Real-Time Strategy Games Bot Based on a Non-Simultaneous Human-Like Movement Characteristic

Damijan Novak
Laboratory for Real- Time Systems
University of Maribor
Slovenia

Domen Verber
Laboratory for Real- Time Systems
University of Maribor
Slovenia

Abstract ***T*****his paper discusses how to improve the behaviour of artificial intelligence (AI) algorithms during real-time strategy games so as to behave more like human players. If we want to achieve this goal we must take into consideration several aspects of human psychology – human characteristics. Here we focused on the limited reaction times of the players in contrast to the enormous speed of modern computers. We propose an approach that mimics the limitations of the human reaction times. In order to work properly, the AI must know the average reaction times of the players. Some techniques and proposed algorithm outline are presented on how to achieve this.**

Keywords-RTS; AI bot; game behavior; human factors

I. Introduction

Games have developed into rich environments that now offer great sound effects, object interaction with the use of physics simulation, stunning graphics, and last but not least they populate the environment with computer-guided beings and creatures (Non-Playable Characters – NPC's/bot's), which make the virtual world feel more alive. There is only one drawback to the whole story. During decades of game development, the graphical part of a game's engine has evolved far more than the artificial intelligence (AI) module, which is in charge of NPC's behaviour.

We can observe this lack of AI sophistication in many game genres. It can be seen especially in real-time strategy games (RTS) where bots fail to defeat even medium-skilled human players, let alone experienced players, and if they manage to beat a human it is in a very non-human way. We can observe this kind of behaviour in a RTS game where a bot simultaneously uses tens or hundreds of units against a player at the same time and eventually overruns him/her so that he/she loses (brute force attacks), uses game information which is otherwise hidden to the player or by applying cheats (breaking of game rules). The end-result is that game AI becomes more challenging whilst at the same time less human- like. Reason for such results is that a focus of academic research in the AI field is not to offer fun experience to the player, but to try to play the game in the best way possible, which can become frustrating experience for the player if it is too hard to win the game. And frustration is the opposite of having fun, which is the main reason we choose to play the game in the first place.

So our research focused on developing ways of creating human-like behaviour for RTS game AI bots (since bots that play as humans are more fun [1]), whilst maintaining the strength of AI algorithms that other fields (genetic, neural, data mining etc. approaches) provide. The aim of this paper was to research into one specific part of human- like characteristics. Trying to set the ground on how fast a human plays a RTS game, and to then propose movement algorithms that take into account these limitations. The reason for this research was in the fact that computers can perform actions on a millisecond scale and outperform humans in the sheer amount of simultaneous actions. We wanted to limit the number of actions and bring the AI bot operation more to a human level thus operates within the second scale.

The structure of this paper is as follows. In the next section we provide a brief description of real-time strategy games. Section 3 identifies the common characteristics of human-like behaviour. Section 4 presents the challenges of creating human-like non-simultaneous unit movements, together with discussing the limiting factors for algorithms. Section 5 proposes some methods for measuring the reaction times that can then be used as limiting factors. The conclusion is given in section 6.

II. Artificial Intelligence in Real-Time Strategy Games

When a new AI game is being implemented, the developer always bears in mind that a player wants to have fun during the game. This can only be achieved if the game is balanced. Such a thing is hard to achieve because different types of players use different tactics when going up against AI.

A. *Real-time strategy games*

The term 'strategy' comes from the Greek word στρατηγια (strategia) which means general-ship. Generals are at the top of the hierarchy when issuing commands and making decisions that will have an impact on the end-result of the vision being pursued by a general (or someone to whom the general is loyal). Strategy games follow the same principle. One of the most famous strategy games is the game of chess, where two players follow their own strategies through a series of turns in the hope of defeating their opponent. Each player has time to perform actions during their turns. When a player has finished his/her turn (or the turn must end because the turn time is limited), the outcome of the turn is evaluated. The principle is the same for two (or more) players in other strategy games,

DOI: 10.5176/2251-3043_3.2.255

where each player plays the game according to his/her strategy while respecting the rules of the game.

Real-time strategy games (RTS) are a sub-genre of strategy games. The word real-time is added because the times for the turns are so small that it creates the illusion that both players are pursuing actions at the same (real) time. Human players, of course, cannot keep up with the fast-pace of the game, where turns switch within millisecond's so we do not perform an action (or series of actions) every single turn. This does not represent the problem when average-skilled human players are competing with each other, since they all share the equal ground by missing a lot of turns. This paper is focused about the problems that arise when an average human player is competing against a computer, and will be discussed later in the article.

