Using Gaming Environments to Teach the Idea and Application of CBR

Pascal Reuss[1,2], Klaus-Dieter Althoff[1,2]

[1]German Research Center for Artificial Intelligence, Kaiserslautern, Germany
[2]Institute of Computer Science, Intelligent Information Systems Lab, University of Hildesheim, Hildesheim, Germany

Abstract

Bringing the idea of artificial intelligence (AI) methods to students can sometimes be a challenging task, especially with focus on motivation, simplicity, and understanding. This paper describes a platform in development that uses several modules with different gaming scenarios like a First-Person scenario or the board game Settlers of Catan in which Case-based Reasoning (CBR) can be used to solve the given problems and win the game. The idea behind this platform is to make the task of teaching and learning the idea of software agents, CBR, and their application to given problems more flexible and interesting for both, students and lecturers. We describe the general architecture of the platform and take a closer look on existing modules with their scenarios and CBR solutions. In addition, we present a visualization module to support the traceability of decisions made by the AI.

Keywords

Case-Based Reasoning, Software Agents, Computer Games, Teaching AI

1. Introduction

Artificial intelligence (AI) and especially Case-based Reasoning (CBR) is used in games and gaming environments in various ways in the last decades. Most of the approaches are used to prove that AI in various forms is able to play a game better than a human. There are two core objectives for the use of AI are either to play the game well or to play it believably or human-like. [1][2] In addition, AI can be used to control player characters or non-player characters. Three control layers can be distinguished: movement control, decision control, and strategy control. [3] An AI trying to play well as a player character is used for optimizing games with respect to the performance of play, debugging player experience or realistic game play. This can be used for automatic game testing and for evaluation of game design. AI playing as non-player characters is usually used for difficulty adjustment and game balancing mechanism to personalize and enhance the experience for players. But AI is not only beneficial for games, games are beneficial for AI, too. There are several reasons why games are very good domains for studying AI. Among these reasons are the aspects that games in most cases represent hard and interesting problems, have a rich human-machine interaction, and can support the research of long-term AI goals like social intelligence and creativity. In addition, games are very popular and this popularity brings several benefits like more content and data to play and train with. Moreover, different types of

LWDA'21: Lernen, Wissen, Daten, Analysen September 01–03, 2021, Munich, Germany
pascal.reuss@dfki.de (P. Reuss); klaus-dieter.althoff@dfki.de (K. Althoff)

games represent challenges for different AI areas, like neuronal nets, tree search, knowledge representation, and natural language processing. [4] [3] The second aspect, that games are useful for testing and learning AI was the starting point for the idea of a platform for learning and teaching CBR to students in context of a enhances programming practical. In Section 2 the idea and architecture of the intended platform is described. In several subsection (2.1, 2.2, and 2.3) different game modules are presented and associated initial CBR approaches. In addition, we present a visualization module in Section 2.4 and give an overview of the platform impact in Section 3. The paper ends with a conclusion and an outlook to future steps.

2. A platform for teaching and learning CBR

The platform is intended to teach the topics CBR, learning software agents, and multi-agent systems (MAS) during an advanced programming practical course. The students usually attend several theoretical lectures on these topics before taking part in the practical lecture and therefore the programming lecture is used to show the students how the theoretical foundations and the practical realization of software agents and CBR systems are connected. The students design and implement an agent or a team of agents with a CBR system (or later other AI technologies as well) for one of the given scenarios. Depending on the scenario and the students study goal (Bachelor or Master), a single agent or an agent team, which will be able to play the game, has to be developed. There are two main learning goals for the students:

1. The design and implementation of a working knowledge model for a CBR system and
2. the design and implementation of interacting and decision-making software agents.

The students agent or agent team will compete either against the game itself, for example while playing a game like Tower Defense (see Section 2.5) or against other software agents with own knowledge models and decision-making algorithms, for example in a First-Person scenario (FPS). The opposite agent team can be a team built by other students or the "home team", developed by the lecturer. When the platform is fully developed, a tournament between several student-built agent teams would also be possible.

