Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 156 (2019) 404–413

Procedia Computer Science

www.elsevier.com/locate/procedia

8th International Young Scientist Conference on Computational Science

Applying Behavior characteristics to decision-making process to create believable game AI

Andrey Simonov*, Aleksandr Zagarskikh, Victor Fedorov

eScience Research Institute, ITMO University, Saint Petersburg, Russia

Abstract

With the development of artificial intelligence in computer games the problem of creating characters with believable and diverse behavior to inhabit in-game worlds becomes more and more actual. A big number of required characters and high standards of a modern game artificial intelligence makes the problem even more complex. In this paper we propose a utility-based decision-making model which gives the possibility to generate characters with believable behavior. The believability of such characters comes from their decision-making process that takes into account not only assessment of game environment, but also their personal characteristics and social status. Designed model was used to generate AI driven characters for a development of player's opponents with personality traits for a computer card strategy. The model was used to simulate human flows on main railroad hub of 2014 Winter Olympics in order to reveal areas where pedestrian flow should be controlled.

Keywords: Game artificial intelligence; Utility theory; Decision-making systems; Believable behavior

1. Introduction

Artificial Intelligence (AI) takes an important place in games of any genre. Developers use it to create an in-game environment for the improved immersion into gameplay that would help to discover peculiar properties of a game world. Many games are based on AI-driven characters and their reflections on player's actions. Complex behavior

* Corresponding author. Tel.: +7-931-225-2568;
E-mail address: Simonov.andrei.b@gmail.com

10.1016/j.procs.2019.08.222

rules are applied to such characters. Designers of AI put much effort to make these rules more correlated with game project specialties.

One of the most complex and vast cases of AI development is a game with a so called "open" world. The key feature of this type of a game is a player's ability to interact with any in-game object or Non-Playable Character (NPC) at any time. Usually in order to provide such ability developers create a big world inhabited by many characters. Nowadays game developers tend to increase the size of game worlds. However, this raises the required number of characters to inhabit the world. Towns, villages and countries need a big number of agents that implement a territory dependent behavior. Moreover, a contemporary in-game AI must provide every NPC with an individual behavior to make it feel unique. The behavior also must look realistic to make an impression that a NPC makes decisions on its own and doesn't follow any prescribed behavior rules. There can be thousands of such characters in modern games. The problem of creation of so many characters with a diverse and complicated behavior is quite vast and requires a modern and complex approach to solve it.

On the other hand, methods that allow modelling crowd dynamics with high precision are used in a multi-agent simulation to create agents. An example of methods usage is a performing an experiment to test an agent behavior inside a building in a dense crowd. Advanced behavior frameworks can process thousands of agents with different parameters in a real time. A group behavior and realistic agent movement are the main interest points in such experiments. Although the observed methods have a practical potential in game industry, resulting agents are more a part of a crowd rather than characters with individual behavior.

Consequently, two actual problems can be stated. The first one is the problem of creation of a big number of characters with a diverse and believable behavior to inhabit in-game worlds. The second one is the poor ability to perform experiments in pedestrian dynamics. A modern approach is required to solve them.

In this paper, a model regulating NPC's in-game environmental interaction and decision-making is proposed as a problem solution. It is based on an integration of methods of a multi-agent simulation and modern game development methods of an AI creation. The proposed model allows creation of NPCs with a believable behavior that fits the last game development trends and an in-game environment mapping for area and interactive object positioning.

2. Related works

2.1. Definition of "Believable" behavior

The definition of realism is one of the most important things in AI-driven NPC behavior evaluation. That's why it is important to define the meaning of the "believable behavior" in the context of the current work. Even though the definition describes an agent's behavior, perceiving a character like a real person depends more on an agent observer. Hence the psychological aspects of the definition should be taken in consideration.

The believability of an agent's behavior depends on observer's expectations. This follows from either both game rules and character's appearance and backstory. For example, in general case, a robot NPC's believable behavior quite differs from that one of a human NPC. Accordingly, in a case of a computer game, the believable behavior is a behavior that fits into game rules and corresponds to a character's personality.

