the generated 3D meshes. The OpenDRIVE® data is converted into the road network representation described in Section 6.2 within Unity using an in-house tool. Table C.1 summarizes the characteristics of the three generated scenes. Figure C.1 shows the OpenStreetMap[3] data used to create the Hachenburg scenario and the road network generated from that data. Two visual representations of the Siegburg scenario were additionally generated using alternative methods; one by manually creating the scene based on satellite images and one using the Esri CityEngine [PM01].[4]

For all scenes, the fit between the 3D scene and the road network was assessed to verify the correctness of the generation algorithm. Additionally, to empirically validate the road network, it was populated with agents and traffic simulations were run. Another major advantage of an automatic generation is the time saved over a manual creation of the network. Therefore, the generation time for each network was recorded and compared to

[1] `https://vires.com/`, [online: May 2, 2023] VIRES Simulationstechnologie GmbH

[2] `https://trian3dbuilder.de/`, [online: May 2, 2023] TrianGraphics

[3] `https://www.openstreetmap.org`, [online: May 2, 2023] Geofabrik, OpenStreetMap

[4] `https://www.esri.com/en-us/arcgis/products/esri-cityengine/overview`, [online: May 2, 2023] Esri CityEngine, Advanced 3D City Design Software

Table C.1: Characteristics for the three road network evaluation scenarios. The times for generating the road network representations and a manual setup of the same network were compared during the evaluation. The latter is extrapolated from an average setup time of 10 minutes for a single lane and its connections. Table from [SvS6]. © 2014 IEEE.

Scenario	Region	Dimensions	OpenDRIVE file size	# of lanes	# of connectors	# of junctions	# of nodes	Generation time	Est. manual setup time
1	Siegburg	1 $km \times 0.65$ km	1.85 MB	254	492	63	2902	1.84 s	42.33 h
2	Sankt Augustin	2 $km \times 1.5$ km	3.49 MB	573	950	154	6012	4.63 s	95.5 h
3	Hachenburg	10 $km \times 5$ km	14.18 MB	2017	3575	563	23902	62.14 s	336.17 h

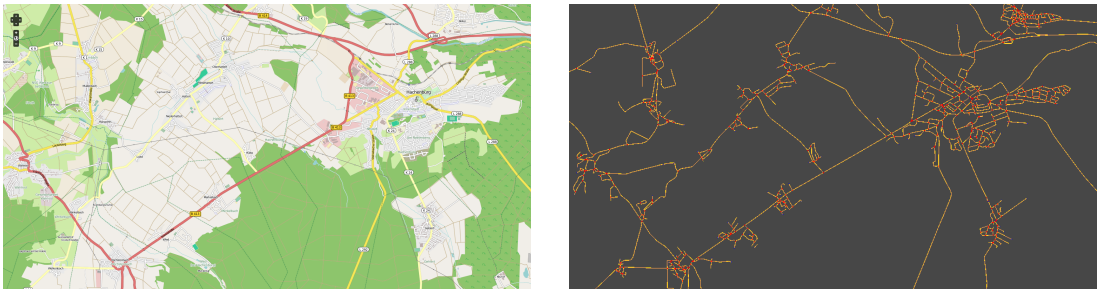


Figure C.1: OpenStreetMap data for scenario 3 and the road network representation generated from the data using the workflow discussed in Section 6.2 and detailed in [SvS7, IVC7]. Images from [SvS8]. © 2014 IEEE

an estimated manual setup of the same network. Further, scenario 2 was used for a manual inspection by an independent individual to document the quality of the generated network compared to the real world.

C.1 Applicability of Generated Road Network Semantics

For all scenarios, the 3D representation generated using the Trian3D Builder perfectly fit the generated road network. The Trian3D Builder generates both the 3D model as well as the according OpenDRIVE® data. Since the latter is used to generate the road network, the result was expected but also confirmed that the implemented algorithm correctly converts the data into road networks for CA^{2}RVE agents. Several examples from the result of scenario 3 are depicted in Figure C.2. The composition of the roundabout Figure C.2 (a) and the refuge island in Figure C.2 (c) seem like odd representations but are conform with the OpenDRIVE® standard. All scenes were successfully used to simulate agents navigating the network. An automatic validity check of the generated elements and their relationships was not realized during this thesis project but could be an interesting topic for future work.