The platform consists of several game modules, a visualization module, and an administration module. Currently, a FPS module, a board game module, and a cooperative card game module are implemented as well as a visualization module. All modules will be developed further. The modules were developed as stand-alone version on the first iteration and will be connected to the overall platform in the next months to be controlled by a platform administration module. Several other game modules are in different states of development, for example an economic simulation, a Tower Defense game, and a turn-based tactic space game. Each game scenario provides different challenges for a case-based agent and therefore for the students. Figure 1 gives an overview of the architecture of the teaching platform. In the following subsections the different modules or scenarios that are currently developed or under developing are described.

2.1. Module 1- First person scenario

The first realized module of the platform is a FPS game. The goal of the classic FPS game is to reach a certain amount of scores. During the fight an agent can lose life points. The

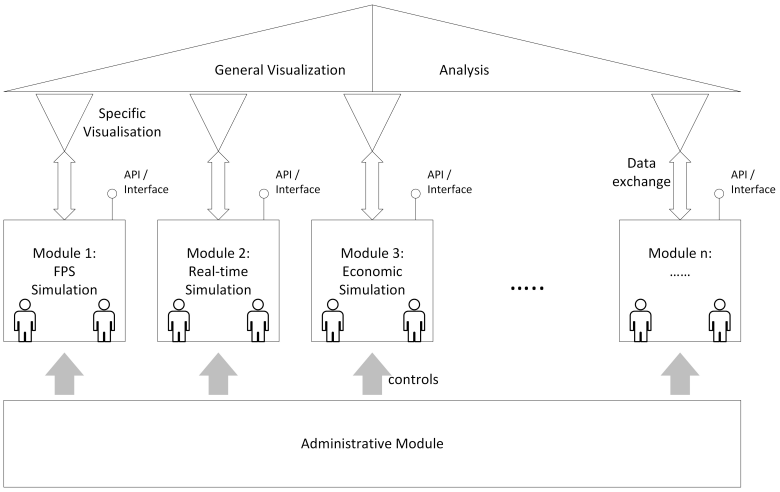

Figure 1: Overview of the platform architecture

player de-spawns as soon as the life points become zero and the opponent gains a point. The de-spawned player spawns again with full life points and the fight continues. The power-ups have also spawn points, where they appear. If a power-up is collected a certain amount of time passes by until the power-up spawns again. There may be several different weapons in an arena. These weapons usually differ in the amount of damage they deal to the life points of a player, the effective range, or the accuracy. More details about this form of FPS games can be found in [5]. The application of AI in FPS games have been researched in the last years. Auslander and his colleagues developed an approach called CBRetaliate. They combined CBR with Reinforcement Learning to find action strategies in the game Unreal Tournament [6]. Other approaches in the FPS genre use other technologies. Several approaches use imitation learning to improve the opponent AI by observing the player's actions in the game Quake 2 and Quake 3 [7][8].

Our chosen FPS scenario is also a combat scenario. Two software agents fight against each other in a small arena with obstacles that affect the field of view for the agents and can be used as cover. The goal is not to achieve a certain score to win, but to fight a given amount of time. The agent with the highest score at the end of the round is the winner. An agent gains a point, if he brings the life points of the other agent to zero and loses a point if he loses all his life points.

Every agent and the environment is built in Unity 3D [9] and Boris.Net [10] and thus implemented as a C# project. A player is realized as an agent team consisting of three different agents: Player Agent, Planning Agent, Communication Agent. The *Player Agent* receives information throughout its sensors from the game environment or from the proposed plans of the planning agent. When a new plan is required, the Player Agent sends a request to the *Communication Agent.* This agent is connected with the myCBR [11] component which uses JAVA as a programming language. Each time the Communication Agent receives a request, the agent formulates a request to the CBR system to retrieve the most similar case to the current situation. Once the most similar case has been retrieved, the proposed solution (which usually results in a proposed action) will be sent to the *Planning Agent.* This agent evaluates the proposed action from the Communication Agent and forms a plan according to the game actions, which will be sent to the

Player Agent and will be followed until a new plan will be proposed. As a simple example, the Player Agent perceives that he is low on health (e.g., $< 20\,\%$ hit points (HP)), so that perception will be transferred to the Communication Agent. This agent retrieves as the most similar case, that picking up a piece of pizza leads to gain back most of the lost hit points. Thus, the planning agent formulates a plan to find and to pick up a piece of pizza. This plan will be followed, until it has been picked up, until the agent dies, or until another plan gains a higher priority (e.g., self-defense or perceiving another, better, healing item).