Discussions about believability evaluation, realism and AI progression in computer science often includes references to the Turing test. The aim of the test it to discover whether machines can think or not. During the test a man is suggested to hold one dialogue with a real person and another one with a computer driven by an AI. After the conversation the man must determine a computer amongst his companions depending on the received answers. The test is considered passed when the computer is chosen as a real person. In other words, the test is depending on how a believable computer imitates human behavior. But many scholars criticize the test. They say that its results don't prove machines to have intelligence and that machines can't have it [1]. However, in computer games the ability of an AI to mimic human behavior and act natural in predefined circumstances is essential.

Thus, perception of a character's behavior by a spectator is the main factor to determine the believable behavior. The perception is unique for every spectator due to an individual life experience rather than usual processes in a human's brain. A player expects an AI to behave like people around him in his everyday life.

Parameters that determine a virtual agents' behavior have been observed in several science papers. Iskander Umarov has formed a list of requirements for a believable AI [2]. The main themes of the requirements are the ways agents perceive an environment and making these ways to be more like those of a player. Another list of requirements is described in a book dedicated to an AI in computer games [3]. The list consists of follows:

- A behavior that fits into game rules
- Dynamic behavior adjustments
- An environmental perception that resembles the one of a player
- Personal characteristics that allow to determine a particular character
- A social interaction between agents
- An appropriate prediction level (too predictable or too unpredictable behavior makes a gameplay less believable)

Thus, in the current work the "believable behavior" is defined as a behavior that meets the listed requirements. However, in the end the believability of a behavior is always determined by a spectator.

2.2. Paradigms of decision-making systems

The Decision-making system is an essential component in any in-game AI. Such systems provide NPCs with an ability to choose an option amongst possible ones and allows to describe a behavior. NPCs can use the provided ability to find a particular object in a game world or to make a decision about something. There are several approaches for creation of such systems in game industry. The most potential and frequently used ones are described below with analysis of their advantages and disadvantages.

2.2.1. Planned Behavior

The most primitive in terms of internal architecture is the approach that uses a predefined NPCs' behavior [4] stored in a game logic. This results in a non-flexible, pre-planned behavior that doesn't provide any choices. This approach is useful for creation of an everyday behavior when a particular action must occur in a particular time. Actions changes in a strict order according to a schedule. Moreover, such behavior is preferable in cases of a fixed game scenario. But in other cases, this approach requires more time to implement. It doesn't meet the requirements of the current work in creation of a dynamic behavior.

2.2.2. Finite-state machine

The most common character behavior management models are the ones based on a finite-state machines [5]. These models are appropriate for NPCs with a few possible states. An NPC that patrols some area is a good example for such model.

However, the bigger the number of possible states the more complicated the finite-state machine becomes, thus increasing an implementation time. Moreover, these machines are strongly tied to the generated circumstances and it is hard to use a single one in multiple projects simultaneously. Consequently, the majority of modern computer games doesn't utilize finite-state machines in a NPC development. Although some of them do use improved versions of the machines [6].

2.2.3. Behavior Trees

Nowadays, behavior trees dominate amongst the technologies for an AI creation in computer games [7]. A behavior tree is a hierarchical structure whose nodes represent an agent state and contain logic that is executed when the node is activated. A vast variety of nodes that differs by ways of management of thread of execution provides a high potential of creation of complex behavior structures.

Behavior trees are superior to finite-state machines due to lack of problems in character state changes [8]. Moreover, this approach allows to create more complex behavior structures and to add new behavior patterns with ease.

Behavior trees also have several disadvantages. First, Depth-first search (DFS) takes a lot of time to complete in a big tree when the search occurs every frame. Second, behavior trees can't fully implement a believable character behavior due to the lack of an ability to make random decisions. Third, a complex behavior must be described with a huge tree that is hard to navigate and debug.

There are some extensions to behavior trees that can include parallel computation improvements [9] or integration of deep learning methods [10]. But game industry tends to use more modern approaches for AI creation.

2.2.4. Machine Learning methods

Due to the increase in computing potential the Artificial Neural Network (ANN) becomes more popular in computer games in the last several years. The main usage of the ANN is to provide a player with a proper challenge. In 2017 a bot (OpenAI) for the game DOTA 2 was created with the usage of neural networks. Another example is an ANN called "DeepMind" for the game StarCraft 2 [11].

In general case, machine learning methods in AI classifies action parameters by input data depending on a network type and architecture. In most cases these networks learn in offline mode when a game is in development due to high computational resource requirements and unpredictable results.