War takes place within a virtual (fictitious) environment (also called a battle-map), where players lay out there strategies through the use of various units' movements. A player usually has only a part of the map visible on his/her screen. In one part of the player's screen there is usually a 'mini-map', which shows the positions of all the friendly units (those of the player and his allies) and part of the visible enemy units. Only part of enemy units are visible because RTS games have fog-of-war (FOW) implemented which by lack of information adds the unknown factor to the game (increasing the difficulty of strategy planning). FOW is the part of a map that a player has not yet explored or does not have any friendly units nearby.

B. *Game artificial intelligence*

If we want to define game artificial intelligence, we must first define the term 'artificial intelligence'. The Oxford dictionary for the word artificial states the next definition »Made or produced by human beings rather than occurring naturally« and for word intelligence »The ability to acquire and apply knowledge and skills«. So by combining both definitions together we can conclude that artificial intelligence is something that was made or produced by human beings rather than occurring naturally and it has the ability to acquire and apply knowledge and skills. The term as we can see cannot be exactly defined, but it can give us a hint of what we want to achieve. Definition of game artificial intelligence on the other hand is slightly different from that of the (academic) artificial intelligence and it follows other goals. The first part of the combined definition stays the same, but the second part is looser. As game AI does not always need to acquire and apply knowledge and skills to look human, it can fake, copy, or just create an illusion or simulate human-like behaviour; as long as the behaviour looks convincing to the human.

III. Related work - identified characteristics of human-like behaviour

There has been extensive research done by different authors who were trying to pinpoint what makes the human style of playing RTS games different to the computer bots play style. In article [2] the aim did not focus on creating a bot that behaved like a human, but rather to find the characteristics of human-like behaviour. Their findings on human characteristics were as follows: simultaneous movements, overall strategy, attack tactics, placement of buildings, level of activity, ability to adapt, build order, player strength, map knowledge, intelligence and creativity, and monotony.

Article [3] had similar goals where authors from the NASA Ames research centre conducted an informal survey of experienced players of the game Starcraft and summarised significant differences between human and computer play into three categories: fine motor control, visual field-of-view, and visual attention. Within the category of fine motor control they identified one critical difference between a human player and a typical computer player - the speed with which commands can be issued. Whilst this speed can be essentially unlimited for the computer, human speed depends on dexterity when controlling the interface. Within the second category, field of view, the inability of a human player to view the full map in detail produces vulnerability within a range of deceptive tactics. In the third category, visual attention, their findings were that the limit of people's ability to pay attention to all available visual stimuli has a variety of effects. Two of more important ones were that a player often fails to detect units that are visible but non-salient (a unit may be camouflaged by similar background, partially obstructed by another object etc.), and the second effect was to delay situation assessment (understanding the nature of the attack).

The third article [4] presented PAR AI believability criteria, where PAR stands for Plan, Act, and React. Within a planning category AI should demonstrate some degree of strategic/tactical planning, be able to coordinate actions with a player/other AI, and not repeatedly attempt a previous, failed, plan or action. When an AI acts it should act with human-like reaction times and abilities, meaning, when a decision has been taken, the AI needs to appear to have natural reaction times, and be able to act with a reasonable degree of skill. The react category states that AI should react to players' presence and actions appropriately, react to changes within their local environment and react to the presence of foes and allies.

All three articles pointed out that one key characteristic of achieving human-like behaviour is the speed at which the bot issues commands. In the first article, the speed was identified in two categories: simultaneous movement and level of activity. The second article called it the speed fine motor control and in the third it was contained within the acting part of the PAR AI believability criteria.

IV. Planning of human-like non-simultaneous unit's movements

A. *Core problem*

In the third chapter, we identified that one of the factors that plays a significant role when showing how the game is played by a human player or by a computer bot, is the speed with which the commands are issued for the movement of units. The problem lies in the different response times between a human and a computer if they are both confronted with the same task. So the main goal of this article was to find solutions to this problem.

In the typical RTS game scenario, the map is filled with different units, which are used to pursue the vision that a general has and act according to the strategy. So if an enemy launches an attack, the counter attack or some other counter-measure must be issued by a general. The strategy level will decide the best tactic and units will be ordered to act according to it. Most times this means the movement of unit or group of units into the right position, and when at the right spot issuing appropriate action (an action can be a shot fired, performance of healing on a friendly unit, etc.). Let's present the core of the problem with an example.