The case-based agent team uses cases with a situation description and associated actions to plan its moves in the game level. The situation description was derived from the first person scenario and the basic assumptions made to the scenario . Several attributes are used to model the case structure for the environment including the available ammunition and health, the distances to the enemy and collectibles in the environment, and the last known position of the enemy. Some attribute values can be taken from the environment itself, while other are derived from the current values of attributes. These derived attributes are *isCoverNeeded*, *isEnemyAlive*, *isHealthNeeded*, and *isWeaponNeeded*. The value of these attributes depend on the state of the agent and the environment. For example, the attribute *isHealthNeeded* is set to true if the health points of the agent drop below 70%. The attributes use symbolic or Boolean data types and for all attributes similarity measures in the form of matrices have been implemented. More information about this scenario and the realized CBR agents can be found in [12] and [13].

2.2. Module 2 - Settlers of Catan

Another module of the teaching platform is based on the popular board game *The Settlers of Catan* [14]. This game is turn-based and therefore each player has theoretically unlimited time to plan the actions of their turn. In addition, this means that the environment will not change during an agent´s decision-making process. CBR for turn-based strategy games has been used in research for several years. For example, CBR was used in an open source Civilization clone for strategy planing [15][16], in the board game Monopoly to plan the actions of an agent based on past board states [17], and in the space game CORSAIR for strategic and tactical planning [18].

The main goal in the Settlers is to earn points by building streets, villages, and cities using a combination of five different resources (grain, wool, wood, iron, and clay). The placement of villages and cities is crucial, because a player gets resources from land tiles next to his buildings according to a dice roll. Therefore, the right placement to gather different resources is important. Additional game elements are resource trading between players, a thief to steal resources from other players and block land tiles to prevent resource generation, and several achievements like the longest connected road to get additional points. The first player achieving 10 points wins the game. Given the rules and actions of the game, many strategical and tactical decision points to challenge players and artificial agents can be found. In our module, the basic elements and rules of the board game were implemented: building streets, villages, and cities, gathering resources and trading resources between players. The more advanced game elements are not implemented in the first version yet, but about to come in the future.

The game environment was developed with Unity 3D, the two software agents with Boris.Net, and the CBR systems were developed using JAVA and myCBR. Each agent player has access to

a CBR system with four case bases with different case structures: the first case base is used in the beginning of a game when each player can place two villages at resources tiles. These first two moves are very important, because getting the right resources has a huge impact on the build and trade possibilities. The first case bases is used by the agents to determine where to place their first villages. The cases in this case base have the resources at building places and their probabilities based in the dice number as the problem description and a prioritized list of resources that the agent should try to get as a solution. Because the resource distribution is random in each game, these cases are built from general expert knowledge which resources are important at the beginning. The second case base is used to determine the next move of an agent during the game play. The problem description of these cases consists of information about the villages and cities of the player and the available resources, while the solution is a prioritized list of build items (street, village, or city) and the required resources. The agent tries to build the items on the list in descending order. If none of the items on the list can be built, the third case base is used to determine if a resource trade with the other player or the bank is possible. These cases contain the information about available resources and required resources for a specific building as the problem description and a trade offer as the solution. The cases are only used to trade with other players. If no trade with other players is possible or the demanded resources are higher than for a bank trade, the agent will trade with the bank using the defined game‘s trade ratio. If no trade is possible and no item can be built, the agent skips the turn. The last case base is used, when the thief takes action and a player has more than seven cards in its hand. Then half of these cards have to be discarded. The problem description for the according case base contains the available resources as the problem description and the cards to discard as the solution.

