Praca dyplomowa – magisterska

Metody projektowania postaci współpracy z graczem
Methods for designing DNPC (dependent non-player character)

inż. Dominik Wojnar

słowa kluczowe:
Non-player character
Collaboration
Agent architecture

krótkie streszczenie:
W niniejszej pracy zostało opracowanie przeglądu istniejących metod projektowania niezależnych postaci w grach wideo, ze szczególnym uwzględnieniem postaci współpracujących z graczem. Przedstawiono na przykładzie popularnych gier podstawowe mechaniki współpracy. W ramach pracy przygotowano prosty system wieloagentowy do gry StarCraft II, aby kontrolować jednostki w skoordynowanym sposób.

Ostateczna ocena za pracę dyplomową

Do celów archiwalnych pracę dyplomową zakwalifikowano do: *

a) kategorii A (akta wieczyste)
b) kategorii BE 50 (po 50 latach podlegające ekspertyzie)

* niepotrzebne skreślić

Wrocław 2019
CONTENT

List of figures ................................................................. 3
List of listings ............................................................... 4
List of table ................................................................. 5
Introduction ................................................................. 7
  Description of the problem ........................................... 7
  Objective of the Thesis ................................................. 7
  The scope of the Thesis ............................................... 7
1. About games ............................................................ 8
  1.1. Differences between the game and the play ..................... 8
  1.2. Game mechanics .................................................. 9
  1.3. Progress curve ................................................... 10
  1.4. Conclusions ...................................................... 13
2. The measure of cooperation .......................................... 14
  2.1. Strategy .......................................................... 15
  2.2. Cooperation purposes ............................................ 15
  2.3. Conclusions ...................................................... 16
3. What are NPCs ........................................................ 17
  3.1. Knowledge about the NPC world ................................ 17
  3.2. Decision systems ................................................ 17
    3.2.1. What are the decision-making systems .................... 18
  3.3. Development of NPC ............................................. 18
    3.3.1. Agent architecture ......................................... 19
    3.3.2. World representation ....................................... 20
    3.3.3. Event analysis ............................................... 21
    3.3.4. Heuristic FSM ............................................... 22
    3.3.5. Goal–oriented planning ..................................... 23
    3.3.6. Multi-agent system ......................................... 23
  3.4. Communication methods ......................................... 23
    3.4.1. Gestures made available by the game engine .............. 24
    3.4.2. Text commands .............................................. 25
    3.4.3. Voice commands ............................................. 26
    3.4.4. No communication ........................................... 27
  3.5. Conclusions ...................................................... 27
4. Case studies ................................................................. 28
  4.1. Conclusions ............................................................ 28
5. Designing independent characters and immersion .................. 29
  5.1. Needs of the characters ............................................. 29
  5.2. Proposition of implementation of Freud’s model ............ 30
6. Do characters get better results by cooperating with others or without cooperation? .................................................. 33
  Analysis of results ....................................................... 36
    Average results ....................................................... 37
    Quantiles .............................................................. 37
  Conclusions ............................................................. 38
Summary ............................................................................. 40
Appendix ............................................................................ 41
A. Performance and Optimization ........................................ 42
Bibliography ........................................................................ 44
LIST OF FIGURES

1.1. Progress curve for games with a difficult start. .......................... 11
1.2. Progression curve for games with linear increase in difficulty. .......... 11
1.3. Progression curve for role-playing games. ................................. 12
1.4. Progression curve for games referred to as easy to play, hard to master. .. 12

3.1. Agent architecture graph. .................................................. 19
3.2. Agent architecture graph. .................................................. 20
3.3. Example final state machine for event analysis ........................... 21
3.4. Clustering and refinement example. ....................................... 22
3.5. Schematic diagram of Goal-oriented planning. .............................. 23
3.6. Multiagent system architecture that uses user gestures. ................. 24
3.7. Gesture menu from the *League of legends* game ........................ 25
3.8. Multiagent system architecture that uses text commands. ............... 26
3.9. Multiagent system architecture that uses voice commands. .............. 27

6.1. The initial position of units ............................................. 34
6.2. Distribution of values in the studied groups. ............................. 38
LIST OF LISTINGS

6.1 Overwriting the addition operator of the *Genome* class. Source: own . . 33
6.2 *gen_elem()* method of the *Genome* class responsible for drawing a command for a multiagent system. Source: own . . . . . . . . . . . . . . . 35
LIST OF TABLE

5.1. Factors affecting the satisfaction factor in individual games . . . . . . . . . . . . . . . 30

6.1. Statistics of the characters used in the study . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34

6.2. Implemented commands . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35
ABSTRACT

The purpose of the work was to develop an overview of existing methods for designing independent characters in video games, with particular reference to characters cooperating with the player. Popular games implement the basic elements of the logic of the characters cooperating with the player. As part of the work, a simple multi-agent system for the StarCraft II game was prepared to control units in a coordinated behavior.
INTRODUCTION

In role-playing games often appear alongside the protagonist of the other characters. It is not always enough that they were passive observers. In more complex games, a player may be a member of a team consisting of non-human controlled characters. The development of artificial intelligence increases the capabilities of the characters controlled by the computer, and facilitates their design.

DESCRIPTION OF THE PROBLEM

On the market there are very few games in which well-presented NPCs. Despite spending more time with them than with opponents, cooperation with them is vestigial and boils down to simple game mechanics such as moving around, using a weapon or carrying out scripted dialogues.

OBJECTIVE OF THE THESIS

The main objective of this work was to develop a review of existing design methods as independent cooperating with the player. The second goal was to propose a new approach to modeling the needs of independent characters in order to increase the realism of video games.

