Case-based Action Planning in a First Person Scenario Game

Pascal Reuss[1,2] and Jannis Hillmann[1] and Sebastian Viefhaus[1] and Klaus-Dieter Althoff[1,2]
reusspa@uni-hildesheim.de
basti.viefhaus@gmail.com
klaus-dieter.althoff@dfki.de

[1]Intelligent Information Systems Lab, University of Hildesheim
[2]Competence Center CBR, German Center for Artificial Intelligence, Kaiserslautern

Abstract. Creating a comprehensive and human-similar artificial intelligence in games in an interesting challenge and has been addressed in research and industry for several years. Several methods an technologies can be used to create computer controlled non-player characters, team mates, or opponents. Depending on the genre of the game, for example real-time strategy, board games, or first person scenarios, the tasks and challenges for an intelligent agent differs. In our scenario we choose a first-person scenario, where two software agents play against each other. While the behavior of one agent is rule-based, the other agent uses a case-based reasoning system to plan his tactics and actions. In this paper we present the first-person scenario and the rules and assumptions or it. We describe the knowledge modeling for our case-based agent in detail: the case structure and similarity model as well as the decision making process of the intelligent agent. We close the the paper with the presentation of an evaluation of our approach and a short outlook.

1 Introduction

Artificial Intelligence (AI) in computer games is an up-to-date research topic. There are two main goals for an AI: improve the opponent to be as good as possible and to create a challenge for the player. The first goal assumes, that the computer-controlled opponent will do everything to win a game, while the second way is to improve the game experience for the player. In this case the AI has to adapt to the players skills to stay beatable. In both ways the AI should learn from past experiences to adapt to the players strategy and tactic. Case-based Reasoning (CBR) as a problem solving paradigm that transfers human problem solving behavior on a computer, is a very interesting approach to use past experiences for solving new problems, especially in a dynamic environment like games.

There is several research that deals with CBR in video gaming scenarios. Most of them uses CBR in context of real-time strategy (RTS) games. Cheng and Thawonmas used CBR to enhance non-player characters (NPC) in RTS games. In many RTS games NPC behavior can easily be predicted, therefore the goal of Cheng and Thawonmas was to make the behavior less predictable using CBR.[8] Fagan and Cunningham used CBR

to predict the actions of the player in the game Space Invaders[9]. Several researchers worked with the Wargus mod for Warcraft 2 to improve the AI in this game[2],[12]. Another research in the RTS genre was done by Weber and Mateas. They used CBR to managed build orders of buildings in an effective way[16]. Cadena and Garrido used CBR and fuzzy logic to improve the opponent AI in the RTS game Starcraft[7]. An approach for an adaptive AI with CBR was developed by Bakkes and his colleagues[5]. Szczepanski and Aamodt developed an AI for the game Warcraft 3. The approach focuses on the micromanagement of units in this game.[1]

The CBR-based approaches take place in the RTS genre, but there are also approaches for AI improvement in first-person scenarios (FPS). Auslander and his colleagues developed an approach called CBRetaliate. They combined CBR with Reinforcement Learning to find action strategies in the game Unreal Tournament.[3] 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.[13],[15] Another approach was also used in Quake 2 by Laird. He uses anticipation for opponent AI to predict the actions of the player[11].

CBR was used in computer games in the last years with many approaches, but mainly focused on the RTS genre. In this paper we present an approach to use CBR in an self-developed first person scenario to retrieve action plans for a software agent. We describe the rules and assumptions of our first person scenario and the structure of our implemented application. We describe the knowledge modeling for our CBR system and the improvement we made to our modeling. We also describe the evaluation and the result of our case-based agent with the original knowledge model and with the improved model. We close the paper with a short outlook to future work.

2 Case-based action planning in a first person scenario game

The basic rules for a first person combat scenario are simple. Two or more players compete in an arena that consists of floors, rooms, and obstacles. The players in the FPS game move through the arena, collect so-called power-ups and fight each other. The power-ups could be health, weapons, or ammunition. The goal of the classic FPS game is to reach a certain amount of scores. During the fight an agent can loose life points. When the life points of a player reaches zero, he de-spawns and the other player 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 differs in the amount of damage they deal to the life points of a player, the effective range, or the accuracy. A more detailed description of this form of FPS games can be found in [11].