To evaluate the quality of a fit between network and 3D representation, the 3D mesh for scenario 1 was also generated using the Esri CityEngine. This additional tool had not been used during the development of the generation workflow. The input for both the CityEngine and the road network generation workflow was a common OpenStreetMap shapefile. As a result, the reference lines for each road exactly match the center of the road's geometry. A close but not perfect fit could be achieved using the mesh generated by the CityEngine and

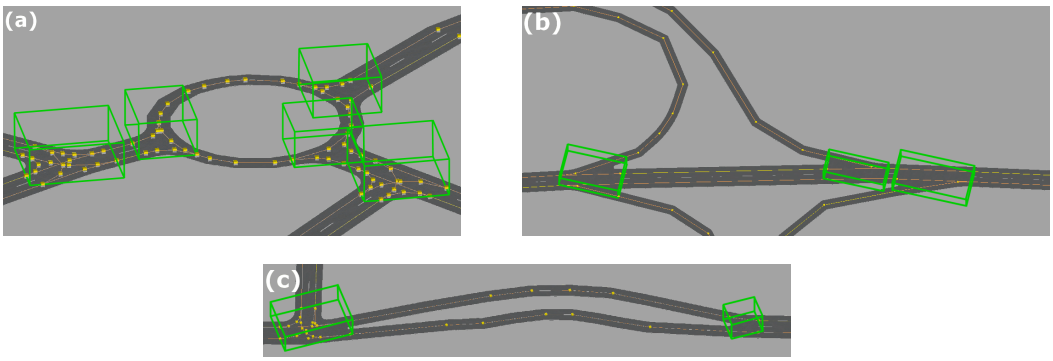


Figure C.2: A roundabout (a), a highway including exit and entrance ramps (b), and a refuge island (c) from scenario 3 are depicted to demonstrate the fit between the generated road network and 3D geometry. Green wireframe rectangles indicate junctions and yellow cubes represent waypoints (nodes) of *connectors* and *lanes*. Trian3D Builder was used to generate the geometry. Images based on [SvS8].

the generated road network as shown in Figure C.3. Only minor changes were necessary in select areas (e.g., positional and road width adjustments) to improve the fit and to allow simulating agents within the scenario.

The final 3D geometry was modeled manually as part of the FIVIS project. Due to different input data and imprecisions introduced by the manual modeling process, the generated network diverged noticeably from the created road geometry. Fitting the network to the geometry is still possible albeit requiring several manual adjustments, such as positioning of nodes and junctions. Manually adjusting an automatically generated network is still advantageous compared to a complete manual setup of the road network as demonstrated by the following section. Once adjustments were made, it was possible to populate the scene with agents and simulate the scenario.

C.2 Time Savings of Automatic Road Network Semantics Generation

Generating a road network using the described automatic process avoids errors and imprecisions of a manual creation, especially for large scenes. However, another major advantage is the time saved by an automatic process. Being able to efficiently generate networks allows for fast iterations and generating larger road networks. To estimate how much time the generation process saves compared to manual setups, several lanes of varying complexity were manually created. Setting up one lane includes: defining its geometry using waypoints, adding its information, creating succeeding and preceding connectors, and correctly connecting the lane to all adjacent connectors. The average of recorded creation times was 10 minutes, which is subsequently used to estimate the time savings for the three scenarios. All networks were generated using an Intel Core 2 Quad Q6700 processor and 4GB system RAM. The times required for generating the three evaluation scenarios and the estimates for a manual setup are listed in Table C.1. The table exemplifies that even for rather large areas, the road network can be generated in minutes while a manual setup would take weeks or