THE SCOPE OF THE THESIS

This paper describes the methods of designing independent characters, with particular emphasis on collaborating characters. In addition, the methods of communication between the multi-agent system and the player are described. Also, the basic concepts are clarified, such as cooperation, strategy and progression curve.
1. ABOUT GAMES

Let us ask ourselves the question at the beginning:

*Is the Hatoful Boyfriend computer program a game?*

The answer to this question is complicated, because answer can be both positively and negatively. *Our major problem is the actual situation of the study of traditional games: lack of clear definitions and theories; more functionalist approach rather than formalist; fragmented analysis from different disciplines.*

Games have a result: they define a winner and a loser; plays do not. The game theory does not describe some single-player games, because the players are supposed to be more than one. Not all game have any characters or even conflicts. a perfect example is *Tetris*. Nobody competes with anyone (within one game).

### 1.1. DIFFERENCES BETWEEN THE GAME AND THE PLAY.

The games are described by game theory or ludology. The play can be described only by ludology. Depending on the field of perception, the game can be defined differently. Unlike fun games, they have joy. a simulator of armed conflict, which is a game may not be fun. a simple play can also be listening to music. This is not a game because it does not require a strong interaction unless the music is interrupted (ie also repeated or changed), the recipient’s attention is invalid. The game requires strong interaction [10]. Fun does not have a functional aspect[12].

The ludology definition will be used to describe single player games. According to Classic game model presented by Jasper Juul (2005) a game is:

1. a rule–based formal system;
2. with variable and quantifiable outcomes;
3. where different outcomes are assigned different values;
4. where the player exerts effort order to influence the outcome;
5. the player feels emotionally attached to the outcome;
6. and the consequences of the activity are optional and negotiable [13];

Dr Tomasz Majkowski in his article *Gry wideo i kultura autentyczności* on the subject of video games and the culture of authenticity suggests that the higher the complexity of the game, the more important is its interactivity[16].

---

1 Lalande, 1928, quoted in: Frasca, 1999, section Play and game, paragraph 7[8]
Games defined by game theories and through ludologies are not separate collections. Ludology defines the concept of meaningful play (meaningful play) as a game that has a result\cite{21} and when the relationships between actions and outcomes in a game are both discernable and integrated into a larger context of the game\cite{31}. The game theory assumes that players are playing to win. Ludology puts the emphasis on fun - the pleasure of the game. A game based on game theory is a special case of a play that is defined by ludologies.

The game theory does not describe some single-player games, because the players are supposed to be more than one. Not all games have any characters. A perfect example is Tetris. Nobody competes with anyone (within one game).

The outcome of the game is a numerical measure that is used to assess the player’s skills. How to evaluate the quality of intelligent NPCs?

According to J. M. Bocheński a categorical basic order is: "Live so that you may live a long and well you prosper" \cite{2}. The philosopher in a short handbook has dozens of laws that describe wise behaviors. It would seem that it is enough to determine if all the criteria are met and the verdict can be issued on the basis of this: whether the character behaves wisely.

The first problem turns out to be whether all orders have been met - they are written in natural language, e.g. "3.22 Use the moment" or "6.1. Keep moderation in mind". Another problem may be the incomplete set of rights. Therefore, the philosopher assumed that the first law, which is a ruthless and fundamental order, should be considered its superiority and universality. All moral injunctions come from other moral precepts, or are obvious. If it is possible to derive one law from another, it is much harder to implement to make it obvious. A philosophical approach can help in creating a coherent model of decision algorithms, but it cannot act as a character intelligence criterion.

The advantage of this solution is that the criterion under consideration works well in human life. This criterion, however, is immeasurable in a hasty manner – it would seem that in order to measure in this way the quality of intelligent NPCs should be carried out the survey for the players who have to deal with these characters.

1.2. GAME MECHANICS

It is clear that the creation of the model is equivalent to the creation of restrictions. However, the use of models does not have to convict a designer for repeatability, but only allows drawing common elements for many characters. Thanks to this modeling can be just as creative, and mechanics can be more accurately implemented. Even characters with identical mechanics and features can differ significantly through the way they perform, as well as the role they play in the plot. An example
of this is the figure of a blind orc from the game *Tibia*, which could be found behind the northern border of the first city. He sells items using standard mechanics, just like all other NPCs by talking. The significant difference was not only the appearance but also the language. To make a purchase, the orcs language should be used.

### 1.3. PROGRESS CURVE

Before defining the concept of the game result, the concept of curve progression should be introduced. Determining the appropriate level of difficulty is not easy. If the game proves too difficult, the player will be frustrated and may end up with a bad impression of the game. On the other hand, if the game is too easy then the player may become bored and quit before the game design has really had a chance to really show off its potential. As the player puts more time into the game, he will become increasingly adept at the challenge it offers, so the game must therefore increase in difficulty over time in order to stay interesting. The speed at which the difficulty increases should ideally define a curve, gradual at the beginning but increasing at a steady rate until the end of the game.

The problem game designers face is the definition of the difficulty curve, a term used to describe the progression of challenge from the beginning of a game to the end. This is a measure of the change in difficulty level relative to the progress in the game. In RPG (role-playing game) the progress can be defined as the players level, in survival games, it is the duration of staying alive and in case of PvP-like games it depends on the position in rankings. Progression curves are different for different species, however, you can distinguish the four most common types:
Fig. 1.1: Progress curve for games with a difficult start.

Image: own, based on http://www.davetech.co.uk/difficultycurves

Fig. 1.2: Progression curve for games with linear increase in difficulty.

Image: own, based on http://www.davetech.co.uk/difficultycurves
Fig. 1.3: Progression curve for role-playing games.


Fig. 1.4: Progression curve for games referred to as easy to play, hard to master.

1.4. CONCLUSIONS

Basic concepts such as: game, fun and difficulty of the game have been discussed. In the following chapters will be agreed what is the result, which is a measure of coping with the difficulties of the game. The concept of the game also will be used by the proposal for a new approach to modeling needs.
2. THE MEASURE OF COOPERATION

Not every game has a finale result. The state of the game is a set of states of game components. The state of game can be treated as a result. It seems that all games with states have score, but for the game to be significant result must be comparable. The result should be a direct determinant of making good decisions by the player or the effectiveness of using the game mechanic. Player’s losses related to randomness or plot should not affect the result[28]. The state of each game can be written in the form of a number, but not always this number can be interpreted as a result – the answer to the question whether the player handled well. In this work, the game will mean a game with a final score that can be saved in the form of a positive number, the higher the better the player advised.

