Visualizing the behavior of CBR agents in an FPS Scenario

Philipp Yasrebi-Soppa[1], Jobst-Julius Bartels[1], Sebastian Viefhaus[1], Pascal Reuss[1,2], and Klaus-Dieter Althoff[1,2]

[1] University of Hildesheim
Samelsonplatz 1
31141 Hildesheim
{reusspa, bartelsj, viefhaus, yasrebis}@uni-hildesheim.de
[2] German Research Center for Artificial Intelligence (DFKI)
Trippstadter Str. 122
67663 Kaiserslautern
kalthoff@dfki.uni-kl.de

Abstract. The analysis and visualization of agent behavior enables a developer to identify unexpected or faulty behaviors and can show room of improvement. Therefore, visualization tools can be helpful to analyze behaviors during or after simulations. This paper presents a visualization tool VISAB developed for analyzing and visualizing the movement and actions of CBR agents in a first-person scenario. We describe the settings of the scenario and in more detail the visualization possibilities to get a better understanding of agent behavior during game-play.

Keywords: Visualization · Case-Based Reasoning · Agent behavior · Multi-Agent System · First-Person Scenario

1 Introduction and motivation

Visualization of agent behaviors can also be helpful for teaching Artificial Intelligence (AI). It can be used to support students and inexperienced AI developers during knowledge modeling and designing decision making processes for agents. The goal of this work was to create a visualization tool to display agent behavior in a first person scenario. This scenario is part of a platform for teaching Case-based Reasoning (CBR), learning software agents and multi-agent systems (MAS) during a advanced programming practical. The basic idea of this platform is to have several modules that represent different scenarios that can be played and solved by software agents. The students design and implement an agent or a team of agents with a CBR system (or other AI technologies that enables decision making) for one of the given scenarios. The implemented agent then plays the scenario and the behavior of the agent will be analyzed and visualized to

give feedback to the students. The students should be able to watch their agent while playing and get an evaluation after completing the scenario. In addition to play a scenario solo, it will be also possible to compete with the "home team" of agents in directly competitive scenarios. Figure 1 givens an overview of the desired architecture of the teaching platform.

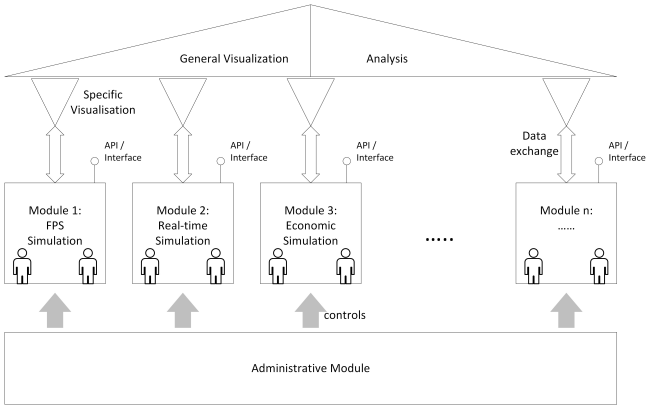

Fig. 1. Overview of the planned platform architecture

Currently, several modules are being implemented and in different states of completion: a first-person game, a real-time strategy game, an economic simulation, and several board game implementations like Settlers of Catan. In addition to these playable scenarios, an administration module and a first visualization module for the first-person game are under development. The visualization component records a game and enables the user to view the game later with several options to configure the displayed information.

The visualization of agent behavior in given environments with defined surrounding conditions enables a developer to identify unexpected or faulty behaviors and can show points of improvement. In addition to visualizing the behavior of the agent itself, background and contextual information can be visualized, too. Using this additional information, agent behaviors and decisions can be analyzed and the decision-making process of an agent will become more comprehensible to the developer and can be better optimized. [1][5][9]

This paper provides an overview of the visualization module and the implemented features. The remaining paper is structured as follows. Section 2 gives an overview of some related work in the field of agent behavior visualization. Section 3 describes the first-person scenario (FPS) and then in detail the visualization module for the participating agents, while Section 4 provides an overview of the performed evaluation. The paper concludes in Section 5 with a short summary and an outlook to future work.