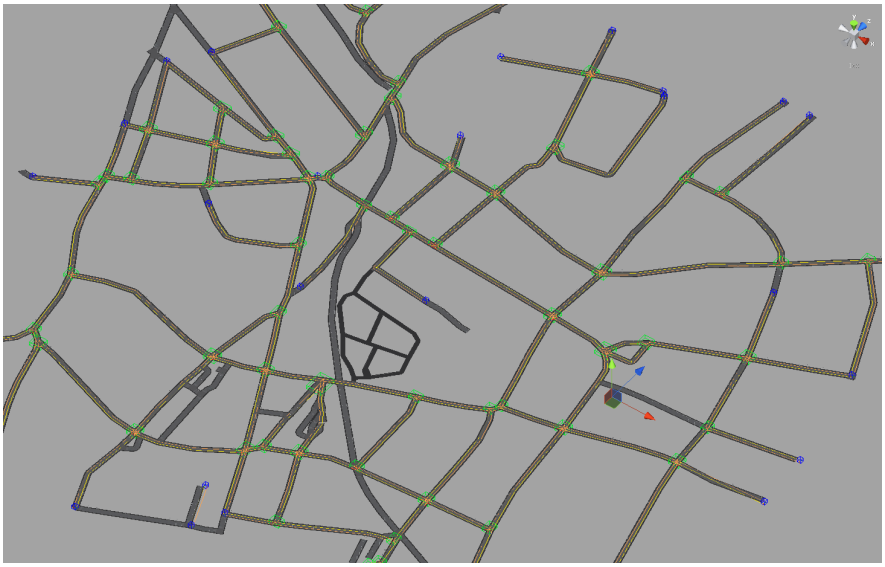


Figure C.3: Fit between the 3D model generated using the Esri CityEngine and the road network generated using the Trian3D Builder workflow. Some positional deviations exist and some roads are not represented. The latter is due to the different treatment of road annotations by the Trian3D Builder and the Esri CityEngine. Image based on [SvS8].

even months. For example, the road network for scenario 3 was generated in about one minute. The estimated manual setup time is 333.17 hours, which is equivalent to about 42 workdays or 8.5 work weeks. Even considering the time necessary to find or generate appropriate input data (i.e., OpenStreetMap files), pre-processing the data using the referenced workflow, and manually adjusting select areas, a significant amount of time can be saved.

Errors introduced by a manual setup were not considered in this comparison. However, the more complex a scenario, the more likely are mistakes made by a human designer. Finding and correcting theses mistakes requires additional time. Further, if a network needs to be revised, the generation process can be used to regenerate the network without the need for manual revisions.

C.3 Manual Inspection of Road Network Semantics Created by Different Means

To gain an independent assessment of the quality of the network generation process, one additional evaluation was performed. To this end, one more road network was generated for a scenario that had not been considered before. The generated road network was examined by a test person not involved in the project. The examiner was asked to generally compare the road network to reality using a digital map and satellite images. Afterwards, selected areas of interest were inspected and findings, documented by the examiner, were analyzed. To guide the examiner in documenting the findings, a questionnaire was provided that included questions such as: “In the presented section of the road network, does the course of the road match the map data?” or “Does the presented section of the road network contain all junctions present in the map data?” The complete list of questions can be found in [IVC7].

(a) (b)

Figure C.4: A section of the city of Sankt Augustin, Germany, selected for manual inspection to evaluate the quality of the fit between a generated road network and the real traffic network. The OpenStreetMap data shows the section used as input including 5 areas of interest selected for closer inspection (a). Areas 1 and 2 include intersections, 2 and 3 include roundabouts, and 5 includes a section of the Autobahn with entrance and exit ramps and a bridge crossing the Autobahn. Black lines in (b) represent the generated road network, which is displayed on top of a Google Maps section of the same area. Some deviations can be observed, which are due to the different base data. Images from [IVC7].

The criteria for selecting the scenario were the following:

- The examiner had to be sufficiently familiar with the scenario to reasonably assess the plausibility of the generated network.
- The examiner had to have easy access to the real scenario in case available geographic map data was not sufficient for comparisons.
- For a representative selection, the scenario should include:
  - Intersections of different sizes and complexities
  - At least one multi-lane road or an Autobahn section
  - At least one roundabout
  - At least one section with two roads crossing each other on different levels (e.g., an underpass)
  - At least one bikeway

The test subject was recruited from staff of the Bonn-Rhein-Sieg University of Applied Sciences. Therefore, a part of the city of Sankt Augustin, Germany, was selected that fit the listed criteria and that is situated near the campus of the university. Figure C.4 shows the selected part including the selected areas of interest as well as an overlay of the generated network on top of the map. Additional information about the scenario is listed in Table C.1.

The test subject documented that the course of the generated road network generally fits the map data well, which can also be seen in Figure C.4 (b). However, by directing the examiner's attention to certain details, some issues were documented as well. Minor

issues, like missing side roads, could be fixed by providing a more detailed mapping during the generation of the OpenDRIVE® data. Other issues were missing turning lanes at intersections, which are often omitted in available input data to reduce complexity. To fix this problem, the missing information needs to be added manually. The major problem, documented by the test subject, were issues with connections between road elements, i.e., paths across junctions. The examiner found missing connectors, connections where none should be allowed, and incorrect connecting paths. Most of these problems could be traced to missing or incorrect input data and some are the result of the conversion process from OpenStreetMap to OpenDRIVE® data by the Trian3D Builder software. Some problems can be solved by configuring the conversion process more carefully, but others result from internal processes, which cannot be easily manipulated (e.g., implemented heuristics to complete missing information). The latter type of problems would require manual adjustments within the game engine editor.