To consider cooperation in a scientific way, one should find a measure for it. The easiest way is to calculate the average cost-effectiveness ratio defined as:

\[ Q = \frac{\sum(P_t)}{\frac{n_t}{n_a}} \]

(2.1)

Where:

\[ \sum(P_t) \] – is the sum of results of players who cooperated with each other

\[ n_t \] – is the number of results of players who did cooperate with each other

\[ \sum(P_a) \] – is the sum of the results of players who did not cooperate with each other

\[ n_a \] – is the number of results of players who did not cooperate with each other

The Q coefficient is the ratio of the average results obtained as a result of cooperation by results without it. Certainly the coefficient can be useful for normal distribution results and similar deviations in games with a linear progression curve.

The factor Q has a very important disadvantage: it concerns the average. It is impossible to determine which strategy should be chosen by the players in order to obtain a higher score. The term strategy will be defined in the next chapter.

To compare two sets of results, a parametric test should be used (if possible), eg
Student’s t test. The first thing to make sure is the test groups have a normal distribution. Due to the characteristics of the studied group, it may turn out that the inclination coefficient does not allow the distribution to be considered normal. This skewness may occur due to the uneven level of difficulty of the game described by the progression curve. If the groups do not have a normal distribution, a non-parametric test should be used. One of the tests that is suitable for this is the Mann-Whitney U test. When comparing more than 2 groups, a post-hoc analysis should be performed. In the absence of normal distribution, Bonferroni correction can be used. Analysis of each pair separately when using this patch is conservative.

In scientific works it should be remembered that the average profitability coefficient describes only average values, so it is not suitable for research on the quality of adopted strategies.

2.1. STRATEGY

According to the definition proposed by Jan Franciszek Jacko, the strategy is:

1. a repetitive way of acting
2. which is intentional (has a specific purpose or goals),
3. which increases the efficiency (effectiveness) of this activity,
4. which increases the efficiency of this operation,
5. which is a designated and consistent set of rules,
6. which takes into account possible variants of the development of the situation – all or the most probable ones[12].

It is worth noting that the strategy contractor does not need to know its goals. The last point takes into account that sometimes all possible solutions can not be predicted. Strategies that take into account all options are full strategies, others are incomplete strategies. Strategies consist of tactics that serve to achieve specific goals. It should be noted that the author of this strategy definition does not decide its truthfulness.

2.2. COOPERATION PURPOSES

In zero-sum games, all players compete against each other to get a higher win. It seems that in other significant games players will always try to gain an advantage over other players. However, as in real life, in games people sometimes help each other. Sometimes the game mechanics requires the player to protect computer-controlled characters (so called escorts). Individuals can also help. One of the first games in which it was so is Another World from 1991. Half-Life 2 has the form of a mechanical dog. It selflessly helps the main hero. In turn, a pig character
from the Do not starve game series helps the hero only when they are fed. Even in cooperative games there can be competition. Especially if the points are allocated to the players separately (as in Chicken invaders) and not jointly (as in Heroes of Might and Magic III).

Cooperation can be defined as morally positive behavior, and in particular:

1. An attempting to stop or reduce the effects of events or behaviors that may lead to someone’s suffering.
2. Enabling others to achieve wellbeing.
3. Lack of acceptance for causing harm to others.
4. Expressing a positive attitude towards other people[4].

2.3. CONCLUSIONS

This chapter shows what the outcome of the game should be, in order to be able to measure cooperation. Also introduced the concept of strategies that will be used in the study. The question was asked why co-operation exists. This will be developed both in the study and in the proposal of a new approach to the design of the needs of the characters.
3. WHAT ARE NPCs

Not all players are people. In the single player games, it is possible to cooperate with the NPC (Non–Player Character). NPC is any character in the game world, which is managed by computerized Artificial Intelligence. The independent characters are, among others, team members (unless they are guided by players), characters associated with the story, shopkeepers, passers-by, as well as all kinds of opponents[15].

3.1. KNOWLEDGE ABOUT THE NPC WORLD

It is obvious that character decisions must be tailored to the task they are to perform. Thus, decision systems for FPS games are not useful in RTS games, although as will be shown below, the decision algorithms may be similar.

Of course, mere knowledge of the state of the game does not guarantee a good result. The game with relatively simple rules is the Capture the flag mode in Team fortress II. This mode introduces new victory conditions. To win, a set number of times (3 by default) must be taken over by intelligence material (often called a flag) and taken to your base. You can not refer the flag if the enemy also took over the flag. To be able to refer the flag, you must pick it up from your opponent[26].

Although it does not follow directly from the rules of the game, moving the entire team to attack the enemy base, where the object is to be intercepted. At that time, the strengths of defense-oriented characters are not used. Knowledge about other players is necessary to choose the right tactics: their condition, plans, opportunities and risks. This knowledge is needed to create a decision system.

3.2. DECISION SYSTEMS

The Britannica online encyclopedia lists 3 features of Rational decision-making:

1. self-interested
2. purposeful
3. efficient[22]

Business decision systems are created to solve specific problems with greater efficiency than using manual planning. At the current level of knowledge, the question of self-interested computers should not be answered[27].
The main problem in the processing of information is not only that, decisions are made on the basis of knowledge, but that information may be uncertain, incomplete inaccurate and ambiguous[25]. Advanced decision systems are able to undermine the truth of the value of the input vector, and the facts by which they determine the decision. The probability of occurrence of a specific configuration of the input vector’s value or the system state and the constraint base rule is used.

3.2.1. What are the decision-making systems

Decision systems are expert systems that make a decision based on the input data vector and rule base, which is passed to the actuators. In contrast to decision support systems, decision systems do not require user interaction. Thanks to this autonomy they operate much faster, but they should only be used in applications where they do not make decisions about surviving people. It seems, therefore, that characters in video games can be controlled by means of decision systems.