2.3. Module 3 - Card game Hanabi

A third module that was realized is a card game called Hanabi [19]. This game is a cooperative card game and a team of agents have to play together to beat the game. The agents have to play the cards sorted by color and number. The 50 cards in the game have numbers from one to five and five different colors: red, yellow, green, blue, and white. Each number exists twice for each color. Each color has its own stack and the goal is to put all cards on the according stack ordered by their number. The interesting point of this game is that an agent does not see his own cards, but only the cards of the other agents. Therefore, they have to communicate with each other to get hints which cards to play in their turn. One challenge is that there is a limited number of hints a player can give and get and a hint can be given for a color or a number. There are eight hint token in the game and every time a hint is given, one token is flipped. There are some situations where a hint token can be flipped and made available again. Having only three chances overall to play a correct card is another challenge. Each time an agent does not play a correct card, one chance is gone. Because an agent has to choose the card to be played without seeing his cards, it could happen, that the played card does not fit on one of the stacks. If all three chances are gone, the agents lose the game.[20]

The agent team for this game was implemented with two decision-making components: a rule-based component and a case-based component. The rule-based component checks the available information of the active player about his own cards and the cards of his teammates

and decides which action to take: play a card, discard one, or give a hint to a teammate. The priority of the actions is playing a card, give a hint, and at last discard, because discarding has the least benefit and will increase the probability of playing a false card. In most game situations, the rule-based component generates the decision about the action. But if an agent is in a situation where no hints are available and there are not enough information to play a safe card, then the case-based component takes over. For each card in the agent´s hand, a retrieval is performed to decide whether to play, keep, or discard the card.

A case represents a specific situation with focus on a cards number value. The problem descriptions contain seven attributes: The cards value as an integer, five Boolean values (false, true, and unknown) for the different colors, and the number of false cards played. The solution is the proposed action: play, discard, or keep. For each card in the agents hand a retrieval is performed. If the found solution is to play or discard the card, the retrieval process stops and the action is executed. If all retrievals propose the solution of keeping the card, then the rule-based component takes over again and the agent asks for a hint. Currently, the hints are rule-based using the amount of information about color or number. If the agent has less information about colors than number, a hint for a color is asked and vice versa. In the future, these hint questions will be supported by cases, too.

2.4. Visualization module

Visualization of agent behaviors can also be helpful for teaching AI. It can be used to support students and inexperienced AI developers during knowledge modeling and designing decision making processes for agents. The visualization module was first developed for the FPS module to analyze the CBR agents behavior after the game. Therefore, the visualization module 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 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. [21][22][23]

Many existing visualization tools were reviewed to find a suitable and applicable tool or framework that could be used in the use case of our intended platform. But none of the reviewed tools fits completely. Especially the requirement of visualizing CBR-specific information and the planned application of the visualization component to all existing and future game modules of our platform could not be covered by the existing tools from our point of view. This led to the development of our own solution, the module VISAB (**Vis**ualization of **A**gent **B**ehavior).

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. The logic layer is responsible for data processing and information preparation for the visualization. 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. VISAB requires the game data in a JSON similar format. This

game data is stored in a text file and contains eighteen different properties like the coordinates of the playing agents during each frame, statistics about victories and defeats, and the current plans of the agents. The value of each property is stored for each frame of a game cycle. The generic statistics visualization component of VISAB displays any information in the given format.

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. Figure 2 shows the PathViewer perspective with sample data from a five minute game-play.