On the other hand, reinforcement learning has gained special attention in AI researches. This is a learning without a teacher where a system compares computed results with expected ones on its own and changes weights between them depending on similarities in both results. Such approach allows to ignore the lack of an appropriate mapped database. However, a strict definition of a network error approximation function is required for the methods to work. Sometimes it is a hard task to define such function. The famous examples of reinforcement learning in game industry are an AI that completes Atari games [12] and systems for character training in AI competitions [13].

2.2.5. Utility AI

The Utility AI is a technique based on an estimating profit of executing actions [14]. The utility algorithms [15] gain popularity nowadays amongst some others. They provide a flexible and scalable approach for creating NPC decision-making systems. When it becomes necessary to choose between several options, an NPC calculates the benefit (utility) of each option and then chooses the option with the highest utilities. The process of estimating benefits considers not only character internal parameters, but also the assessment of a game environment state.

The estimation of benefits brings the decision-making process to a level of fuzzy logic [16][17], and that leads to the ability of agents to make believable decisions even in a game situation that was not predicted during the design.

One of the main advantages of the utility approach is the way of presenting behavior rules. They may be described by a curve representation, that shows the superposition of multiple behavior rules, or by a table representation that provides the general overview on changing of utility value in different game situations. The other advantage of the Utility AI is that the behavior built with it becomes highly reusable and expandable. Expansion of a character's behavior comes with adding new available options to his utility-based decision-making system, and the behavior rule developed for one character may be easily translated to another one.

2.2.6. Research objective

In the current work our goal is to design and develop a decision-making model that allows to easily create a big number of characters with a believable behavior to inhabit game worlds and allows to perform experiments in pedestrian dynamics. To meet the latest requirements for an AI the model must include methods of behavior diversification.

The next set of tasks has been formulated to achieve the goal:

1. Development of a decision-making component based on profit estimation function;
2. Design and development of an agent behavior diversification component;
3. Implementation of modules that allows to integrate developed complex into visualization system;
4. Testing of the developed model for a character creation for game worlds and a pedestrian dynamic simulation.

3. Method

The description of the developed model and its essential components are presented in this section. The model has been designed based on several core concepts: Observe-Orient-Decide-Act (OODA) decision cycle, Utility AI, multi-agent modeling and game industry approaches to create characters with a believable and diverse behavior.

3.1. Two-Factor decision-making cycle

The OODA loop is characterized by two phases of decision-making: Orient and Decide. In the Orient phase the decision-making process takes into account agent's personal characteristics: for example, his social status or current emotional state. In this regard the Orient phase provides a big impact on a believability of an NPC behavior. On the other hand, in the Decide phase an emotional decision made from the Orient phase is adjusted by more rational considerations, based on rules of interacting with a game environment.

Therefore, it was decided to split the decision-making cycle into two components. The first will operate in the Orient phase of the cycle and will choose an Intention for an agent. Then, in the Decide phase, agents will choose their next Action based on the current intention.

3.1.1. Intention selection component

Intentions are basic entities of a decision-making system. Intentions are a representation of an agent's desires in a game environment. To select a suitable intention, agents should assess surrounding circumstances and their personal characteristics.

It was decided to apply the Utility AI approach to evaluate this assessing. To select an intention an agent will estimate a utility (benefit) from accepting each intention, and then the intention with the highest benefit will be chosen. To estimate an intention's current benefit, we propose to match an urge value to each intention. The urge value shows how much an agent desires to accept this intention at the moment. To calculate the intention, we propose to use a utility growing function that calculate current utility for intentions based on their urge and agents' inner parameters. To achieve more flexibility in tuning characters' behavior we can set the utility growing function for each intention individually. To describe the dynamic of changing intention the urge values may rise or drop with time or under the influence of outer forces. The equation 1 describes the process of urge changing with time, where s stands for urge, g is an urge growth speed and t is a predefined period of time.

$$s = s + g * t \tag{1}$$

Therefore, to select a dominant intention, each agent forms a set of intentions available at the moment. Then, for each intention they calculate a utility value and find the intention with the highest benefit. One thing to mention is that each agent could have only one dominant intention at the moment, so the cycle of selecting intention will not be firing until the current intention will become fully satisfied. In general case most intentions use a predefined utility calculation function, for example equation 2 and equation 3 describes sigmoidal and tangential utility functions. But to achieve more flexibility of a decision making there is an option to set a unique function to each intention.