The decision system is therefore all game engine modules responsible for controlling independent characters.

3.3. DEVELOPMENT OF NPC

Youichiro Miyake (lead AI researcher at Square Enix) noticed: intelligence is the answer to the environment. Therefore, intelligence is required with the complexity of the environment[19].

The logic of independent characters is implemented depending on the challenges faced by these characters. In the only movement mechanics of characters cooperating with the player one can distinguish several levels of technology advancement.

The simplest possible to implement character control logic is to repeat the player’s movement. It occurs mainly in older games such as Rachet & Clank or Kingdom: New Lands (dog).

Higher level of development is achieved by adding functionality to circumvent obstacles. There are a lot of games in which characters go for the hero and autonomously avoid obstacles, because it is relatively simple to implement. In the Tibia and World of Warcraft online games, the functionality of automatically following another player has been added. In World of Warcraft, however, the game engine did not care about obstructing objects, which meant that the player behind the target could be stopped by a protruding trunk or not very high stone. Another, quite obvious stage is the use of alternating mechanics to follow the player and control using a set of commands (the so-called script). This is a trade-off between the time needed to implement complex behaviors to the game engine and the sophistication of character behavior unattainable with simpler methods.
Half-life 2 ep.2: While it may seem that the Advisors do not possess any actual AI, this is in fact false. In the Source SDK, file `npc_advisor_shared.h` includes this line: `#define NPC_ADVISOR_HAS_BEHAVIOR 0`. If set to 1 and compiled, the Advisor will pick up objects with its telekinesis and throw them at the player, as well as forming a makeshift shield with any physics objects at hand.

The highest stage of development is to create a coherent and effective system that allows to perform all actions by a computer-controlled character offered by the game engine for the player.

Youichiro Miyake in his presentation at the Korean Game Conference KGC2007 presented the development of methods for creating independent computer characters. It is worth noting that the next stages of development not only do not negate the previous achievements, but they base them or require them.[20]

### 3.3.1. Agent architecture

The simplest and basic object the agent has distinguished is. An agent is a unit that processes knowledge about the game world and makes changes in it.


Therefore, each agent has a Sensor, through which it receives information from the environment. The information received is processed and a decision is made. This decision is implemented by effecter.
The figure 3.3.1 shows the flow of information in the game engine. The green elements indicate the processes taking place in the agent, and the pink arrows indicate the time and general direction of the data.

The logic of agent information processing can be complicated and have states (memory). Youichiro Miyake in the C4 game to implement the behavior of the dog Duncan used the tree of knowledge, or graph of objects that occur in the game. When an object was detected, the representation of the given object was added to the character’s memory together with the certainty that this object was recognized. Information extracted and classified through percept tree is stored in working memory sequentially. Sequence behavior allows for prediction. If the prediction turns out to be unsuitable for the observed events, Surprise takes place. Object recognition errors are similar to mistakes made by people. This makes the reasoning mistakes accepted by the player because they are immersive. Thanks to state observations, machine learning can be implemented.

3.3.2. World representation

A very important step needed to develop methods for creating independent characters is the autonomous processing of information from the game environment. Information can be the position of players, terrain obstacle matrix, light sources or even a set of 3D map objects. This information is transformed by the element of the game engine into a simpler form. For example: a 3d map grid can be turned
into a tree of possible paths along with information on lighting. Thanks to this, each character will choose the best path, taking into account that it will not be noticed by others.

3.3.3. Event analysis

The next stage in the development of independent characters is to add another layer of abstraction: Situation Analysis while processing information received from the game engine. Each character interprets the received information in its own way and can emit events predefined by the programmer. Event handling allows better planning of character movements and simplifies behavior design. The state machine can be used to handle these events.

![Event Analysis State Machine](http://halo.bungie.org/misc/gdc.2002.haloai/talk.html?page=21&fbclid=IwAR1QZauRPUy_K17pVtA62VAx0ikwZMd-vXz0DmwG0CZ6UJBTf7zV1mWk1F8)

Fig. 3.3: Example final state machine for event analysis
3.3.4. Heuristic FSM

Sometimes a simple state machine does not simply describe the complex states required by designers.

For example: in Rimworld, characters can transport other people. Depending on whether they are opponents, neutral characters or allies and whether a transported or transporting character requires medical help, another action will be made.

For N states can be \( N^2 \) possible Transitions. For several state machine states the number of transitions is acceptable, but with the increase in the number of states of the complexity of the system is growing rapidly. Another problem is the addition of subsequent statuses or the removal of existing ones. [6]

The solution is hierarchical state machines (HFSM). This technique on the design of such a state machine that each state can store another state machine. Can be eliminated in repeated transitions using clustering and refinement.

![Fig. 3.4: Clustering and refinement example.](https://web.stanford.edu/class/cs123/lectures/CS123_lec08_HFSM_BT.pdf?fbclid=IwAR004eZTys36u_pzIjhdIOwHCQ-APfquzg6fVc9msxSKMf18IEjtmbiD1PE)

Four types of nodes:

1. Root node – no parent, one child
2. Composite node (Control flow) – one parent, and one or more children
3. Leaf node (Execution) – one parent, no child (Leaves)
4. Decorator node (Operator) – one parent, one child

HFSM combines hierarchy with programming-by-difference, which is otherwise known in software as inheritance. As class inheritance allows subclasses to adapt to new environments, behavioral inheritance allows substates to mutate by adding new behavior or by overriding existing behavior. State nesting introduces another fundamental type of inheritance, called behavioral inheritance.
HFSM can contain 2 special states: sequence and selector which uniquely indicate what transitions between further vertices should be possible. Game Halo 2 has 3 layers: character, situation and terrain.

3.3.5. Goal–oriented planning

Goal–oriented planning is a method of creating AI along with decomposing the problem. Modeled the character creates a hierarchy or series of problems, and then solve them accordingly. For video games, developer must prepare some goals for AI, and AI select suitable one of them.