The enemy attacks a friendly building using a group of tanks. The general decides that the building is of great strategic importance and must be defended. So a command is issued to reallocate friendly units in an attempt to save the building. Our concern is how this action will be viewed by the observer of the game. If the general in charge is an average human player, he/she will start clicking on units across the map and giving them orders to move and defend the building. Therefore, he or she will have to travel across the map to find all the units, click on each of them and then issuing them the right orders when they are positioned in the correct places. If units are sparse, he/she will have to repeat this multiple times to gather all the necessary units. To the outside observer the action will look like this. Enemy started attacking the building. After a short while, the player moves first unit(s), moments later the second unit(s) received orders and started moving and so on, until all the unit(s) had been moved. So between the first and last unit(s) movement, a considerable amount of time could have passed. The movement, as we can see, is non-simultaneous – Fig. 1.

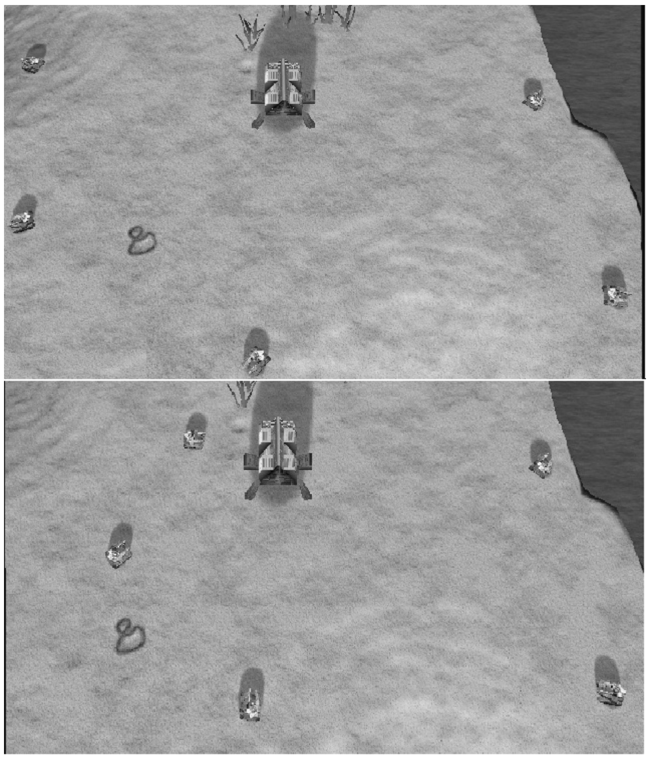
Fig. 1: Non-simultaneous movement of units

The top half of the picture shows the starting positions whilst the bottom half shows the results from the non-simultaneous movements of units after a few short moments.

If an AI bot were in charge of defending the building, the same action would look very different. When the bot wants to relocate the units, all of those units that must respond will start moving at once. The reason for this lies in fact that an AI bot does not have to search for units across the map (it has the units saved as objects in memory along with their exact location) and the issuing of movement commands for all the units usually takes no more than a few milliseconds (the movement interval for a bot is in time span of few milliseconds). Therefore, the movement would look simultaneous to the human eye. Our task was to ensure that the AI bot unit movements would be non-simultaneous as well, so that it would appear more human-like – Fig. 2.