$$U_S = \frac{1}{1 + e^{(-1*s)}} \tag{2}$$

$$U_T = \frac{e^s - e^{(-1*s)}}{e^s + e^{(-1*s)}} \tag{3}$$

3.1.2. Action Selection Component

The second phase of a decision-making process is a selection of suitable action that will be executed by characters on the last phase of the OODA loop. An action is a character's reaction to chosen intentions. Upon execution each action changes the agent's inner state, for example a movement to an object in a game environment or start animation sequence. Also, an action may contain some logic that the character will inherit after accepting action. This can contain rules of agent positioning in a game world or rules that affect other agents' state.

Unlike the intention, the action selection process is influenced by in-game rules and an agent role inside it. The actions are selected based on the current dominant intention. The selection algorithm is based on a measuring utility of executing actions. To do this a list of available actions is associated with each intention. In the Decide phase the

Action Selection component calculates the utility of executing each action from the list of the dominant intention. After that the agent executes the most benefit action. In the simplest case the benefit from executing an action is calculated using the value of the action's priority. The higher priority gives more chances that the action will be executed. But in most general way actions require setting an individual utility calculation function with a lot of parameters.

The actions fall into two categories: inner and outer. The inner actions are designed to change its executor inner state and cannot affect other objects. On the other hand, an outer action can alter a state of interactive objects of a game environment or modify a state of other agents, for example, modifying urges of their intentions. Also the actions are divided by their duration. For instance, some actions may reduce the urge of a matched intention, and continue its execution until the urge will not fall under a predefined value. Another action group may last for a certain amount of time and then being interrupted.

To achieve the believability of a character behavior we should complement it with the methods of adding more characters' personality to the decision-making process, to make their decision more unique and diverse. To achieve this, we can add components that will functionate in the Orient phase of the OODA loop.

3.1.3. Agents Roles Component

According to the two-factor decision-making model in the Orient phase of the OODA loop an agent should obtain the list of available for him intentions and choose the most suitable from them. To create this list, we design a Roles Component. This component provides the list of intentions based on agents' current roles. A role is an entity that helps describing characters and game situations. Each role contains a list of intentions and actions, which characters inherit from them. The roles are divided into two groups, based on the duration of their effect: Behavior Groups and Temporary Roles

The Behavior Groups are attached to characters on the design stage. They determine characters' goals in a game environment and the way characters interact with the objects in a game world. The Behavior Groups are constant and cannot be changed during the game simulation, providing a prior intentions and actions for characters. With attaching Behavior Groups to characters they are divided into social categories, for example according to their professions or beliefs. One agent may have multiple Behavior Groups, describing various aspects of the agents' personality. But characters with the same set of Behavior Groups will have the same behavior and the believability of a game will be breached.

To avoid this problem, we propose to accompany Behavior Groups with Temporary Roles. The list of such roles is constantly changed during the game. It is updated according to the situation in the game environment around the agents, for instance when the agents attend some area, the Temporary Role may be added to this list. So the Temporary Roles used to describe the agents' personal infatuations, outer exposure or environment context around them. According to the list of Temporary Roles characters inherit the list of intentions and actions which modify their behavior with changing states of the game environment. The behavior rules inherited from Temporary Roles are deducted when the matched role stops affecting the agent.

Moreover, roles may contain internal variables that change the calculations of the utility values for Intentions and Actions matched to the role. With this approach the number of behavior rules that are implemented in characters is perceptibly reduced. This, in turn, leads to increasing the speed of developing a big number of characters with diverse and believable behavior. The behavior rules that are stored in roles may be used to keep an agent's progress in the role context. For instance, it can contain logic of changing an agent behavior according to some agent skills.

In summary, the list of available for a character Intentions and Actions for the Orient and the Decide phases is formed accordingly to the character's combination of Behavior Groups and Temporary Roles. Such approach grants an ability to alter the behavior of the agents taking into account the game environment state and their global goal.