![Fig. 3.5: Schematic diagram of Goal-oriented planning.](http://igda.sakura.ne.jp/sblo_files/ai-igdajp/kgc/YMiyake_KGC_2007_11_9.pdf)

Planning has two methods:

1. Chaining: Planner chains actions automatically.
2. Hierarchical: a big goal calls some goals, and its goal calls smaller goals

3.3.6. Multi-agent system

It is a method used to connect autonomous characters into a team. Based on the events that have been extracted by individual characters, the supervisory system sets strategic, tactical and behavioral goals. Simple commands are created based on these goals. Depending on the implementation, the supervisory system can send very simple commands, or those that the agent will process to many smaller commands.

3.4. COMMUNICATION METHODS

The communication is an important aspect of cooperation. Thanks to it, players are able to obtain much better results[18]. As shown in the second chapter, it means that players are doing better. In games where the cooperative aspect counts, there is
usually the possibility of communication between players. There are various methods of communication. In this study three approaches will be discussed:

1. Gestures.
2. Text communication.
3. Voice communication.

Games in which the NPC cooperate with the player should have the architecture of the stack:

![Multiagent system architecture](source: own)

**Fig. 3.6: Multiagent system architecture that uses user gestures.**

3.4.1. **Gestures made available by the game engine**

Some games have commands that the player can call. Among the games that such commands are found is *League of Legends*. Each player can send to his team any of the 4 signal-markers that will show up in a specific place. Thanks to this, players can show that the enemy has disappeared from view, that the player will go to a place, that the place is particularly dangerous or that the need for support. Players can play with the NPC, but they do not use these tags.
Another game in which commands are available, through which communication can be used predefined hotkeys is *World Of Tanks*. They are assigned to the function keys:

- F2 – Attack!
- F3 – Back to base!
- F4 – Follow me!
- F5 – Affirmative!
- F6 – Negative!
- F7 – Help!
- F8 – Reloading!

Any place on the minimap can be selected. This communication is used only by players, because there is no NPC in this game.

Unambiguous predefined commands by the game engine can be transferred to the supervisor of the multi-agent system.

### 3.4.2. Text commands

Communication between the player and the multi-agent system can be done by sending text commands. In order for the multi-agent supervisor to emit commands based on the specified text, the game engine must extract keywords. To facilitate this task, the words should be replaced with the most similar and stemmed.
3.4.3. Voice commands

Modern network games enable voice communication with other players. In addition to the game *Tom Clancy’s EndWar*, the author of this work has not found any other game that allows to perform movements with the very voice. *Tom Clancy’s EndWar* is a real-time strategy in which the player controls 3 types of troops: tanks, helicopters and transports. To control the game, the player must issue a command consisting of the name (number or type) of the unit, the name of the command and the target.

To extract a command from spoken text, use the speech recognition system (ASR). It must be a real-time system. To speed up the processing, use a small dictionary (less than 100 keywords). Speech recognition applies only to the player, so the whole system is dedicated to one speaker and continuous speech recognition, not individual words. The commands can be created from the recognized words, as described above.

Speech recognition is a task that requires high computing power. Due to the dynamic gameplay in most computer games, a system with a high processing time would be unacceptable. The commands used in the system must be simple, logical and consistent, they should also be reflected in reality. The correctness of the system operation should be as high as possible.[30]
3.4.4. No communication

Even if there is no communication with the player, the game engine can predict the player’s next moves. As with speech processing, the game engine with Markov’s Hidden Chains can ”guess” the player’s intentions. It should be noted here that the player must behave in a consistent and rational manner.

3.5. CONCLUSIONS

This chapter shows the methods of NPC design in the form of agents. It also presents what the multi-agent system is and how the player can communicate with him to facilitate cooperation. These concepts are frequently used in the study.
4. CASE STUDIES

1. Alyx Vance – *Half-life 2* Featurefully is a friend of the main character Gordon Freeman. During the game, he moves with him. Sometimes he follows him, and sometimes he overtakes him, or goes to the place where he has to go on a plot. He shoots when he sees the opponent. Like other characters from *Half-life 2*:
   — will avoid grenades
   — will take cover when weapon is changed
   — will retreat to ally character
   — will lob grenades
   — will take cover to reload
   — will pick up health kits
   — will change to melee at close range
   — will verbally situational communicate with the player and likewise with enemies interacting with each other
   The last point is a facilitation for the player. Thanks to this, he knows whether he can charge or defend a companion who reloads his weapon.

2. Visitors – *RimWorld* – When one person from the group is attacked, the whole group will enter the fight. The groups in which most people die will seek to escape. This means that a multi-agent system has been implemented.

3. Mechs – *Chromehounds* – the game offers an advanced multi-agent system. The team can set goals such as:
   — Protect Friend
   — Rescue Friend

4.1. CONCLUSIONS

All presented characters use the agent’s model and to what extent the communication mechanism. Thanks to this, all presented functionalities are possible for them. The complexity of these tasks is significant. It should be noted that characters come from real-time games.
5. DESIGNING INDEPENDENT CHARACTERS AND IMMERSION

The issue of what and what creates the character of a computer character is coming forward. Emphasis will be placed on human figures, or on characters about human psychology. A counter-example can be GLaDoS from Portal 2, which was a human, but by turning into a specific cyborg lost some of its human nature (it is guided by instincts “itching”).

It would seem that mechanics are the most important method of creating characters because: The game Loneliness\(^1\) does not have a story, and the main character’s figure exists, there is one and we control it. However, the game Loneliness is a special case of the game, where game mechanics is a metaphor for the story. More about this game says James Portnow from the Extra Credits channel in the episode Mechanics as Metaphor – And How Gameplay Itself Tells a Story – Extra Credits and Mechanics as Metaphor – II Creating Narrative Depth\(^2\).