2 Related Work

Over the last decades many research activities and approaches for modeling and visualizing the behavior of software agents in different use cases were realized. To prove the functionality of an artificial intelligence, multi-agent systems became a popular application area. We focus on the FPS domain, a sub-genre of action video games. In a typical FPS game, there are two teams of typically human players trying to overcome the opposing team by either eliminating each member of the opposing team or by successfully complete another objective, such as planting a bomb at a certain place or by preventing the opposing team to do so. While both teams are actively playing at the same time, most FPS are limited by a round-time of approximately five minutes and a limited map size.

Agent modeling frameworks like NetLogo[25], REPAST[11], or Pogamut[5] provide simulation and visualization capabilities beside their core design and modeling features. NetLogo was developed in 1999 and is still an active multi-agent environment today. It enables a developer to simulate complex situations with several thousand of agents in 2D or 3D. In addition, NetLogo offers to visualize several background information and to analyze the agent models with the help of different diagrams. An interesting feature of NetLogo comes with the extension HubNet. It enables the development and execution of participatory simulation for lectures. In these simulations every participating student controls a part of the overall system, for example a traffic light in a traffic simulation. [26][23]

The Recursive Porous Agent Simulation Toolkit (REPAST) is an open source framework for modeling and simulation agents. Similar to NetLogo it offers the modeling and simulation of agents in 2D and 3D environments and several visualization features with diagrams and graphs.[11] REPAST was developed further into two different version, REPAST Simphony and REPAST for high performance computing (REPAST HPC). REPAST Simphony offers additional visualization capabilities with the help of a geographic information system to visualize the movement of agents.[2][16]

Pogamut is another free development environment for modeling and simulation agent behavior in a 3D environment. It is used mainly in combination with the Unreal game series and is designed to support research and education. The current version of Pogamut was released in 2015[18]. During the debugging of implemented agents, Pogamut allows the visualization of several information, for example the position of agents on a map, the view direction of agents, and the planned movement.[6][7] In addition to development frameworks with visualization capabilities, there are several more or less pure visualization tools, that a designed with the main purpose to test and evaluate the behavior of software agents.

GameBots is a virtual testing environment to evaluate software agents in games. The goal was to develop a tool that can be used on computer games to enable the use of these games for research and education in the fields of AI and MAS. The GameBots tool provides three components to visualize background information about agents and their environment: a 3D virtual world, a global

VizClient for analyzing the entire simulation, and a local VizClient for analyzing the behavior of a single agent. With these three components, GameBots is able to visualize positions, view directions and field, movement, points, messages, and decision of the individual agents and agent teams.[1][20]

The Unreal Tournament Semi-Automated Force (UTSAF) is a military-based agent simulation in the 3D environment of Unreal Tournament. In the context of UTSAF, an agent can be a ground or an air unit. UTSAF uses the tool GameBots in combination with special information brokers to visualize agent behavior. These information broker modules can collect the information about individual agents, agent teams, or the entire environment and passes them to different instances of the GameBots tool. This enables the user to act as a spectator for the simulation and get an overview of the entire environment or see the simulation through the eyes of an individual agent.[12][17]

Lithium is a tool that was developed to enable analyses in multiplayer computer games. The tool visualizes information via overlays on top of the running game. Lithium was designed to analyze the entire situation of a computer game, and not specific information about a single agent. The goal is to capture the entire dynamic of the computer game. Lithium can visualize information on a local and global level. Local level information is based on specific positions or parts of the environment or specific events, while the global level is used to visualize trends or behaviors over time. The tool provides information about the position and movement of agents, combat behavior, and agent views on the local level. On the global level, Lithium can display information about the agent density on a specific part of the map, the medicine density and needs, control areas for specific teams, and combat information like fire support ranges.[9]

HeapCraft is a free tool for visualization and data search with a focus on the analysis of agent behavior in interactive virtual worlds. The tool can be used by administrators and players of multiplayer game servers and aims at changing a player‘s behavior in positive and social way. In addition, problems in the game world and with player activities can be identified and failure diagnoses can be performed.[13] A prominent game HeapCraft is applied to, is the 3D computer game Minecraft, but can be applied to various games with virtual worlds. The tool provides several components that can be integrated into a game as plug-ins. The Epilog Dashboard provides visualizations and analyses about player behavior in real time. In context of Minecraft the dashboard visualizes for example the online time, the covered distance, build blocks, and gathered resources. In addition, the dashboard computes an index for collaboration with other players. With the help of the Map Miner player activities can be tracked, analyzed, and visualized on a 2D map. The plug-in Classify is able to analyze the behavior of one player over a complete day and visualizes the information in form of diagrams. [14][15]