3.1.4. Behavior characteristic component

Agents of one Behavior Group will have equal behavior in the same game situation. This means that even if the behavior of a single character is believable, the behavior of several such characters will become predictable and repetitive. To solve this problem, we propose a Behavior characteristic (BC) concept that allows tuning agents' behavior according to their individuality and personality.

Behavior characteristic is a numerical value $h \in (0,1]$ that influence the utility calculation process for a predefined group of Intentions and Actions. For each Behavior characteristic a set of influence values $w \in [0,1]$ is matched, that shows how much BC will change a utility value for a certain utility group. In this way, the general function of such influence on Intentions' and Actions' utilities is shown at equation 4.

$$U_0' = \begin{cases} U_0 & w_c = 0 \\ \dfrac{U_0}{w_c h_0} & 0 < w_c \leq 1 \end{cases} \tag{4}$$

Also BC may individually determine the way it changes utility values to be able to create complex behavior rules. The BC value may be increased or decreased during the game simulation to show the changes in agents' personalities. They also provide the way to bring more individualism in the characters' interaction with the game objects or other agents by tweaking it according to BC values. The agents may also receive a new BC upon entering areas in the game world or by receiving a new Temporal Role to tune their behavior with respect to the environment state.

To create a set of Behavior characteristics several possible approaches can be used. Thus, BC may be based on a psychological aspect of a human personality (for instance, five-factor personality model) or the approach from role-playing computer games for describing physical and mental state of the agents.

The BC approach provides the easy way to generate characters with a diverse behavior by generating unique set of BC values for each character independently.

3.2. Decision-making algorithm

By combining all the designed components in to a single model, we can formulate general algorithm (see Fig 1) of the two-factor decision-making process for characters of computer games and for agents of crowd simulation experiments. The algorithm is based on the OODA cycle and uses the utility AI concept to create a believable character behavior.

In this way, at the game's start the characters are divided according to their Behavior Groups, and they start to act according to the rules inherited from Behavior Groups. During the game process the characters adapt to changes of the environment state by receiving new behavior logic from the Temporal Roles. In addition, all the decisions made by the characters are affected by an individual set of each agent Behavior characteristics.

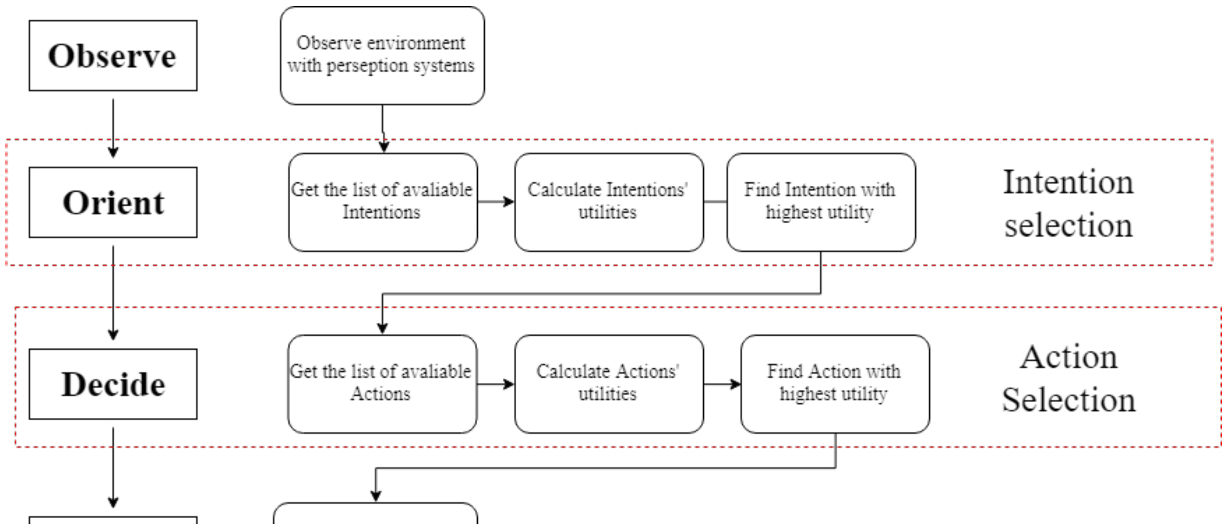
Observe
Observe environment with perseption systems
Orient
Get the list of avaliable Intentions
Calculate Intentions' utilities
Find Intention with highest utility
Intention selection
Decide
Get the list of avaliable Actions
Calculate Actions' utilities
Find Action with highest utility
Action Selection

Fig. 1. Decision-making algorithm

4. Experiments

4.1. Olympic Park train station

The first application of the designed model was a project of pedestrian mobility simulation in Olympic Park train station in Sochi during Winter Olympic 2014 to validate the effectiveness of decisions made by architectures during the construction planning [18]. This train station was a main transport hub to reach Olympic objects. The mean number of station visitors during Olympic Games was 8000 persons per hour.