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 looses a point if he losses all his life points.

2.1 Application structure and software agents

The game application consists of three components: the game component, the multi-agent system, and the CBR system. The game component was made with Unity 3D, a game developing engine with several features to ease the creation of graphics and logic for entities in games[14]. This game component was developed during a student's thesis and therefore was available and configurable for our approach. Alternatively, we could have used the open source engine Unreal Engine 4[10], but we already have experiences with Unity 3D and not with Unreal Engine. Because Unity 3D has a similar set of pre-build resources for FPS scenarios, we decided to work with Unity 3D. The multi-agent system was implemented with the framework Boris.NET, a C# specific implementation of the Boris framework for multi-agent programming[6], and the CBR system was implemented with the open source tool myCBR[4]. 1 shows the structure of the application with the used programming languages.

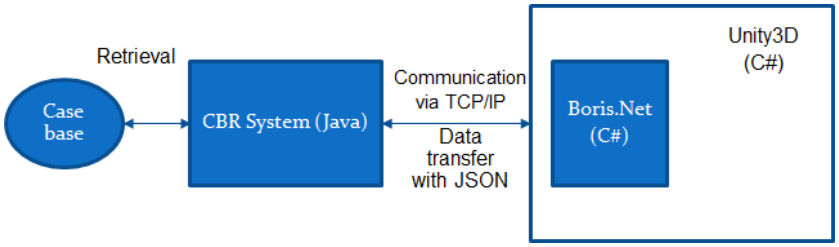

Fig. 1. Structure of the application

The Unity framework was used to design the arena in which the software agents competes each other and to visualize the movement and actions of these software agents. The arena was designed to met the conditions of the first person scenario described above. Figure 2 shows a screen shot of the arena.

The arena has a floor and two visible walls as boundaries. On the other two sides are also walls, which are invisible to enable the view on the acting agents. Inside the arena several obstacles can be found. These obstacles act as cover and block the agent's field of view and shooting. Three different collectibles were implemented in the arena: health container (visualized as a piece of pizza), ammunition container (visualized as a small green box), and a weapon. Figure 3 shows all three collectibles in the arena.

Throughout the arena, five spawn points for the players were created. Every time a software agent de-spawns, he randomly spawns at one of this five points. The points are distributed in the four corners of the arena and one in the middle. Before an agent spawns at a certain spawn point, it is checked if the other agent is at or near the spawn point, to avoid a situation, where both agents starts at the same spawn point. In addition to the player spawn points, several collectible spawn points were defined: two spawn points for health, two spawn points for a weapon and five spawn points for ammunition. If a collectible was collected by an agent the spawn point remains idle for 20 to 30 seconds before the collectible spawns again.

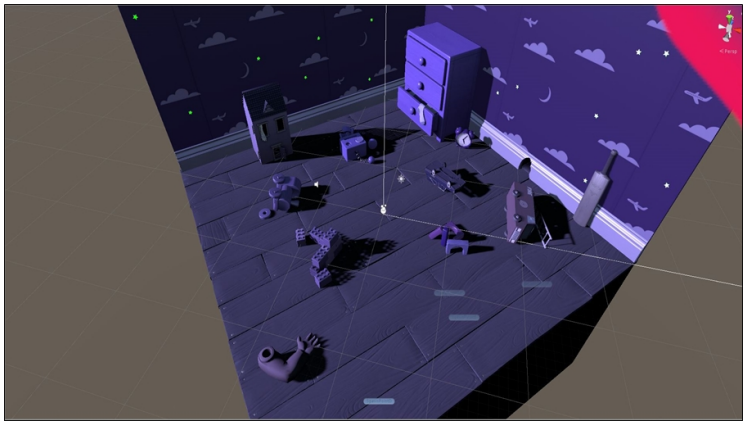

Fig. 2. An overview of the game arena

Fig. 3. Collectibles in the arena

In addition to the arena, Unity3D was also used to implement the basic controlling of the player with six basic actions:

- MoveTo - The agent moves to a specific point in the arena
- CollectItem(T) - The agent moves to the position of a given collectible T and collects it
- Reload - The agent reload the currently equipped weapon
- Shoot - The agent shoots at a visible enemy
- SwitchWeapon - The agents switches the currently equipped weapon
- UseCover - The agents moves behind a near obstacle to avoid the line of fire of the enemy

All these actions are implemented to be executed in the arena. An action plan of an agent may contain several actions. Some actions have to be executed in sequence:

reload and shoot, switch a weapon and shoot. The other actions can be executed in parallel. An agent is able to move and shoot at the same time or use cover and reload.

The multi-agent system consists of four agents: two player agents, a communication agent, and a planning agent. One player agent represents the scripted bot that uses rules to act in the arena. The behavior of scripted AI is based on five rules:

- if enemy visible → move to enemy and shoot
- if better weapon visible → collect weapon
- if health or ammunition needed → collect needed item
- if enemy not visible & last position known → move to last known position
- if enemy not visible & last position not known → move to the middle of the arena and look for enemy

The case-based player consists of the other three agents. We decided to distribute the tasks over three agents rather than using only one agent to perform several tasks in parallel. The player agent for the case-based player is responsible for acting in the arena. It has access to the movement and action scripts of the game component. It has information about the current situation and passes this information to the communication agent. This agent is responsible for the communication with the CBR system. The situation description is transformed into a JSON representation and passed to the CBR system. The CBR system performs a retrieval based on the situation description and delivers an action plan as solution back to the communication agent. The communication agent passes the solution to the planning agent, which is responsible for translating the JSON solution into an executable plan for the player agent. A newly retrieved plan replaces the current plan of the player agent and he starts to execute the new plan. Because of the task distribution, the player agent can act in the arena, while the two other agents can retrieve and build an executable plan. In the version of our application available to the publication time of this paper, the retain phase of the CBR cycle is not implemented yet. Learning from experiences during the game and storing new cases with new or adapted plans is part of the concept, but will be realized in the near future.

2.2 Knowledge modeling for the case-based agent

The case-based agent uses cases with a situation description and an associated actions to plan his moves in the game level. The situation description was derived from the first person scenario and the basic assumptions made to the scenario . In our first version, the description contained 17 attributes:

- currentAmmunition - the current amount of ammunition in the active clip
- currentOverallAmmunition - the current amount of ammunition in all clips
- distanceToAmmunition - the distance to the nearest ammunition collectible
- distanceToCover - the distance to the nearest cover
- distanceToEnemy - the distance to the enemy agent
- distanceToHealth - the distance to the nearest health collectible
- distanceToWeapon - the distance to the nearest weapon collectible
- equippedWeapon - the current equipped weapon

– isAmmunitionNeeded -
– isCoverNeeded -
– isCovered - true if the agent is currently in cover, false if not
– isEnemyAlive - true if the agents knows his enemy agent is active
– isEnemyVisible - true if the enemy agent is currently visible by the agent
– isHealthNeeded - true if the agent needs a health container
– isWeaponNeeded - true if the agent needs a better weapon
– lastPosition - the last know position of the enemy agent
– ownHealth - the current amount of life points of the agent

The attributes have integer, symbolic, or boolean data types. The attributes *currentAmmunition*, *currentOverallAmmunition*, and *ownHealth* use a integer data type. All attributes starting with an "'is"' uses a boolean data type and the remaining attributes use a symbolic data type. The distance to an entity in the game is not represented a an absolute number, but it is transformed into a four value symbolic representation: near, middle, far, and unknown. This way the similarity measure is less complex. The transformation is the same for all distance attributes. 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. Figure 4 shows the similarity matrix for all distance attributes. The global similarity on case level is computed using a weighted sum of all local attribute similarities. For the initial knowledge modeling the weights were all set to one.

	far	middle	near	unknown
far	1.0	0.7	0.2	0.0
middle	0.7	1.0	0.7	0.0
near	0.2	0.7	1.0	0.0
unknown	0.0	0.0	0.0	1.0

Fig. 4. Similarity measure for distance attributes

The solution of specific situation description is an action plan with several single actions. The plan representation is very simple. The plan is represented as a string that contains two or more actions. We modeled 15 initial cases based on human behavior in specific situations. Figure 5 shows the situation description for the first five case and Figure 6 the actions plan of these situations.