The Visualization Toolkit for Agents (VISTA) is a framework to visualize the internal reasoning processes of software agents. It aims at evaluating the behavior and decision making process of agents and can be used during or after a simulation.[21] VISTA was developed with four purposes: providing a generic

framework capable to be used in as many agent architectures and systems as possible, providing a domain-independent framework, enable the tracking of internal reasoning processes, and provide visualizations of agent behaviors during run-time as well as after the termination of a simulation by recording the behaviors. For the visualization of the internal reasoning processes, VISTA uses a so-called Situational Awareness Panel (SAP) to collect and display all information about the agents, their interactions, and communications. In addition, VISTA is capable of generating explanations for the behavior of agents with focus on objects and situations.[21][22]

3 Visualization of CBR agent behavior

This section describes first briefly the developed game with the FPS scenario. A more detailed description can be found in [8] and [19]. Then we will describe in more detail the conceptual idea of the visualization component called VISAB (Visualization of Agent behavior).

3.1 Settings of the FPS scenario

The FPS scenario was developed as a multi-agent system and consists of three components: the multi-agent system itself, the game logic and visualization component, and a CBR component. The game component was developed with Unity 3D[24], while the multi-agent system was implemented using Boris.Net[4]. The CBR component was modeled and implemented using the open source tool myCBR[3]. The Unity 3D framework was used to design the environment in which the software agents compete each other and to visualize the actions of the agents to the user. The agents are implemented within the Unity 3D component with the help of Boris.Net. There are three different agents implemented: the player agent, the planning agent, and the communication agent. The *player agent* gets an update of the situation through the Unity framework. With each sensor update, the *player agent* sends the information to the *communication agent*. This agent transforms the received data into a JSON string and passes it to the CBR component. The CBR component performs a retrieval and answers the request of the communication agent by sending the most similar cases back, also in a JSON string. The solution of the cases contain possible plans that can be executed by the *player agent*, but have to be transformed into Unity 3D specific orders to be executable. Therefore, the retrieved solutions are passed to the *planning agent*. This agent evaluates the received solutions and forms a plan that is passed to the *player agent*. The *player agent* then executes the new plan. If no new plan can be formed, the *player agent* continues executing the current plan.

The case structure modeled within the CBR component consists of a situation description based on the sensor input of the *player agent* and associated actions that form an executable plan. The situation description consists of seventeen attributes with various data types. The attributes have integer, symbolic, or Boolean data types. The distance to an entity in the game is represented by four

symbolic values: near, middle, far, and unknown. This way the similarity measure is less complex. Is the distance to an entity less than 15 Unity scale units, the distance is considered near, between 15 and 30 scale units the distance is set to middle, and between 30 and 50 scale units the distance is set to far. If the distance is greater than 50 scale units or the position of an entity is unknown, the distance is set to unknown. The developed FPS scenario was extended with a reinforcement learning (RL) approach to enable the CBR component to learn from different situations and the executed plans. Therefore, a reward function is used to calculate the win probability of a situation based on several important attributes. The results showed that the integration of an RL approach into the system leads to an improvement in the performance of the CBR agent. More details on the multi-agent system, the individual agents, their relationships, and the reinforcement learning approach can be found in [10].

3.2 Visualization tool for the FPS scenario

The analyses of the CBR agents performance were made by logging information about victory and defeat in a CSV file and by observing the live game-play. But this kind of analysis is not very sufficient and incorporates only few available information. Especially during observation in the live game-play, important actions can be missed. Therefore, a visualization component was required that was capable of collecting all relevant information during the game-play, aggregating and evaluating these information, and display them to the developer to understand the behavior of the agents. Many existing visualization tools were reviewed to find a suitable and applicable tool or framework that could be used in our use case. But none of the reviewed tools fit complete for our application and the desired platform. Especially the requirement of visualizing CBR specific information and the planned application of the visualization component to all planned modules of our platform could not be covered by the existing tools from our point of view. Some tools could be used only partially, other are not actively developed anymore. This led to the development of our own solution VISAB.

The architecture of VISAB is a classic three-layered model. The presentation layer is the interface to the user and contains the visual and graphical components to display the information about the agents. The logic layer is arranged under the presentation layer and is responsible for data processing and the preparation of information for the visualization. The components of the presentation layer are controlled and managed by the logic layer. In addition, the logic layer also enables the processing of user interactions and information requests. The undermost layer is the data layer that provides the data streamed from the game during live game-play or from a recorded file. The data layer loads and saves the data from text files with a format based on keys and values. It is the connection to the Unity game platform. Figure 2 gives an overview of the VISAB architecture.