To create the believable and realistic behavior of such amount of agents, it was decided to divide them into Behavior Groups accordingly to their purpose of visiting the station. The Behavior characteristics were used to bring more individuality to the agents' behavior, for instance they were affecting the agents' speed or each agent's personal comfort zone. By applying this BC, the human flow becomes more heterogeneous and realistic.

The simulation was developed in an open source game engine Unreal Engine 4 [19] with the application of crowd dynamic simulation framework Menge to control pathfinding process for the agents. But the methods of Menge pathfinding were not able to plan long routes with the required level of believability. That's why it was decided to use decision-making methods to split agents' routes into smaller segments to maintain the agents' realistic movement. Each train platform was divided into separate areas, which assign a Temporary Role to agents with defined rules of behavior on the platform. These rules also regulate agents believable spreading across the platforms.

Menge also has some limitations related to the 2-dimensional navigational mesh. But the Olympic park station has several layers which may contain agents; consequently, agents were not able to move from one layer to another. To solve this problem, it was decided to use an area with local roles assignment. Each layer became an independent area, which contained the behavior rules of transferring to the neighboring layers. To find the connection between layers the stairs and escalators were segregated to areas, which contained behavior roles that controlled the agents' movement.

In summary, the believable behavior of big human crowds was achieved by applying the developed decision-making model and combining methods of multi-agent modeling with a game development technologies approach. Each agent in crowd had its one unique character of movement that was defined by his goals, mental and physical state. With the help of the designed model an agent became adapted to the station environment, and the disadvantages of Menge algorithm were compensated using the advanced decision-making technique (See Fig.2).

The results of these simulations were introduced at "Biennale Architettura 2018" 16th international architecture exhibition in Venice, May 2018. Full video sequence of the simulation is available at https://www.youtube.com/watch?v=UNY4z5ftbZA

4.2. Game project Team7

Another example of the system usage is a game that has been developed on the sidelines of Wargaming Academy 2018 [58]. The game project is a collectible card game where an opponent is driven by an AI and acts according to the in-game rules. The project has been developed with Unity Engine [20] and rewritten with C#.