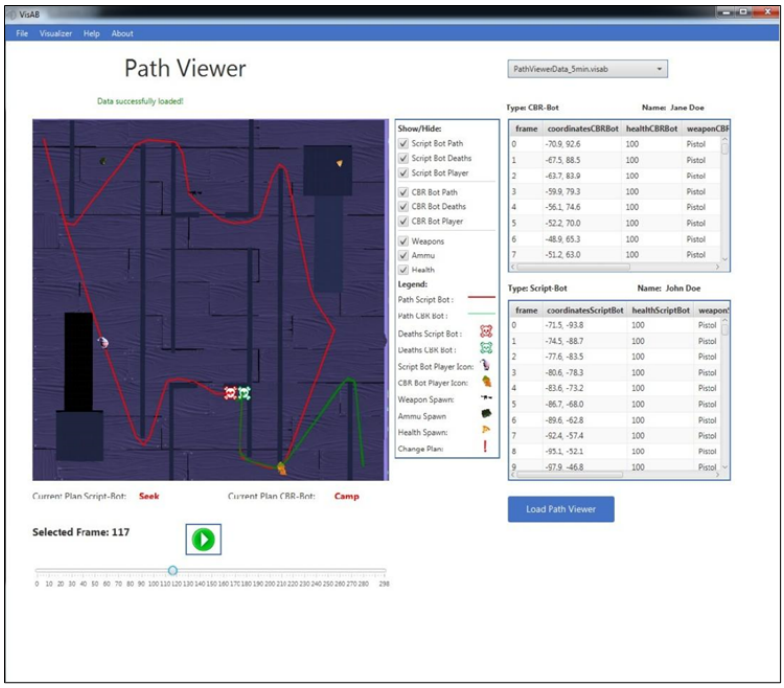

Figure 2: PathViewer perspective of VISAB with visualized information on the map

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. More detailed information on VISAB can be found in [24].

2.5. Additional modules

Besides the described modules, some other modules are currently under initial development. One module is based on the idea of tower defense. In this type of games, the player competes against the game itself fighting waves of enemies with build and upgrades defense towers. The player can take actions only between the enemy waves, while during the waves only the existing turrets can be used. Each defeated enemy gives some amount of money that can be used to purchase new towers or upgrade the existing one. In the current state three different turrets: a standard turret, an explosion turret, and a laser turret. Each turret can be upgraded once to increase the damage. The explosion turret has the ability to damage several enemies at the same time in a specific radius and the laser turret slows enemies in addition to the damage dealt. A purchased tower can be sold for a specific amount of money that is lower than the original purchase price. The CBR component will be used to decide which actions should be done during the build phase between enemy waves. As the problem description the types of enemies for the next wave, the currently built towers, their positions on the map, and the available money is used. The solution is a list of potential actions that will be useful and affordable.

Another module is a simple turn-based tactical space simulation, where two players gather resources, build and move different types of ships, research technologies, and fighting each other. The goal is to destroy the headquarter of the opposing player. Therefore, the players have to conquer planets to gather resources, research new technologies, and build ships to fight through a solar system with several planets. For each turn a specific amount of action points is available that can be used to one or more of the mentioned actions before. The CBR component in this game will be used to deliver a list of actions that can be done during the players turn. The actions are divided into four fields: resource mining, building, movement and fights, and research. For each of these fields individual case bases will be used with different case structure. During the retrieval all four case bases will be queried to get a list of possible actions and then a fifth case base will be used to select the most appropriate actions from the list.

Additional research on using CBR in real-time strategy games, especially in the game StarCraft 2, is conducted by another colleague in our lab. He uses a MAS with CBR components based on the SEASALT architecture to combine rule-based and case-based decision making for units and buildings [25]. While the game and the proposed possibilities for CBR research and learning would fit into the learning platform, the integration of StarCraft 2 as a module causes some problems. The game and its interfaces are set and therefore the communication interfaces with the administration and visualization modules have to built on top.

3. Platform impact

The introduction of the platform and the possibility for the students to use the existing gaming scenarios, even if not yet complete, had a great impact on the number of participating students over the last two years. In addition, the motivation of the students and the quality of the resulting solutions for the software agents and CBR systems have increased as well. The initial development of the platform started three years ago and the first version was used two years ago. Before introducing the platform and gamification into the programming practical course the number of students participating was stable between ten to twelve students starting the

course and six to eight students completing it. The grades of the completing students were mostly between "B" and "C". Since the introduction of the platform, the number of participating students increased as well as the given grades. The grades for the currently running course are not set. The following figures show the development of participants and their grades during the last five years.