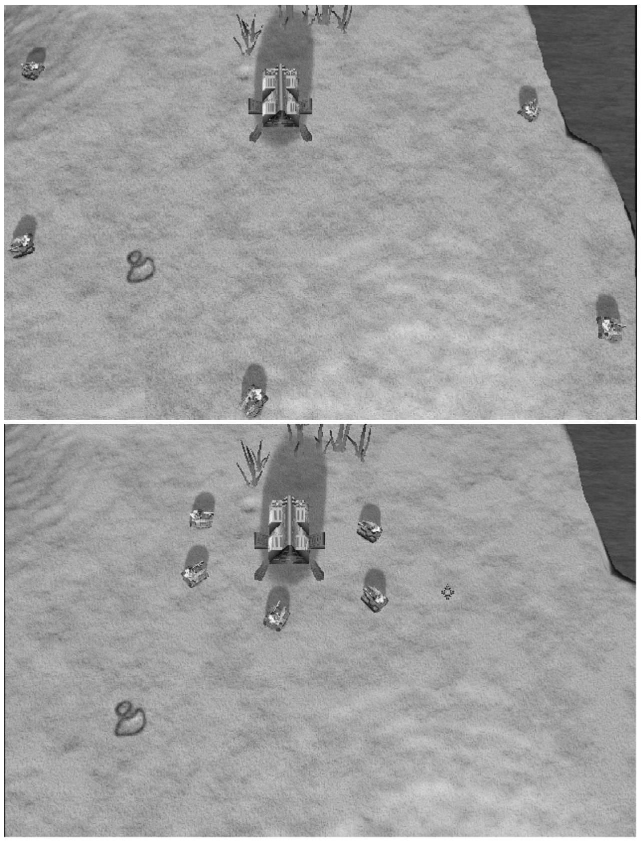
Fig. 2: Simultaneous movement of units

B. Key non-simultaneous movement factors

Two key time-delay factors must be considered and implemented into AI bot algorithms if we want to achieve non-simultaneous movements. The first one is the time-delay between the two movement orders to the units (let's refer it as movement delay - MD), and the second factor is the time-delay of travelling across the map to find the right unit or place to move to (travelling delay – TD). For the second factor, we must also take into consideration the size of the movement visible frame (MVF). With usage of MVF we are limiting the vita of the battlefield. Just like the human player has a limitation when viewing just a part of the battle map and not the whole map. Therefore, if he/she wants to see other part of the map, he/she has to change the view by travelling there.

In chapter 5 we propose methods by which the time-intervals for MD and TD could be set experimentally ((0, max. MD], (0, max. TD]). Maximal MD and maximal TD are the longest reaction times for a human player to perform an action, and if we want to the set the intervals correctly, we need to measure the reaction times of the players. We can set different intervals for different skill sets of players (beginner, moderate, expert, professional etc.).

C. Proposed algorithm outline

There are many different approaches, techniques, and technologies involved in programming an AI bot and our purpose was not to limit the programmer in any way that a bot would fail to deliver functionalities envisaged by him/her when the bot was being implemented. We just wanted to limit the speed of issuing movement commands. So we propose the next algorithm outline (procedure for setting time delays while executing moving orders to units):

```
procedure setTimeDelays (units)
  // get minimal number of MVF's that cover all units
  arrayOfMVF  = getMVFs (units)
  for each  MVF in arrayOfMVF do
    // optional – if we have all units in one MVF or not
    if not ( size (arrayOfMVF)  ==  1) then
      // simulating time of traveling to next frame
      delay(random(0, max_TD ))
    end if
    // get all units positioned in current MVF
    arrayOfUnits = getUnits (MVF)
    for each Unit in arrayOfUnits do
      // if unit performs action outside current MVF
      if outsideFrame (Unit, action, MVF) == true then
        // simulating time when going outside of this MVF
        delay(random(0, max_TD))
      end if
      // simulating time of unit movement delay
      delay(random(0,  max_MD))

      // place for unit actions
      // (pathfinding, attack or defend moves ...)

      //if unit performs actions outside current MVF
      if outsideFrame (Unit, action, MVF) == true then
        // simulating time of getting back to right MVF
        delay(random(0, max_TD))
      end if
    end for
  end for
end procedure
```

If we want to move N units from point A to point B the total cumulative of delays based on the outline algorithm will be (please note that the equation for total sum of delays is set for worst case scenario – meaning, executing body of all if statements):

$$CD = \sum_{i=1}^{n} (TD + \sum_{1}^{u_i} (MD + 2 * TD)).$$

Value n represents minimal number of MVF's that cover all units; value u_i represents number of units in an i-th MVF.

V. Measuring the reaction times of the players

We can examine the reaction times of players during a game with different methods: using controlled experiments, by analysing the players' logs of real RTS games, or by observing the players on-line whilst they are playing the game.

A. Controlled experiments

One way is to build a mock-up environment with all traditional RTS game elements. With this, we can perform different experiments under pre-set conditions. The players are exposed to situations in real games and asked to perform some tasks. After being given some visual or aural cues, they must select an object, move a unit, scroll the screen, etc. We also prepare questioners that are given to the players before and after the experiment. One of the goals of the questioner is to determine the skill level of a player. During the experiments, we measure the player's reaction times. We can run the same experiment repeatedly with different groups of players and/or under different starting conditions. After experiments, we can perform thorough statistical analysis of the recorded data to obtain the average reaction times for some tasks by different player groups.

We have already implemented a prototype of such a mock-up program. Firstly, we considered using implementation with a simple graphic library. Instead of hi-definition textures, we decided to use simple geometric shapes. A screen-shot of the program is shown in Fig. 3.