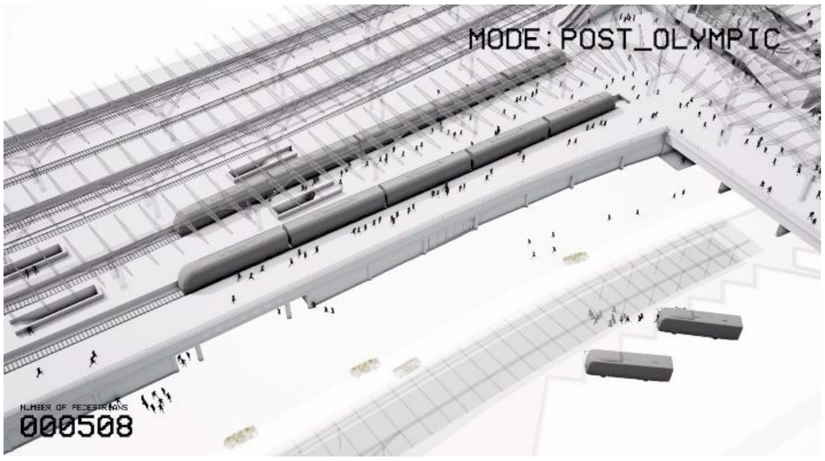

Fig. 2. Olympic Park train station pedestrian mobility simulation

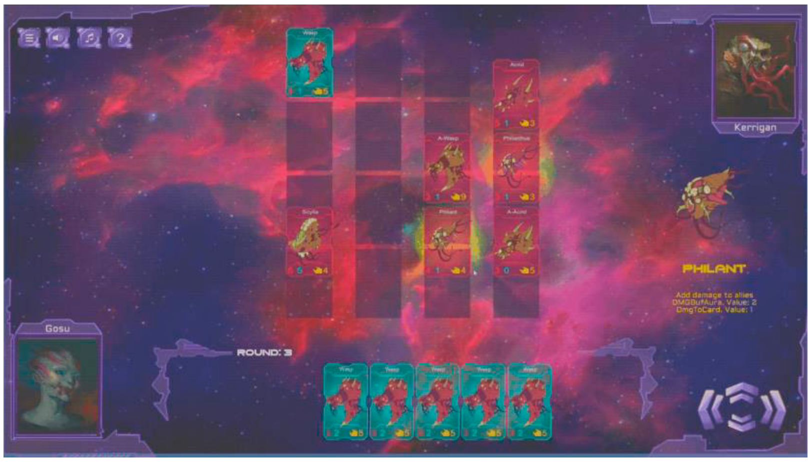

Fig. 3. Game project Team7

Opponents in the game have been implemented to determine their behavior based on different personality traits such as aggression, chaotic nature, accuracy, cautiousness, fearfulness and despotism. Character's actions are strongly restricted by the game rules that allow performing deeper calculations of those. A character can perform three actions: to place a card, to move a card on the field and to use an in-game ability. When the character must perform an action he estimates a profit of every currently possible action and chooses the most profitable one. The profit estimation consists of the evaluation of the power of an opponent's cards and a player's cards. Formulas for the profit calculation based on a character's personality traits are presented below. Equation 5 demonstrates the profit calculation for the "place a card" action, where D – a character's despotism; s^a - the power of a character's cards presented on the field before the action; s_0^a - the power of a character's cards presented on the field after the action; F – a character's fearfulness; s^p - the power of player's cards presented on the field before the action; s_0^p - the power of player's cards presented on the field after the action.

$$U_{placing} = D \cdot (s^a - s_0^a) - F \cdot (s^p - s_0^p) \tag{5}$$

Equation 6 demonstrates the profit calculation for the "move a card" action. A – a character's aggression; $cards_d$ - the number of enemy cards that would be destroyed during the action; $damage_r$ - the amount of damage character's cards would receive during the action; SF – a character's accuracy; M – a character's chaotic nature; C – a character's cautiousness; $cards_l$ - the number of allied cards that would be lost during the action.

$$U_{movement} = A \cdot cards_d - damage_r \cdot (SF - M) - C \cdot cards_l \tag{6}$$

Equation 7 demonstrates the profit calculation for the "use an ability" action. A character has several alternate abilities but is restricted to use only one. In this case an additional profit calculation is performed based on the effects of available abilities.

$$U_{ability} = max_{i=0\ldots n}(M \cdot \left(s_i^a - s_{i,0}^a\right) - C \cdot \left(s_i^p - s_{i,0}^p\right)) \tag{7}$$

An AI with such behavior model can provide a proper challenge to a player. And by using the Behavior characteristics to model opponent's personal traits player got a sense that he plays against real person. The game (See Fig.3) got a place in Wargaming Academy 2018 and got a glowing review from the jury.

5. Conclusions

The decision-making model presented in this paper provides a way to create complex behavior rules for the game characters controlled by an artificial intelligence. With the help of these rules it becomes possible to generate characters with a believable behavior that meets modern requirements for a game artificial intelligence. To achieve the believability, the model uses a combination of the Behavior roles and Behavior Characteristics components. They allow characters to bring more individuality into their decisions and adapt their behavior according to the state of a game environment. In turn, generating unique sets of Behavior Characteristics solves the problem of creating a big amount of agents with a diverse and realistic behavior to inhabit huge game worlds.

The developed model was implemented using game engines Unreal Engine 4 and Unity to create characters for a computer game and for a project to visualize human crowd flows. The projects were presented in several game cups and international architecture conferences including Biennale Architettura 2018.

To reach more believability of characters' behavior the developed model can be complemented by the machine learning methods to bring more diversity into an agent behavior or to modify their decision-making cycle to adapt to the changes in a game state or a player in-game progress. In addition, the model may be coupled by the methods of procedural generation to bring more effectiveness to the generation of characters' personalities or to develop an approach to automatically inhabit game settlements by agents with an artificial intelligence.