It is sometimes seemingly masked giving character to the whole type to which a given unit belongs, for example, canoners, as the vast majority of Warcraft III characters have a unique (one for the whole type) set of jokes, jokes, icons and an animated portrait. However, it is much better to create characters who have their own story and their own desires. An excellent example of a strategic game that combines the uniqueness of characters on a story level with the universality of behavior is the game RimWorld, which at the time of creating this document is still in the alpha phase. Characters are generated procedurally, not only statistics are drawn, but also names, factions and stories. Generating a character in a procedural way usually serves to create characters that are less significant to the story. This method was used in the game Skyrim\(^24\).

5.1. NEEDS OF THE CHARACTERS

In different games the needs of the characters are expressed differently. In many games, the only value visible to the player is the number of hit points. Much less game developers implement such mechanics as: hunger, thirst, satisfaction, need for beauty, starch, consciousness, pain or hygiene needs.

\(^1\) http://www.necessarygames.com/my-games/loneliness/fash
\(^2\) https://www.youtube.com/watch?v=pP_qNm-96Dc
### Table 5.1: Factors affecting the satisfaction factor in individual games

<table>
<thead>
<tr>
<th>Category</th>
<th>Games</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rim World</td>
<td></td>
</tr>
<tr>
<td>Unfinished apartment</td>
<td>when part of the wall or roof is missing</td>
</tr>
<tr>
<td>Entertainment</td>
<td>yes</td>
</tr>
<tr>
<td>Beauty</td>
<td>yes</td>
</tr>
<tr>
<td>Social needs</td>
<td>yes</td>
</tr>
<tr>
<td>Hygiene</td>
<td>no/yes (mod Hygiene)</td>
</tr>
<tr>
<td>Ecology</td>
<td>yes (Green hand attribute)</td>
</tr>
<tr>
<td>Pets</td>
<td>yes</td>
</tr>
<tr>
<td>Death of a familiar character</td>
<td>yes</td>
</tr>
<tr>
<td>Good food</td>
<td>yes</td>
</tr>
<tr>
<td>Drugs</td>
<td>yes</td>
</tr>
</tbody>
</table>

Characters have to strive for pleasure. Some kind of pleasure is to satisfy basic needs such as hunger, thirst and safety. Caring for these needs is crucial to the immersion of the game. As Jan Maria Bocheński wrote: *Make sure you have as much pleasure and as little distress as possible. Such a principle seems obvious, as it is the equivalent of one of the basic laws of life, but man, and perhaps only he, can also violate this law. Some philosophers even raised this violation to the dignity of moral maxim. Well, to be able to use a life, that life must have meaning. To have meaning in life and to be happy are not the same[2].*

Sometimes in video games, the characters that collaborate with are immortal. This is the case with Alyx (from *Half-life II*) in the sense of the game moments (e.g., Travel by car), in a different way.

Just like incorrectly designed game mechanics, the player loses the sense of "soaking up" – the same imitator disappears in the wrong implementation of the models. In the game *Sherlock Holmes Versus Arsène Lupine*, a collaborating character – Dr. Watson does not follow the protagonist, and teleports after him when the distance between them is too big and when the detective will look in a different direction.

### 5.2. PROPOSITION OF IMPLEMENTATION OF FREUD’S MODEL

For better results, the game designer can use existing psychological models. However, their implementation may be a problem. One of the simpler models to implement is the model proposed by Sigmund Freud. The Freud model has a speculative–deductive character. It is based on three dogmas:
1. Determinism – all behaviors and experiences are caused by forces over which man has no control. These forces, however, are the effects of impulses.
2. Conflict
3. Unconsciousness

Man as a subject subject to impulses, strives for pleasure and avoids distress. Sometimes the impulse is blocked – in a given situational or social context, the desire can not be fulfilled[23].

**New idea: Inner game** The internal game is to maximize the benefits of the decision that the character takes. It is a universal concept based on Freud’s theory. Thanks to generalization it is possible to go beyond the teachings of the creator of psychoanalysis. His achievements are often used in scientific works, including narrative role-playing games[3].

Let the game be defined by the game’s theory, in which players receive tokens (each player can receive a different number of tokens) at the beginning of each round. Each player receives a fixed limit on how many tokens can be at its maximum. When the limit is exceeded, the player loses, but can return to the game with fewer tokens.

The round consists of:

1. Disclosure of round card containing information about the change in chips for each player for winning and losing.
2. Fights for the reward – bidding tokens: who will give the most, will win the round.
3. Updating the chip status after knowing the winner on the terms described in the round card.

The following are example of players:

1. The first player represents basic desires such as sleep, food or direct safety.
2. Let there be such a mental power whose task is to be like other representatives of a given type. Another player in the internal game may be the need to instill and to pursue the happiness of the surrounding figures.
3. This game is not very stable: any player can lose without his fault. So another player can be added who has a small influence on rewards and penalties, and can move chips from one player to another, in exchange for receiving a certain number of chips. Its goal is the stability of the game. It loses when any player loses.

The advantage is a more vast concept than in previous chapters. Each decision authority fights for its reward. For example, a hungry man can eat a meal himself, but he can give it to a person who is very undernourished to save his life. Mental
strength responsible for survival (Id by default) can be treated as a player who wants to buy a pot (i.e., meal) for himself, the authority responsible for social life (superego by default) will want to give a meal for the gratitude of the other person. Players can be more than 2.
6. DO CHARACTERS GET BETTER RESULTS BY COOPERATING WITH OTHERS OR WITHOUT COOPERATION?

The test has been done to answer the above question. Cooperation in this case means not only any interaction between units of one faction, but also other behaviors that would not have occurred if the entity would be alone.

The study was conducted in the StarCraft 2 environment with the pysc2 framework, on a specially prepared map. pysc2 is a framework that allows to control the units in the game Starcraft II using the Python language. Python is an interpreted, interactive, object-oriented programming language. It incorporates modules, exceptions, dynamic typing, very high level dynamic data types, and classes. Also it allows overwriting operators[7].