VISAB requires the game data in a JSON similar format. This game data is stored in a text file and contains eighteen different properties. The value of each

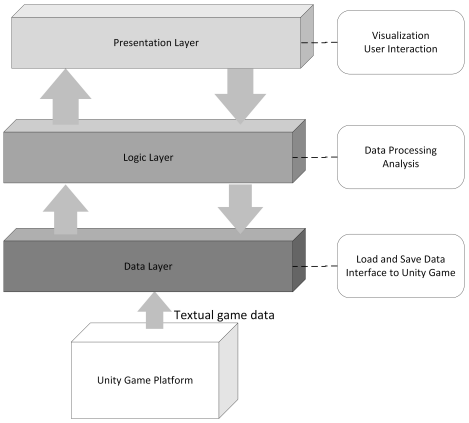

Fig. 2. Overview of the VISAB architecture

property is stored for each frame of a game cycle. The properties are listed in the following:

- **ammoPosition** - the positions of ammunition crates
- **coordinatesCBRBot** - the coordinates of the CBR agent
- **coordinatesScriptBot** - the coordinates of the scripted agent
- **healthCBRBot** - the current health of the CBR agent
- **healthScriptBot** - the current health of the scripted agent
- **healthPosition** - the positions of health containers
- **nameCBRBot** - the name of the CBR agent
- **nameScriptBot** - the name of the scripted agent
- **planCBRBot** - the current plan of the CBR agent
- **planScriptBot** - the current plan of the scripted agent
- **roundCounter** - a counter for the current round
- **statisticCBRBot** - victories and defeats of the CBR agent
- **statisticScriptBot** - victories and defeats of the scripted agent
- **weaponCBRBot** - current weapon of the CBR bot
- **weaponScriptedBot** - current weapon of the scripted agent
- **weaponMagAmmoCBRBot** - current ammunition for the equipped weapon of the CBR agent
- **weaponMagAmmoScriptedBot** - current ammunition for the equipped weapon of the scripted agent
- **weaponPosition** - the positions of weapons

A VISAB file can contain any type of information if it has the format *[key = value]*, for example *[weaponCBRBot = Pistol]*. The generic statistics visualization component of VISAB displays any information in the given format. In addition to a VISAB file, the game data can also be provided during live game-play using a data stream from the Unity game platform.

The presentation layer has two main perspectives to display the visualizations based on the data. The first one is a generic statistics overview of the provided

data. The properties and their values are displayed in a table to give the user overview of the available data. In addition, for each agent a diagram with the executed plans for each agent is displayed using the processed data. The diagram shows how often a specific plan is executed during game-play as well as the total number of all executed plans. Figure 3 shows the statistic perspective of VISAB with some sample data.

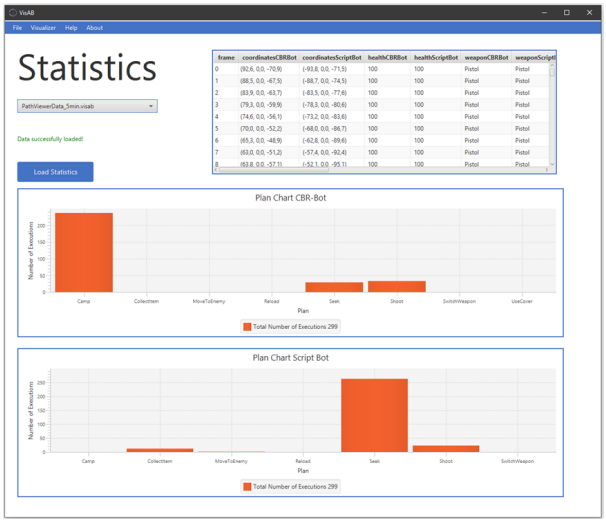

Fig. 3. Statistic perspective of VISAB

The second perspective is the so-called PathViewer and the heart of VISAB. It allows a detailed visualization of a game with focus on different aspects of the game and agent behaviors. The PathViewer consists of several elements: a map of the level, where the game took place, two tables with the detail information of the playing agents, a configuration panel, and a time frame. In the map, all information of the game can be visualized by symbols and paths and therefore displays a replay of the game from a bird's eye perspective. The configuration panel allows the user to enable or disable the visualization of certain information and serves also as a legend for the displayed information. The time frame is used to control the playback of the recorded game. This can be done by playing the complete game or by selecting a specific frame that should be displayed. Figure 4 shows the PathViewer perspective with sample data from a fifteen minute game-play at the beginning of the visualization.