Acknowledgements

This research is financially supported by The Russian Science Foundation, Agreement №17-71-30029 with co-financing of Bank Saint Petersburg.

References

[1] Searle J. The Chinese Room Argument. Encycl. Cogn. Sci., Chichester: John Wiley & Sons, Ltd; 2006. doi:10.1002/0470018860.s00159.

[2] Umarov I, Mozgovoy M. Believable and Effective AI Agents in Virtual Worlds. Int J Gaming Comput Simulations 2012;4:37–59. doi:10.4018/jgcms.2012040103.

[3] Gomes P, Paiva A, Martinho C, Jhala A. Metrics for Character Believability in Interactive Narrative In proceedings ICIDS 2013. 2013.

[4] Eccles V, Dissertation BA. Procedurally Generated Background Characters. 2017.

[5] Fernando Bevilacqua. Finite-State Machines: Theory and Implementation 2013. https://gamedevelopment.tutsplus.com/tutorials/finite-state-machines-theory-and-implementation--gamedev-11867 (accessed December 10, 2017).

[6] Macedo BHF, Araujo GFP, Silva GS, Crestani MC, Galli YB, Ramos GN. Evolving Finite-State Machines Controllers for the Simulated Car Racing Championship. 2015 14th Brazilian Symp. Comput. Games Digit. Entertain., IEEE; 2015, p. 160–72. doi:10.1109/SBGames.2015.19.

[7] Miyake Y. Current Status of Applying Artificial Intelligence in Digital Games. Handb. Digit. Games Entertain. Technol., Singapore: Springer Singapore; 2015, p. 1–32. doi:10.1007/978-981-4560-52-8_70-1.

[8] Yannakakis GN, Togelius J. Artificial intelligence and games. vol. 2. Springer; 2018.

[9] Colledanchise M, Natale L. Improving the Parallel Execution of Behavior Trees. 2018 IEEE/RSJ Int. Conf. Intell. Robot. Syst., IEEE; 2018, p. 7103–10. doi:10.1109/IROS.2018.8593504.

[10] Sprague CI, Ögren P. Adding Neural Network Controllers to Behavior Trees without Destroying Performance Guarantees 2018.

[11] Vinyals O, Ewalds T, Bartunov S, Georgiev P, Vezhnevets AS, Yeo M, et al. StarCraft II: A New Challenge for Reinforcement Learning 2017. doi:https://deepmind.com/documents/110/sc2le.pdf.

[12] Liang Y, C. Machado M, Talvitie E, Bowling M. State of the Art Control of Atari Games Using Shallow Reinforcement Learning. Proc 2016 Int Conf Auton Agents Multiagent Syst 2016:485–93.

[13] Torrado RR, Bontrager P, Togelius J, Liu J, Perez-Liebana D. Deep Reinforcement Learning for General Video Game AI. 2018 IEEE Conf. Comput. Intell. Games, IEEE; 2018, p. 1–8. doi:10.1109/CIG.2018.8490422.

[14] Graham R. An introduction to utility theory. Game AI Pro 2014;2:327–342.

[15] Russell SJ, Norvig P. Artificial intelligence: a modern approach. Malaysia; Pearson Education Limited,; 2016.

[16] Yager RR. Multiple objective decision-making using fuzzy sets. Int J Man Mach Stud 1977;9:375–82. doi:10.1016/S0020-7373(77)80008-4.

[17] Frenzel C, Sanneck H, Bauer B. A Fuzzy, Utility-Based Approach for Proactive Policy-Based Management, Springer, Berlin, Heidelberg; 2013, p. 84–98. doi:10.1007/978-3-642-39617-5_11.

[18] Simonov A, Lebin A, Shcherbak B, Zagarskikh A, Karsakov A. Multi-agent crowd simulation on large areas with utility-based behavior models: Sochi Olympic Park Station use case. Procedia Comput Sci 2018;136:453–62. doi:10.1016/j.procs.2018.08.266.

[19] Unreal Engine 4 n.d. https://www.unrealengine.com (accessed May 30, 2018).

[20] Unity n.d. https://unity.com/ (accessed April 23, 2019).