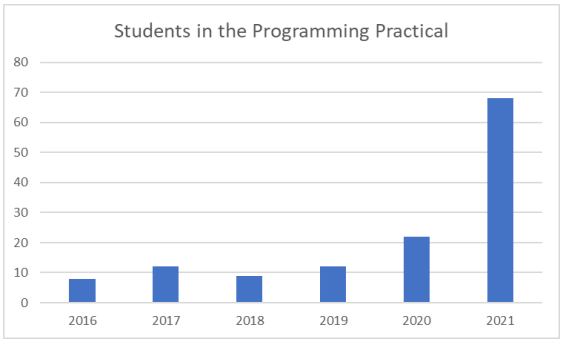

Figure 3: Development of participating students over the last years

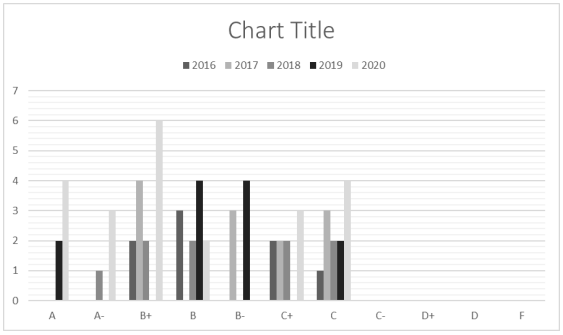

Figure 4: Development of the grades over the last years

To get more information about the students opinion on the learning platform and the gamification aspects an evaluation was planned for the current programming practical course, but had to be delayed because the deployment of the current platform state could not be performed as planned. The evaluation is now planned for the next turn of the programming practical course.

4. Conclusion and Outlook

This paper describes a learning and teaching platform for CBR being in development and partial use during a programming practical course for software agents and CBR. We give an overview of the platform idea and architecture and describe briefly several game modules with different scenarios that can be solved by participating students by designing and implementing individual agents and multi-agent systems with CBR systems. Currently only several modules are deployed individually. The next step is to finish the administration module for the overall platform and

deploy the platform with all implemented modules. In addition, the modules under development will be finished and all modules will be developed further. The goal for the next year is to have a fully deployed platform with six game modules to be challenge-able by the students and the visualization module coupled to all of these game modules to allow a traceable development of the software agents and the CBR systems.

References

[1] J. Togelius, Ai researchers, video games are your friends!, in: IJCCI, 2015.

[2] F. Safadi, R. Fonteneau, D. Ernst, Artificial intelligence in video games: Towards a unified framework 2015 (2015).

[3] I. Millington, J. Funge, Artificial Intelligence for Games, Second Edition, 2nd ed., Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2009.

[4] G. N. Yannakakis, J. Togelius, Artificial Intelligence and Games, Springer, 2018. http://gameaibook.org.

[5] J. E. Laird, It knows what you're going to do: Adding anticipation to a quakebot, 2000.

[6] B. Auslander, S. Lee-Urban, C. Hogg, H. Muñoz-Avila, Recognizing the enemy: Combining reinforcement learning with strategy selection using case-based reasoning, in: ECCBR, 2008.

[7] C. Thurau, C. Bauckhage, G. Sagerer, Combining self organizing maps and multilayer perceptrons to learn bot-behaviour for a commercial game., 2003, pp. 119–.

[8] S. Priesterjahn, O. Kramer, A. Weimer, A. Goebels, Evolution of human-competitive agents in modern computer games, in: 2006 IEEE International Conference on Evolutionary Computation, 2006, pp. 777–784. doi:10.1109/CEC.2006.1688390.

[9] UnityTechnologies, Unity 3d overview, 2021. URL: https://unity.com.

[10] A. Bojarpour, Boris.net, 2009. URL: http://www.alibojar.com/boris-net.