To make use the PathViewer, at first a User has to select a data file or a live stream that should be visualized (1). Then the data is loaded (2) and displayed in the two tables (3). In the next step, the user selects the information to be visualized or accept the default configuration (4). The information will be displayed in the map (5) after starting the visualization (6). During the

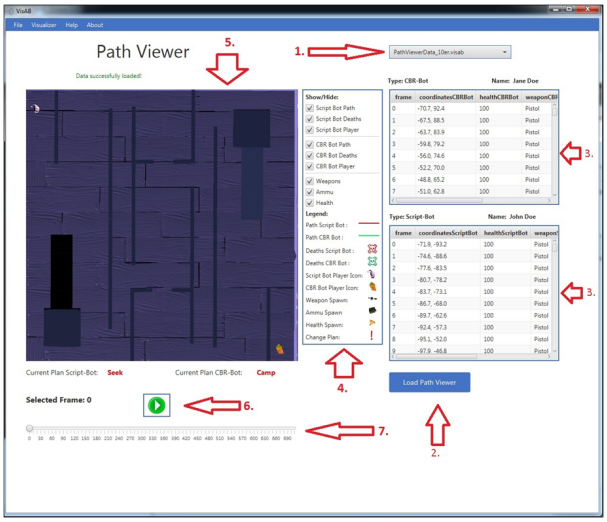

Fig. 4. PathViewer perspective of VISAB at the start of the visualization

visualization, the user can always see the current displayed frame or select a specific frame (7). The single steps are marked in Figure 4.

During the visualization, different information can be found on the map. First the path of the playing agents will be displayed to retrace the routes during a game session. Along the paths several icons can be found. The current position of the agents is also displayed along the routes for every frame. This allows to see the detailed movement on the routes during playback. Every time an agent is defeated a symbol is placed on the map to visualize the situation. In addition, every time a weapon, ammunition, or health is spawning, the corresponding icon is displayed on the map. If it is collected, the icon disappears. At least, another important information can be visualized: the point during a game, when an agent changes his action plan. This is displayed using an exclamation mark to make the specific situation visible. Using the visualized information on the map and the detail information about the agent status and behavior in the two tables, a user can retrace and analyze agent behavior and can identify situations with bad or even wrong agent behavior. This allows a better adaptation of agent behavior than just viewing the live game without the contextual information or just using the logged CSV file. Figure 5 shows the PathViewer with several information visualized in the map.

4 Evaluation

The evaluation of VISAB was performed to test the correct and complete visualization of the analyzed information. In addition, the performance while displaying different kinds of information were tested. To verify the correctness and

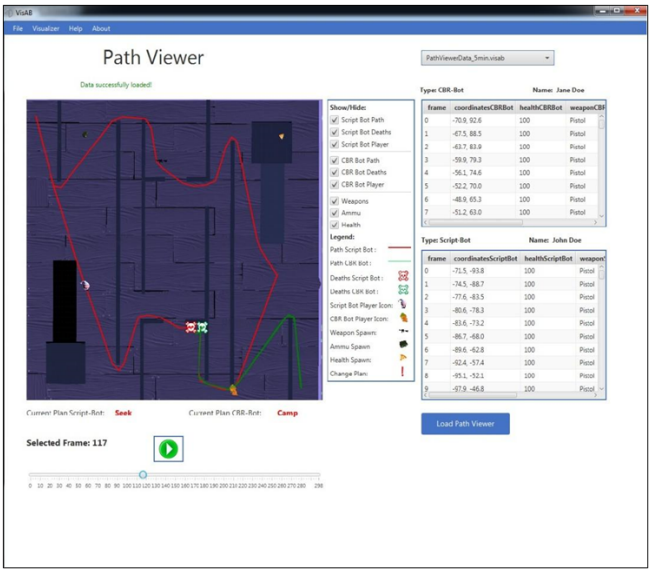

Fig. 5. PathViewer perspective of VISAB with visualized information on the map

completeness of the displayed information, a live game was directly compared with the visualized information in VISAB. The game-play was recorded by using a the ReLive function of a RADEON graphic card. This recorded game-play was then compared with the VISAB visualization, frame by frame. For the verification three game-play with a length of ten, twenty, and thirty minutes were recorded and compared. The result was that every occurred entity and situation during the game-play was recorded complete and correctly. The performance of VISAB was evaluated to test the handling of big data sets recorded during long game-play sessions. This was done with two data sets for a game-play of 90 minutes. The tests showed only minimal delays when selecting a specific frame with the slider. Therefore, the performance is more than sufficient for the planned use cases.