The genetic algorithm was used in this study. Genetic algorithms are search procedures, based on mechanisms of natural selection and heredity. To find a solution for some task using the genetic algorithm it is necessary to have an input state space and a function for calculating the result. The set of states on the basis of which the result is calculated is called the genome. In turn, the function for calculating the result is called a fitness function[11].

```python
def __add__(self, other):
    if isinstance(other, Genome):
        size = len(self.genom_array)
        half_size = int(size / 2)
        ans = []
        positions = [(i<half_size) for i in range(size)]
        shuffle(positions)
        for i in range(0, size):
            if positions[i]:
                ans.append(self.genom_array[i])
            else:
                ans.append(other.genom_array[i])
        return ans
    else:
        raise ValueError("Genome value is required")
```

Listing 6.1: Overwriting the addition operator of the Genome class. Source: own

The program was written in accordance with the eXtreme programming metho-
dologies. This methodology is considered to be perfectly suited to scientific research due to the continuous change of the process depending on the partial results obtained[17]. Three characters were used. They were divided into 2 teams:
— 2 marines
— 1 ultralisk
The values of character class statistics have been modified:

<table>
<thead>
<tr>
<th></th>
<th>Marine</th>
<th>Ultralisk</th>
</tr>
</thead>
<tbody>
<tr>
<td>speed</td>
<td>2.4</td>
<td>2.3</td>
</tr>
<tr>
<td>health points</td>
<td>30</td>
<td>60</td>
</tr>
<tr>
<td>attack range</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>attack damage</td>
<td>6</td>
<td>15</td>
</tr>
<tr>
<td>attack speed</td>
<td>0.86</td>
<td>0.86</td>
</tr>
</tbody>
</table>

Ultralisk is controlled by the default algorithm. The \textit{pysc2} framework was used to control the marines, along with the library that manages the command strings. The result is measured after the last command was issued, even if it does not make it. The main goal of the character is survival, because the opponent has a significant advantage and it is impractical to defeat him alone. Another goal is to eliminate the opponent.
To achieve this, a simple decision system was created, issuing precise commands to the characters being managed. The following table lists the possible commands for the multiagent system, on the basis of which the commands to the agents were sent. The nomenclature used by the creators of the StarCraft II Learning Environment framework, which is the basis for pysc2, may seem misleading due to the fact that the Agent means one player, not one controlled unit\cite{29}.

<table>
<thead>
<tr>
<th>Command</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>look</td>
<td>none</td>
<td>sets the center of view on the opponent</td>
</tr>
<tr>
<td>selectmarine</td>
<td>0 or 1</td>
<td>select first or second (if exists) marine</td>
</tr>
<tr>
<td>run</td>
<td>[x, y]</td>
<td>send an order to selected marine to move</td>
</tr>
<tr>
<td>attack</td>
<td>none</td>
<td>send an order to selected marine to attack</td>
</tr>
</tbody>
</table>

Table 6.2: Implemented commands

The code 6.2 contains the implementation of the generator of one command.

```python
def gen_elem(self):
    ans = choice([[ACTION_SELECT_MARINE, ""], [ACTION_ATTACK, ""],
                  [ACTION_LOOK, ""], [ACTION_RUN, ""]])
    if ans[0] == ACTION_SELECT_MARINE:
        ans[1] = choice([0, 1])
    if ans[0] == ACTION_RUN:
        x = 10*choice(range(0, self.screen_size))
        y = 10*choice(range(0, self.screen_size))
        ans[1] = [x, y]
    return ans
```

Listing 6.2: gen_elem() method of the Genome class responsible for drawing a command for a multiagent system. Source: own

Thanks to the knowledge of goals and priorities, a speculative formula for the game result was created in a speculative way:

\[
s = \begin{cases} 
    o & \text{when } o < c_{\text{max}} \\
    100 \cdot (m) - 100 \cdot h_e + c_{\text{max}} & \text{when ultralisk is alive and } o = c_{\text{max}} \\
    m \cdot 1000 & \text{when ultralisk is dead and } o = c_{\text{max}} 
\end{cases}
\]

where:
- \( h_e \) – normalized value health points of the ultralisk
- \( m \) – the number of marines
- \( o \) – number of last observation
- \( c_{\text{max}} \) – number of commands

\[
(6.1)
\]
The results of the study show that the created multiagent system obtained a result higher than the use of random commands and the lack of control. Observable effects of the obtained result are such that the characters managed to escape from the opponent, but they did not defeat him. It can be concluded that the fitness function was designated correctly, although it was created in a speculative manner.

Simple preliminary tests (225 repetitions for each type of test) were carried out of which conclusions were drawn:

1. In the absence of commands for Marine: each time when the marines were dead ultralisk had 33% of health points.
2. Using random marines commands: each time when the marines were dead ultralisk had more 50% of health points.

*Starcraft II* is mostly deterministic. To prove the hypothesis, it must be shown that there is a sequence of commands, with which a better result will be obtained than using random commands and no commands. It was assumed that 100 commands should be enough to win.

The genetic algorithm was used to create the desired command string. Due to the fact that the commands should be considered as qualitative features, no mutations have been implemented. The command generating program was written in java. Saved to a file command string was read by the program using the framework *pysc2*.

**ANALYSIS OF RESULTS**

The genetic algorithm did not create in the first 500 generations of the sequence that allowed to overcome the ultralisk. However, the Marines are able to flee before his death. The last 255 of scores were compared with the scores from preliminary tests.

In order to compare the results, it was checked whether the groups of results have a normal distribution. The *Shapiro-Wilk test* was performed. For each group the null hypothesis is: *Group has a normal distribution.*

\[ P_{value} \] were obtained:

\[
p_{n1} = 4.802931796873645e - 32 \\
p_{n2} = 4.680334019008432e - 16 \\
p_{n3} = 7.133230367061677e - 14
\]

Assuming \( \alpha = 0.05 \), the zero hypotheses should be rejected. Each group has values with a distribution other than normal. To compare the value of the groups should be formulated to reject the null hypothesis. Two null hypotheses were put forward:
The first null hypothesis: The results of the preliminary test without entered commands are higher or equal to the results obtained by using a genetic algorithm.