With this knowledge model, we performed an evaluation as described in 2.3. From our perspective there were two main problems for the CBR agent. One problem can be found in the knowledge modeling. All attributes of the situation description have the same weight. This means all aspects of the situation have the same priority. The scripted bot has a priority to gather the better weapon. As a consequence the scripted agent has more often the better weapon than the case-based agent and therefore deals

St/S	isEV	dE	lP	isEA	oH	eW	cA	cOA	isWN	isAN	isHN	isCN	isC	dW	dA	dH	dC
S1	True	Near	Near	True	Full	Machine Gun	Much	Mi-ddle	False	True	False	False	True	uk	Mi-ddle	uk	Near
S2	True	Mi-ddle	Mi-ddle	True	Full	Machine Gun	Full	Much	False	False	False	False	True	Far	Mi-ddle	Mi-ddle	Near
S3	True	Far	Far	True	Full	Machine Gun	Mi-ddle	Mi-ddle	False	True	False	False	False	Mi-ddle	Mi-ddle	Far	uk
S4	True	Near	Near	True	Mi-ddle	Machine Gun	Few	Few	False	True	False	True	False	Mi-ddle	Far	Mi-ddle	Near
S5	False	uk	uk	True	Full	Pistol	Mi-ddle	Mi-ddle	True	False	False	True	False	Near	Mi-ddle	Mi-ddle	Far

Fig. 5. The situation description of the first five cases

S1	MoveTo <-> Shoot
S2	MoveTo <-> Shoot
S3	MoveTo <-> Reload -> Shoot
S4	MoveTo <-> Shoot <-> UseCover -> Reload
S5	MoveTo -> CollectItem<Weapon> -> SwitchWeapon

Fig. 6. The action plans of the first five cases

more damage to the case-based agent. The other problem can be found on the agent implementation, more precise in the frequency of the retrieval. The agent asks for a new plan every time the situation changes. This means every second a new plan is retrieved and the current plan is replaced. In the worst case, a working plan is replaced with a bad plan. For example both agents are visible to each other, move towards each other and deal damage to each other. While the case-base agent looses life points during the fight, he spots a health container. The newly retrieved plan forces the case-based agent to move to the health container and stop shooting at the enemy. While the case-based agent tries to reach the health container, he loses all his live points to the damage of the enemy agent.

As a consequence we adapted the knowledge model of the CBR system. We added a new attribute called target priority. This attribute is not set as a situation aspect, but is derived from a situation. Based of a given situation, the target priority can differ between *no priority*, *arm and collect*, *protect and hide*, and *search and destroy*. The action plans are bound to a specific priority. This way, we retrieve more appropriate action plans for a given situation. Figure 7 shows on the left side the defined target priorities and the associated attribute values.

In addition, we changed the retrieval frequency. The case-based agent gets an intention reconsidering function to calculate if a new plan should be retrieved or the current plan should be kept. The reconsidering function uses the weightings of the attributes

search and destroy	isWeaponNeeded	false
	isAmmunitionNeeded	false
	isHealthNeeded	false
	isEnemyAlive	true
	isEnemyVisible	true
	distanceToEnemy	{near, middle}
protect and hide	isHealthNeeded	true
	isCovered	false
	distanceToCover	near
	isCoverNeeded	true
arm and collect	isWeaponNeeded	true
	isHealthNeeded	true
	isAmmunitionNeeded	true
	distanceToWeapon	{near, middle}
	distanceToHealth	{near, middle}
	distanceToAmmunition	{near, middle}
	isEnemyAlive	false

currentAmmunition	2
currentOverallAmmunition	1
distanceToAmmunition	1
distanceToCover	1
distanceToEnemy	1
distanceToHealth	1
distanceToWeapon	1
equippedWeapon	1
isAmmunitionNeeded	2
isCoverNeeded	2
isCovered	1
isEnemyAlive	5
isEnemyVisible	5
isHealthNeeded	2
isWeaponNeeded	2
lastPosition	2
ownHealth	2