An evaluation within lectures with participating students is planned for the next semester. The goal will be to evaluate the impact of VISAB on the analysis and development process of the students.

5 Conclusion and Outlook

This paper presents a visualization tool for agent behavior in an FPS scenario to analyze and identify bad or wrong behavior of the participating agents. We described the settings of the FPS scenario and then the idea, architecture, and current realization of the visualization tool VISAB. We also have conducted a small successful evaluation on the features of VISAB. The next steps for developing VISAB further, are to evaluate VISAB with students and extend the

PathViewer with more information of the internal reasoning processes, like case similarities. We are also currently re-implementing the FPS scenario for team play and other game modes like capture the flag. Therefore, we will also adapt the PathViewer to display the additional information generated by the new situations. Adding new perspectives to enable the visualization of the other modules are also planned.

References

1. Adobbati, R.; Marshall, A. N.; Scholer, A.; Tejada, S.; Kaminka, G.; Schaffer, S.; Sollitto, C.: GameBots: A 3D Virtual Test-Bed for Multi-Agent Research. In: WAGNER, T. (Ed.); Rana, O.S. (Ed.): Proceedings of the second international workshop on Infrastructure for Agents, MAS, and Scalable MAS, Volume 5, Montreal, Canada, 2001.
2. Argonne National Labatory, Webseite 2015. - https://sourceforge.net/projects /repast/; last verification: 16.06.2020
3. Bach, K.; Sauer, C.; Althoff, K.-D.; Roth-Berghofer, T.:Knowledge Modeling with the Open Source Tool myCBR. In:Nalepa, G.J. (Ed.), Baumeister, J. (Ed.), Kaczor, K. (ed.): Proceedings of the 10th Workshop on Knowledge Engineering and Software Engineering (KESE10), Praque, Czech Republik, 2014
4. Bojarpour, A.: Boris.Net, Website, 2009 - http://www.alibojar.com/boris-net, last verification: 16.06.2020.
5. Gemrot, J.; Kadlec, R.; Bida, M.; Burkert, O.; Pibil, R.; Havlicek, J.; Zemak, L; Simlovic, J; Vansa, R; Stolba, M.; Plch, T.; Brom, C.: Pogamut 3 can assist developers in building AI (not only) for their videogame agents. In: Dignum, F. (Ed.); Bradshaw, J. (Ed.); Silverman, B. (Ed.); Doesburg, W. (Ed.): Agents for games and simulations, Springer-Verlag Berlin, Heidelberg, 2009, S. 1-15.
6. Gemrot, J.; Brom, C.; Kadlec, R.; Bida, M.; Burkert, O.; Zemcak, M.; PibilL, R.; Plch, T.: Pogamut 3–Virtual humans made simple. In: Srini-Vasan, N. (Ed.); Gupta, A. K. (Ed.); Pandey, J. (Ed.): Advances in Cognitive Science, SAGE Publications Pvt. Ltd, 2010, S. 211-243.
7. Gemort, J.; Brom, C.; Plch, T.: A periphery of pogamut: From bots to agents and back again. In: Dignum, F. (Ed.): Agents for games and simulations II, Springer-Verlag Berlin, Heidelberg, 2011, S. 19-37.
8. Hillmann, J.: "Konzeption und Entwicklung eines Prototypen für ein lernfähiges Multi-Agenten-System mittels des fallbasierten Schließen im Szenario einer First-Person Perspektive" (Conception and Development of a prototype for a multi-agent-system with learning capabilities using case-based reasoning in the first-person perspective szenario). Hildesheim, University of Hildesheim, 2017.
9. Hoobler, N.; Humphreys, G.; Agrawala, M.: Visualizing competitive behaviors in multi-user virtual environments. In: Visualization, IEEE, 2004, S. 163-170.
10. Kolbe, M.; Reuss, P.; Schoenborn, J.M.; Althoff, K.-D.: Conceptualization and Implementation of a Reinforcement Learning Approach Using a Case-Based Reasoning Agent in a FPS Scenario, In: Jaeschke, R. (Ed.); Weidlich, M.(Ed.): Proceedings of the Conference "Lernen, Wissen, Daten, Analysen", Berlin, 2019
11. Macal, C. M.; North, M. J.: Tutorial on agent-based modeling and simulation. In: Simulation Conference, Proceedings of the Winter, 2005, S. 73-83.
12. Manojlovich, J.; Prasithsangaree, P.; Hughes, P.; Chen, J.; Lewis, M.: UTSAF: A Multi-Agent based Framework for Supporting Military-based distributed interactive