Based on the recorded findings, several issues were identified and examined more closely. It was possible to resolve some of the issues, for others the cause could be identified but they were not fixed due to time constraints. Unfortunately, a few causes are only speculations because they are internal to the closed-source tools used in the workflow. While traffic can generally be simulated on generated networks with only minor adjustments, a completely plausible setup would require various corrections or improvements to the generation workflow. A more detailed documentation of the test subject's findings and the discussion thereof can be found in [IVC7].

C.4 Summary

As a solution to efficiently providing agents with knowledge about their surrounding road network, a graph-like representation was defined. To simulate agents in multiple networks, a workflow was devised to quickly generate representations from input data. The workflow was evaluated by generating networks for three scenarios and inspecting the fit of the result to the 3D geometry generated alongside the road network representation. A close to perfect fit could be achieved between network and geometry for all scenarios. Additionally, the fit of one network was checked against geometry generated by applying a different generation tool and against a manually built scene. The fit between the network and these two scenarios was not perfect but could be corrected with only minor adjustments. All networks were used to simulate traffic agents.

One of the biggest advantages of a network generation workflow is the ability to generate multiple and large networks in a short amount of time. This allows for quick iterations, building several test scenarios, and acquiring realistic network layouts. During the evaluation, generation times were recorded, and the time needed for manual setups were estimated from the average setup time of a single lane. Comparing both values for each scenario, time savings of up to months can be achieved for larger networks.

To evaluate the quality of generated networks compared to their real-world input data, an independent examiner was acquired to inspect the network and document the findings. Analyzing the findings showed that the generated networks generally represent reality well. However, closer inspection reveals several shortcomings due to inaccurate input data, non-

transparent heuristics in external tools, or incomplete mappings between input and output. Connections between roads are prevalently affected by these problems.

In conclusion, being able to automatically generate road networks as semantic world knowledge for traffic agents allows efficiently providing multiple test scenarios for simulations. The workflow also enables the generation of large networks within minutes and realistic networks by processing real-world data. However, results are highly dependent on the quality of the input data, which often contains errors or simplifications. Since the road network is the source for the agents' knowledge about all road related information, flawed networks will lead to unusual behavior. Correcting these shortcomings requires intuitive editing tools or an improved generation process. To minimize the adjustment effort, a validation process would be beneficial. However, manual inspection is generally infeasible and realizing an automatic validation process is complex.

About the Author

Born and raised in Jena, Germany, Sven Seele attended IGS "Grete Unrein", where he already developed a strong interest in the MINT subjects and especially computers. During this time, he also spent ten life-defining months abroad, attending Mansfield High School in Mansfield, Arkansas, USA, and graduating in 2002. After also graduating from IGS "Grete Unrein" in 2004, Sven began his studies of Computer Science at Hochschule Bonn-Rhein-Sieg in Sankt Augustin in 2005. Mostly due to his interest in video games, he decided early on to major in media informatics with a focus on virtual reality, image synthesis, and image processing. Always looking out for opportunities to apply his knowledge to new and exciting domains, he took student jobs at the Institute of Visual Computing (IVC) of Hochschule Bonn-Rhein-Sieg, the Fraunhofer Institute IAIS.VE, Wirtgen GmbH, and the Department of Computer Science and Applied Cognitive Science of the University Duisburg-Essen.

During his studies, his interest in virtual environments and virtual characters grew, and his project work enabled him to write his Bachelor's thesis about "Gesture Control in Virtual Environments" at the Fraunhofer Institute in Sankt Augustin and his Master's thesis about "Stereoscopic 3D Games for Movie Theater Audiences" at the University Duisburg-Essen. After completing the graduate program in 2011, he continued his studies of virtual environments in three research projects at the IVC. In the context of his project work, he started his research for this doctoral thesis project in 2013.

In 2017, when an opportunity presented itself to apply his knowledge and skills in an industry setting, Sven decided to seize it and start a full-time position with SICAT as a software engineer working on 3D visualization topics and getting the technical lead role in the I-SLEEP research project. After several adjustments caused by the onset of the Covid pandemic, in August 2021, Sven decided to take on new challenges as a senior software engineer with ELISE (now Synera), an ambitious startup from Bremen. While excitedly helping to revolutionize the tools for dental surgery at SICAT and for connected engineering at Synera, Sven always kept pushing to also finish his thesis project.