The second null hypothesis: The results of the preliminary test with random commands are higher or equal to the results obtained by using the genetic algorithm.

Post-hoc testing is not performed when comparing many groups with non-normal distribution. Thus, the scores were compared using the Mann-Whitney U test. Such zero hypotheses have been chosen which should be rejected. If this happens, alternative hypotheses can be accepted. This is a common practice in statistical hypothesis testing[1]. Bonferroni correction was included:

Factor $\alpha = 0.025$.

$p_{value}$ for first null hypothesis: $p_1 = 5.573279260315225e-40$

The first null hypothesis can be rejected

$p_{value}$ for second null hypothesis: $p_2 = 1.033433747034305e-38$

The second null hypothesis can be rejected

Alternative hypotheses should be accepted:

The first alternative hypothesis: The results of the preliminary test without entered commands are less than the results obtained by using a genetic algorithm.

The second alternative hypothesis: The results of the preliminary test with random commands are less than the results obtained by using the genetic algorithm.

Average results

For each group, the average value of results was calculated:
47.80444444444444 – for control group without control
43.65777777777778 – for a group with random control
64.89777777777778 – for a group with scheduled control

The highest average score was obtained for the group with the planned control.

Quantiles

Another important measure of the distribution are quantile values. There are 3 quantile values. They set boundary values that divide the sorted data set into 4 parallel groups. This means that 25 % of the test data value is less than or equal to the first quartile value, 50 % of the test data value is less than or equal to the II quartile value, and 75 % of the value is less than or equal to the III quartile value. The value of the 2 quantile is the median. The box-plot is created using the quantile values as well as the minimum and maximum values.

Quartiles for the control group without control:

I. quartile: 48.0
II. quartile: 48.0  
III. quartile: 48.0

Quartiles for a group with random control:

I. quartile: 42.0  
II. quartile: 47.0  
III. quartile: 49.0

Quartiles for a group with scheduled control:

I. quartile: 49.0  
II. quartile: 57.0  
III. quartile: 80.0

![Fig. 6.2: Distribution of values in the studied groups.
Source: own](image)

CONCLUSIONS

The created multi-agent system with short learning (500 generations) was not enough for the characters to realize both goals defined earlier. However even a simple multiagent system obtained better results than issuing random commands or lack of control. The characters manage to escape. Naturally, there are still open questions,
such as: If we allow longer researchers, would all the goals be achieved? Should the result be also checked if only one character should be controlled. Created world representation can be used to build a decision system.
SUMMARY

In this master’s thesis, the solutions related to the design of independent characters were analyzed, including the characters cooperating with each other. While modeling characters elements that are described should be taken into account. For example communication with other players. It has been shown that characters achieve better results by cooperating with others. This may be one of the reasons why there is cooperation. A well-designed multi-agent system tries to make cooperation between the NPC and the player.

Due to the unpredictable behavior of the player, it is more necessary to make other decisions than if it were a character controlled by the decision system, which predicts his future movements and "understands" motivations. This master’s thesis shows how human communication methods can be used to manage a multi-agent system. This paper presents a suggestion to build models of human needs as a meaningful game.

There are games such that cooperation is more profitable than playing alone. Thanks to the new approach to modeling NPC needs, games can become more realistic. The difficulty in designing not too difficult and not too simple games can be solved by introducing DNPC at more difficult moments.
A. PERFORMANCE AND OPTIMIZATION

Using complex NPC logic models can slow down the game. The simultaneous number of characters should depend on the target platform of the game. While designing the Halo game, a limit was adopted: no more than 25 active actors, and the average value of CPU usage (pentium III 733 MHz CPU) should not exceed 15% for character logic[5].

A common problem in information processing is the size of variables data in the input vector. For a value of 0-255 (1 byte), the table all expands significantly (exponentially). An even bigger problem is for real values (double type).

The simplest solution is to reduce accuracy. In some cases, even enough value in the range of 0-7. Accuracy does not have to be the same everywhere: the division scale may be non-linear (eg 0, 1-3, 4-6, 7-14, 15-16, 17-20, 21-24, 25-255). A smaller scale simplifies the creation of algorithms (or speeds up learning AI). Sometimes a simplified model is enough to take most decisions. Another approach to optimization is the application of fuzzy logic based on fuzzy sets. As well as reducing accuracy, the use of fuzzy logic facilitates the design and accelerates learning of AI. The advantage is that there is no loss of accuracy, because each time game-developer can change the function of blurring and sharpening, and also choose different peaks of compartments.

Advanced decision-making algorithms (NP-Hard type) can take up a lot of computer resources, and what is directly related to it - including time. If, however, decisions are made for hundreds of characters, the game can not run smoothly. Creating a gigantic world with many NPCs can be problematic not only because of the level of complexity and dependence between characters in the character creation phase, but also during the regular operation of the game engine. In single player games, the main problem is the need to process all NPCs, and in multiplayer games, apart from hundreds / thousands of NPCs, there is a large number of players. To deal with the processing of such large amounts of information, the problem should be decomposed - divided into smaller parts. The characters can be divided into individual types and process each list in parallel. Another division proposal is the territorial division.

Of course, it is more difficult because of the ability to travel some characters. However, a problem arises if decisions are to be made for many characters of the same type on an undivided area. Among the methods of dealing with this problem
one should distinguish the method of grouping decisions for a few units, as well as limiting the frequency of making decisions depending on the distance from the player.

1. It is acceptable that the characters are wrong[9].
2. It’s acceptable that characters have simple models.
3. However, when the characters behave against their logic, it is unacceptable[14].

Characters both cooperating with the player, opposite and neutral should be designed based on behavioral models, because it can help modeling itself, by transferring the burden of creating characters from mechanics to the story. Behavior models should be tailor-made to minimize computing power requirements.
BIBLIOGRAPHY