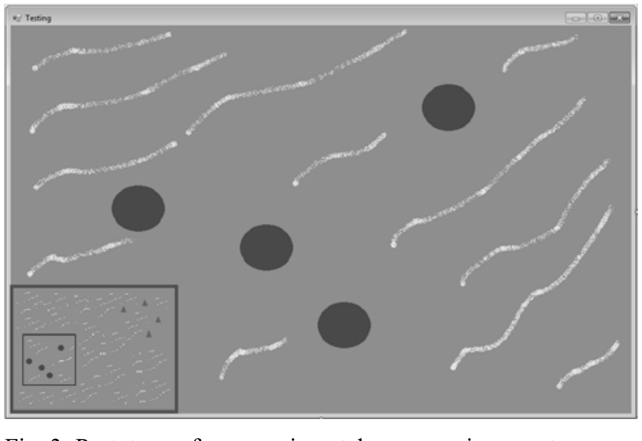

Fig. 3: Prototype of an experimental game environment

The mock-up model is independent of specific games. This brings same benefits and some drawbacks. On the positive side, the results are more general and can be used by the AI universally. On the other hand, the real games may sometimes behave differently than the mock-up. For example, with a different combination of colours or different shapes, some objects may be much more difficult to detect during the game. Therefore, the reaction time of the player would be prolonged. Similarly, during tough game situations, players usually make mistakes and perform poorly.

Secondly, we upgraded the mock-up model with more realistic textures from the real RTS game engine Spring. This way we can simulate different game scenarios while avoiding the complexity of programming for real game engine which is time consuming and still maintaining the most of the RTS aspects and feels. Realistic mock-up model is shown in Fig. 4.

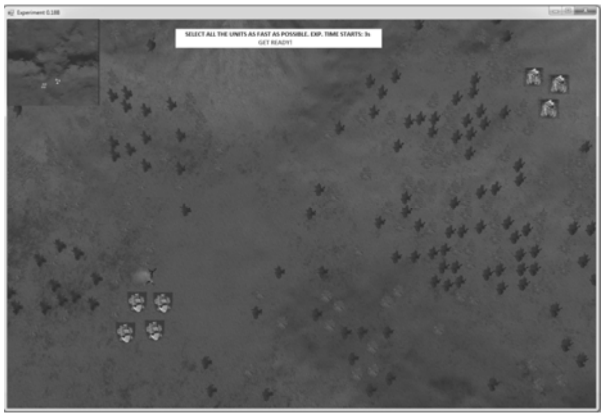
SELECT ALL THE UNITS AS FAST AS POSSIBLE. EXP. TIME STARTS: 3s
GET READY!

Fig. 4: Realistic mock-up model

In the future, we also want to give more attention to other human playing characteristics. To achieve this we will equip the test computers with video cameras and brain computer interfaces (BCI). The cameras would monitor the players' eyes movements, facial expressions, etc. The video feed would be combined with the current screen captures. We believe that this would provide more profound knowledge about players' behaviour while playing games. Something similar was done in [5] for body posture where they measured body lean-angles whilst players were playing a game of chess. BCI was also already successfully implemented in some aspects of gameplay as shown in the article [6].

B. *Analysis of games' log data files*

The second source of data about players' behaviour during games is the game logs. Some games and game engines allow for recording the players' moves during the play. For example we can take a game of Starcraft which is a science fiction RTS game developed by Blizzard Entertainment. This game provides a recording function regarding replays for further review and analysis. Because this game has a huge community, with players ranging from beginners to professional players (some of them even making a living by competing at tournaments for money prizes), we have a huge base of replays at our disposal. These replays are therefore an invaluable source for many researches. [7] [8].

By analysing the logs, we can perform some measurements. However, we cannot get out as much information as with controlled experiments. The logs do not usually record all the parameters of the game and they usually do not store any attributes of the players. From logs we can get data about times when a specific unit or building is first produced. No information about the combats and the outcomes of them are tracked, so only a strategy concerning unit constructions can be inferred. The large amount of data allows for employing different data-mining methodologies, machine learning, etc. Therefore, log files in general cannot give us enough information about the reaction times, but they can be used to measure other characteristics of the players. In the future, we intend to implement complete log systems for our AI engine in order to obtain proper parameters from the game play.

C. *Observing players in real-time during playing*

We can also observe the players in real-time during the game. Usually this would be cheating if the AI were to know all the movements of the opponent. However, during the play, the AI does observe particular movements of the units visible to it and some actions of the player. It observes the time when the opponent's units starts to move, when changing direction, when they fire, etc. With this, the AI may measure the reaction times and adjust its playing capabilities accordingly.