Fig. 7. Target priorities and associated attribute values (left), Attribute weights for intention reconsidering(right)

to calculate the impact of an attribute change on the overall situation. If the sum of the weights reaches a certain threshold, than the situation has changed significantly and a new plan should be retrieved. Figure 7 shows the chosen attribute weights on the right side. The threshold for a significant situation change was set to several values. With a threshold of three we achieved the best retrieval frequency.

2.3 Evaluation

We evaluated the first the knowledge modeling approach with a set of four matches between the scripted agent and case-based agent. Every match lasted 15 minutes. Every positive or negative point was recorded in a CSV file. In all matches the CBR agent starts with the 15 initial cases. The results show, that the case-based agent performs worse than the scripted bot in the overall results. There are several phases during a match, where the case-based agent has an advantage, but in all matches the scripted bot wins. After improving the knowledge model and the case-based player agent as described in 2.2, we evaluated the system again, with the same conditions as the first evaluation. The results show, that the improvements to the knowledge model and the use of intention reconsidering for the case-based player agent lead to a better performance of the case-based agent. While it is not better in general than the scripted bot, is is roughly on the same level. Figure 8 shows the results for eight matches. Four matches between the rule-based agent (Rule I) and the case-based agent with the initial knowledge modeling (CBR B) and four matches between the rule-based agent (Rule II) and the case-based agent with the improved knowledge model and intention reconsidering (CBR Imp).

A deeper view into the log flies of the matches shows that the agent with the better weapon has a lucky streak and scores more often than the opponent. In the first four matches the initial case-based agent ignores the better weapon most of the times and tries to shoot the opponent with the starting weapon, while the rule-based agent gets the better weapon in many situation before engaging the opponent. After improving the knowledge model the case-based agent collects the better weapon more often and

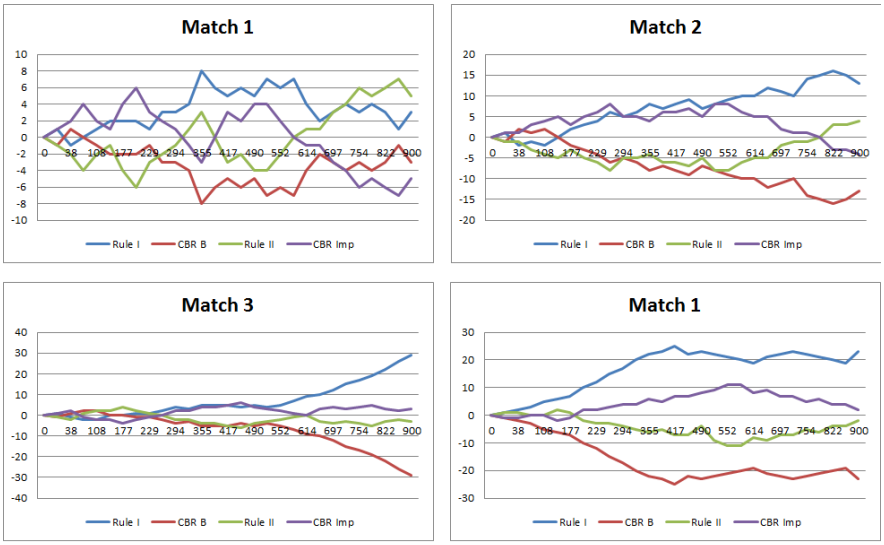

Fig. 8. Evaluation results between the rule-based and the case-based agent

therefore gets more often the lucky streak. The conclusion of the evaluation result on this scenarios is that the collecting and using the better weapon is the key to victory. This victory criterion reduces the complexity of the game far more than intended and therefore the complexity of the game has to be improved to enable different strategies and tactics to achieve victory.