[11] K. Bach, C. Sauer, K.-D. Althoff, T. Roth-Berghofer, Knowledge modeling with the open source tool mycbr, in: G. J. Nalepa, J. Baumeister, K. Kaczor (Eds.), Proceedings of the 10th Workshop on Knowledge Engineering and Software Engineering (KESE10). Workshop on Knowledge Engineering and Software Engineering (KESE-2014), located at 21st European Conference on Artificial Intelligence, August 19-19, Prague, Czech Republic, CEUR Workshop Proceedings (http://ceur-ws.org/), 2014.

[12] M. Kolbe, P. Reuss, J. Schoenborn, K.-D. Althoff, Conceptualization and implementation of a reinforcement learning approach using a case-based reasoning agent in a fps scenario, in: CEUR LWDA 2019 Workshop Proceedings 2454, CEUR, 2019.

[13] P. Reuss, J. Hillmann, S. Viefhaus, K.-D. Althoff, Case-based action planning in a first person scenario game, in: R. Gemulla, S. Ponzetto, C. Bizer, M. Keuper, H. Stuckenschmidt (Eds.), LWDA 2018 - Lernen, Wissen, Daten, Analysen - Workshop Proceedings. GI-Workshop-Tage "Lernen, Wissen, Daten, Analysen" (LWDA-2018), August 22-24, Mannheim, Germany, Universität Mannheim, 2018.

[14] CatanGmbH, Settlers of catan overview, 2021. URL: https://www.catan.com/game/catan.

[15] R. Sánchez-Pelegrín, M. A. Gómez-Martín, B. Díaz-Agudo, A cbr module for a strategy

videogame, in: 1st Workshop on Computer Gaming and Simulation Environments, at 6th International Conference on Case-Based Reasoning (ICCBR, 2005, pp. 217–226.
[16] A. A. Sánchez-Ruiz, P. González-Calero, Game ai for a turn-based strategy game with plan adaptation and ontology-based retrieval, 2007.
[17] V. Menkovski, D. Metafas, Ai model for computer games based on case based reasoning and ai planning, 2008, pp. 295–302.
[18] D. Obradovič, A. Stahl, Learning by observing: Case-based decision making in complex strategy games, in: KI 2008: Advances in Artificial Intelligence, Springer Berlin Heidelberg, 2008, pp. 284–291.
[19] UltraBoardGames, Hanabi game overview, 2020. URL: https://www.ultraboardgames.com/hanabi/index.php.
[20] UltraBoardGames, Hanabi game rules, 2020. URL: https://www.ultraboardgames.com/hanabi/game-rules.php.
[21] R. Adobbati, A. N. Marshall, Gamebots: A 3d virtual test-bed for multi-agent research, in: Proceedings of the second international workshop on Infrastructure for Agents, MAS, and Scalable MAS, volume 5, 2001.
[22] J. Gemrot, R. Kadlec, Pogamut 3 can assist developers in building ai (not only) for their videogame agents, in: Agents for games and simulations, Springer, 2009, pp. 1–15.
[23] N. Hoobler, G. Humphreys, M. Agrawala, Visualizing competitive behaviors in multi-user virtual environments, IEEE/Visualization (2004) 163–170.
[24] J.-J. Bartels, S. Viefhaus, P. Yasrebi-Soppa, P. Reuß, K.-D. Althoff, Visualizing the behaviour of cbr agents in a fps scenario, in: D. Trabold, P. Welke, N. Piatkowski (Eds.), Lernen, Wissen, Daten, Analysen. GI-Workshop-Tage "Lernen, Wissen, Daten, Analysen" (LWDA-2020), September 9-11, Online, CEUR, 2020, pp. 130–141.
[25] J. M. Schönborn, K.-D. Althoff, Towards case-based reasoning in real-time strategy environments with seasalt, in: D. Trabold, P. Welke, N. Piatkowski (Eds.), Lernen, Wissen, Daten, Analysen. GI-Workshop-Tage "Lernen, Wissen, Daten, Analysen" (LWDA-2020), September 9-11, Online, CEUR, 2020, pp. 154–161.