simulations in 3D virtual environment. In: Simulation Conference, Proceedings of the Winter, Volume 1, 2003, S. 960-968.
13. Müller, S.; Solenthaler, B.; Kapadia, M.; Frey, S., Klingler, S.; Mann, R. P.; Summer, R. W.; Gross, M.: HeapCraft: Interactive Data Exploration and Visualization Tools for Understanding and Influencing Player Behavior in Minecraft. In: Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games, 2015, S. 237-241.
14. Müller, S.; Kapadia, M.; Frey, S., Klingler, S.; Mann, R. P, Solenthaler, B.: Statistical Analysis of Player Behavior in Minecraft. In: Zagal, J. (Ed.): Proceedings of Conference on Foundations of Digital Games, 2015.
15. Müller, S.; Frey, S.; Kapadia, M.; Klingler, S.; Mann, R. P.; Solenthaler, B; Summer, R. W.; Gross, M.: HeapCraft: Quantifying and Predicting Collaboration in Minecraft. In: Proceedings of the eleventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 2015, S. 156-162.
16. North, M. J.; Collier, N. T.; Ozik, J.; Macal, C. M.; Tatara, E. R.; Bragen, M.; Sydelko, P.: Complex adaptive systems modeling with Repast Sim-phony. In: Complex adaptive systems modeling, Volume 1, 2013.
17. Prasithsangaree, P.; Manojlovich, J.; Hughes, S.; Lewis, M.: UTSAF: A Multi-Agent-Based Software Bridge for Interoperability between Distributed Military and Commercial Gaming Simulation. In: The Society for Modeling and Simulation International, 2004, S. 647-657.
18. Pogamut, Webseite 2015. - http://pogamut.cuni.cz/main/tiki-view_blog.php?blogId=3; last verification: 16.06.2020.
19. Reuss, P., Hillmann, J., Viefhaus, S., Althoff, K.-D.: "Case-Based Action Planning in a First Person Scenario Game". In: Rainer Gemulla, Simone Ponzetto, Christian Bizer, Margret Keuper, Heiner Stuckenschmidt (Ed.). LWDA 2018 - Lernen, Wissen, Daten, Analysen - Workshop Proceedings. GI-Workshop-Tage "Lernen, Wissen, Daten, Analysen" (LWDA-2018) August 22-24 Mannheim Germany University of Mannheim 8/2018.
20. SIOUTIS, C.: Reasoning and Learning for intelligent Agents. Phd Thesis, University of South Australia, 2006.
21. Taylor, G. E.; Jones, R. M.; Fredriksen, R. A.: VISTA: A Generic Toolkit for Visualizing Agent Behavior. In: Proceedings of the 11th Conference on Computer Generated Forces and Behavioral Representation, 2002, S. 157-167.
22. Taylor, G. E.; Knudsen, K.; Holt, L. S.: Explaining Agent Behavior. In: Behavior Representation in Modeling and Simulation (BRIMS), 2006.
23. Tisue, S.; Wilensky, U.: NetLogo: Design and implementation of a Multi-Agent Modeling Environment. In: Proceedings of agent, Volume 2004, S. 7-9.
24. Unity Technologies: Unity 3D Overview, Website, 2018 - https://unity3d.com/de/public-relations, last verification: 16.06.2020.
25. Wilensky, U.: NetLogo Frogger model. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, Webseite 2002. - http://ccl.northwestern.edu/netlogo/models/Frogger; last verification: 16.06.2020.
26. Wilensky, U.: NetLogo. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, Webseite 2017. - http://ccl.northwestern.edu/netlogo/; last verification: 16.06.2020.