3 Summary and Outlook

In this paper we present an approach for case-based action planning in an FPS game. We described the application structure with the visualization component, the multi-agent system and the knowledge modeling of the CBR system. In addition, we have shown our evaluation and the out coming results as well as the consequences and the improvements for and of our application.

For future work we plan to extend the game in several ways. First of all, we will implement the learning process for the case-based agent. The agent will be able to learn new plans and store feedback about the successful execution of a plan. We will also enhance the arena to have more complex situations with more obstacles and more collectibles, because one problem with the poor performance of the case-based agent seems to be the simple level design. This simple design does not allow the case-based agent to use the possible advantages of CBR. In addition, we will add more possible actions like ambush, strafing during movement, and we will extend the game play from 1 vs 1 to a team-based matched with several agents in each team.

References

1. Aamodt, A.: Case-based reasoning for improved micromanagement in real-time strategy games. In: Paper Presented at the Case-Based Reasoning for Computer Games at the 8th International Conference on Case-Based Reasoning (2009)
2. Aha, D.W., Molineaux, M., Ponsen, M.: Learning to win: Case-based plan selection in a real-time strategy game. In: Case-Based Reasoning Research and Development. pp. 5–20. Springer Berlin Heidelberg (2005)
3. Auslander, B., Lee-Urban, S., Hogg, C., Muñoz-Avila, H.: Recognizing the enemy: Combining reinforcement learning with strategy selection using case-based reasoning. In: Advances in Case-Based Reasoning. pp. 59–73. Springer Berlin Heidelberg (2008)
4. Bach, K., Sauer, C., Althoff, K.D., Roth-Berghofer, T.: Knowledge modeling with the open source tool mycbr. In: Nalepa, G.J., Baumeister, J., Kaczor, K. (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, Prague, Czech Republic. CEUR Workshop Proceedings (http://ceur-ws.org/) (2014)
5. Bakkes, S.C., Spronck, P.H., Jaap van den Herik, H.: Opponent modelling for case-based adaptive game ai. Entertainment Computing 1, 27–37 (2009)
6. Bojarpour, A.: Boris.net (2009), http://www.alibojar.com/boris-net
7. Cadena, P., Garrido, L.: Fuzzy case-based reasoning for managing strategic and tactical reasoning in starcraft. In: Batyrshin, I., Sidorov, G. (eds.) Advances in Artificial Intelligence. pp. 113–124. Springer Berlin Heidelberg (2011)
8. Cheng, D.C., Thawonmas, R.: Case-based plan recognition for real-time strategy games. In: Proceedings of the Fifth Game-On International Conference (2004)
9. Fagan, M., Cunningham, P.: Case-based plan recognition in computer games. In: Proceedings of the 5th International Conference on Case-based Reasoning: Research and Development. pp. 161–170. ICCBR'03, Springer-Verlag (2003)
10. Games, E.: Unreal engine 4 (2018), https://docs.unrealengine.com/en-us/
11. Laird, J.: It knows what you're going to do: Adding anticipation to a quakebot. In: Proceedings of the International Conference on Autonomous Agents. pp. 385–392 (2001)
12. Ontañón, S., Mishra, K., Sugandh, N., Ram, A.: Case-based planning and execution for real-time strategy games. In: Proceedings of the 7th International Conference on Case-Based Reasoning: Case-Based Reasoning Research and Development. pp. 164–178. ICCBR '07, Springer-Verlag (2007)
13. Priesterjahn, S., Kramer, O., Weimer, A., Goebels, A.: Evolution of human-competitive agents in modern computer games. In: 2006 IEEE International Conference on Evolutionary Computation. pp. 777–784 (2006)
14. Technologies, U.: Unity 3d overview (2018), https://unity3d.com/de/public-relations
15. Thurau, C., Bauckhage, C., Sagerer, G.: Combining self organizing maps and multilayer perceptrons to learn bot-behaviour for a commercial game. In: GAME-ON (2003)
16. Weber, B.G., Mateas, M.: Case-based reasoning for build order in real-time strategy games. In: Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2009) (2009)