This method can be easily upgraded for controlled experiments. By using the techniques described above, we can observe the behaviour of the player during a real game either when playing with AI bot or with another human player. Of course, this requires some modifications within the existing game engine.

In the part A of this chapter we mentioned that we want to upgrade the test computer with video cameras and BCI. This could allow the game engine (through the use of right interface) to be able to observe the skill and mood of the player and adapt to them to improve the user experience. That's why we started paying attention to the field called affect computing [9]. By trying to recognise the affect state of the player while playing a game, we can use that knowledge so that a bot changes behaviour in more adjusted human-like manner. e. g, if we can recognise that player is showing an affect state of getting frustrated (maybe because he/she lost majority of his/hers units in defending his/her own base instead into more satisfying or rewarding battles), the bot could change overall strategy management by maybe using more defensive tactics or by using units in a more reckless manner (so that more of its units would get destroyed). This way the player could get more satisfaction if he/she gets the feeling that the luck is shifting into his/hers direction. Changes would not be too dramatic, but they could be enough to make game more interesting for the player. This is just one example where affect state of the player would result in using the correct amount of changes to the bot behaviour and so making it more human like.

VI. Conclusion

This paper discussed how to improve the behaviour of AI during RTS games in order to behave more like human players. Using high processing power, the AI can perform several unit movements or other actions simultaneously. This is unnatural for the human players, who are only capable of processing a limited amount of information during a single time-interval. The proposed methodology and algorithms would narrow the gap between these two behavioural aspects.

We also suggested some methods on how to measure human reaction times, which would serve as a reference for the AI. For this, we built two experimental environments using a mock-game element. During the autumn we will start performing these experiments with the help of students from our Faculty.

Besides the human reaction times, other aspects of human behaviour are important for more likable AI bots in games. As on-going research, we will pursue the study of these aspects in the future and also gave more attention to the field of affect computing which could offer serious impacts on the overall positive experience for the player while playing RTS games.

REFERENCES

[1] B. Soni, “Bots trained to play like a human are more fun”, in Proc. IEEE Int. Joint Conf. Neural Netw., Hong Kong, June 2008, pp. 363–369.

[2] J. Hagelbäck, S. J. Johansson, »A Study on Human like Characteristics in Real Time Strategy«, Computational Intelligence and Games (CIG), 2010.

[3] M. Freed, T. Bear, H. Goldman, G. Hyatt, P. Reber, J. Tauber, »Towards more human-like computer opponents«, AAAI Technical Report SS-00-02, 2000.

[4] D. Livingstone, »Turing's test and believable AI in games«, Computers in Entertainment (CIE) - Theoretical and Practical Computer Applications in Entertainment, Volume 4 Issue 1, January 2006.

[5] J. Sanghvi, G. Castellano, L. Leite, A. Pereira, P.W. McOwan, A. Paiva, »Automatic analysis of affective postures and body motion to detect engagement with a game companion«, HRI 2011 - Proceedings of the 6th ACM/IEEE International Conference on Human-Robot Interaction, 2011, pp. 305-311.

[6] J. J., King Li, T. A. Monsod, P. J. Hao, G. L. Matias, J. R. Cheng, N. Guloy, “Towards the Development of an Affect- Sensitive Game”. Philippine Computing Journal Dedicated Issue on Affect and Empathic Computing, 6(2), 2011, pp. 42-47.

[7] B. G. Weber, M. Mateas, “A data mining approach to strategy prediction”, In Proceedings of the IEEE Symposium on Computational Intelligence and Games, 2009, pp. 140-147.

[8] J. Hsieh, C. Sun, “Building a player strategy model by analyzing replays of real-time strategy games”, In proceedings of the International Joint Conference on Neural Networks. Hong Kong, China:IEEE, 2008, pp. 3106–3111.

[9] R. W. Picard, “Affective computing”, MIT Press, Cambridge, 1997.

Damijan Novak graduated in 2011 in computer science at the Faculty of the Electrical Engineering and Computer Science at the University of Maribor. He is currently a teaching assistant and post-graduate student at the faculty where he received his academic degree. His current research subjects are data mining algorithms and artificial intelligence in computer games.

Domen Verber is an assistant professor at the Faculty of the Electrical Engineering and Computer Science at the University of Maribor. His main research subjects are Ubiquitous computing, High performance computing and Computer games. He has published numerous papers in international journals, books, and conference proceedings.