I would like to thank very much to my supervisor, Mgr. Rudolf Kadlec, who was very supportive and leads me towards successful completing of my bachelor thesis. He was always willing to help and consult my work with me.

My thanks also belongs to the entire AMIS team for creating Pogamut platform without which my project would not existed. Especially I want to thank to Jakub Gemrot for helping with Pogamut issues.

I must not forget to thank to Zafeiros Fountas, David Gamez and Andreas K. Fidjeland for providing NeuroBot and for helping with its installation and running. Then to Jacob Schrum, Igor V. Karpov and Risto Miikkulainen for provind UT$^2$ and support with its operation. And also to Michal Bída for providing KnightHunter.

Special thanks belongs to David Holañ who provided initial bot (Gladiator-Bot) which was used with his kind permission as a base bot for our improvements and give a birth to our Human-like bot.

Last but not least I want to thank to my family for always supporting me and standing behind me. I would not be able to manage it without them.
I declare that I carried out this bachelor thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In Prague on May 23, 2012 Jan Dufek
Název práce: Bot podobný člověku

Autor: Jan Dufek

Katedra: Kabinet software a výuky informatiky

Vedoucí bakalářské práce: Mgr. Rudolf Kadlec, Kabinet software a výuky informatiky


Klíčová slova: bot, autonomní agent, UT2004, Pogamut

Title: Human-like bot

Author: Jan Dufek

Department: Department of Software and Computer Science Education

Supervisor: Mgr. Rudolf Kadlec, Department of Software and Computer Science Education

Abstract: The main goal of my thesis is to create a bot for Unreal Tournament 2004 whose behaviour will be as similar as possible to human controlled players. At the beginning, I measured game play data from both humans and bots. Then, I chose one initial bot and I did five iterations of comparing bot data with humans data, selecting one difference, improving the bot with the aim to eliminate the difference and measuring data again. The successful improvements I made were removing rotations during running, adding reactions on hit and better weapon switching. The improvements that did not make a difference were adding a camping and roaming objective and evade using visibility module. In the end, believability of the bot was tested during a study with human players. The bot will take part in a human-like bot competition named the 2K BotPrize.

Keywords: bot, autonomous agent, UT2004, Pogamut
# Contents

## 1 Introduction

1.1 Unreal Tournament 2004 ........................................... 5

1.1.1 Death match .................................................. 5

1.2 Navigation in a virtual environment .......................... 6

1.3 Pogamut ............................................................ 6

1.4 The 2K BotPrize .................................................. 7

1.5 Why do we need human-like bots? ............................. 7

1.6 Our method ......................................................... 7

1.7 Structure of thesis ............................................... 8

## 2 Related work

2.1 Human-like bots ................................................ 10

2.1.1 NeuroBot ..................................................... 10

2.1.2 UT² .............................................................. 10

2.1.3 Knightbot ...................................................... 10

2.1.4 ICE-CIG2011 ................................................. 10

2.1.5 Long term memory bot ..................................... 11

2.1.6 ICE-UT@RITS ............................................... 11

2.2 Measuring humanness .......................................... 12

2.2.1 Game bot detection ........................................ 12

## 3 Measuring data

3.1 Software for measuring data .................................. 13

3.1.1 GameBots2004 ............................................... 13

3.1.2 Software design ............................................. 13

3.1.3 GameBots2004 messages .................................. 14

3.2 Humans data ...................................................... 15

3.2.1 Participants .................................................. 15

3.2.2 Game server ................................................ 15

3.2.3 Game settings ............................................... 16

3.2.4 Players preparation ........................................ 16

3.2.5 Recording demo ............................................. 16

3.2.6 Data format .................................................. 16

3.2.7 Matches ....................................................... 17

3.2.8 Website ....................................................... 18

3.3 Bots data .......................................................... 18

3.3.1 The BotPrize bots ......................................... 18

3.3.2 The Epic bots ............................................... 18

3.3.3 The Pogamut bots ......................................... 19

3.3.4 Tournament ................................................. 19

## 4 Analysing data

4.1 Units in Unreal Tournament 2004 ............................. 20

4.1.1 Time ............................................................ 20

4.1.2 Distance ....................................................... 20
4.1.3 Angle  ........................................ 20
4.1.4 Coordinates  .................................. 20
4.1.5 Rotation  .................................. 21
4.2 Modes  ........................................ 21
4.3 Measures  ........................................ 21
  4.3.1 Adrenaline  .................................. 21
  4.3.2 Armour  .................................. 21
  4.3.3 Distance from conflict point  ................. 22
  4.3.4 Distance enemy  ................................. 22
  4.3.5 Distance from previous location  .............. 22
  4.3.6 Health  .................................. 22
  4.3.7 Number of rotations per time unit  .......... 23
  4.3.8 Primary ammo  .................................. 23
  4.3.9 Rotation difference  ........................... 23
  4.3.10 Secondary ammo  .............................. 23
  4.3.11 Small armour  .................................. 23
  4.3.12 Standing and rotating duration  .............. 24
  4.3.13 Standing duration  ............................. 24
  4.3.14 Time for switching picked up weapon  ......... 24
  4.3.15 Time from last turn  ........................... 24
  4.3.16 Ultra damage time  .............................. 24
  4.3.17 Is alternative firing  ......................... 24
  4.3.18 Is crouched  .................................. 25
  4.3.19 Is shooting  .................................. 25
  4.3.20 Is standing  .................................. 25
  4.3.21 Is standing for a long time  ................. 25
  4.3.22 Is walking  .................................. 25
  4.3.23 Rotates to enemy after hit  .................. 25
  4.3.24 Rotating when running  ....................... 25
  4.3.25 Weapon  .................................. 26
  4.4 Software for analysing data  ..................... 26

5 Statistics  ........................................ 28
  5.1 Statistics  ..................................... 28
    5.1.1 Histograms  .................................. 28
    5.1.2 Quantities  .................................. 28
    5.1.3 Akaike information criterion  ............. 28
    5.1.4 Parameters estimation  ...................... 29
  5.2 Comparison  .................................... 29
  5.3 T-test  ......................................... 30
  5.4 Another scripts  ................................ 30

6 Methods of implementation  .......................... 31
  6.1 Work flow  ...................................... 31
    6.1.1 Idea  ...................................... 31
    6.1.2 Comparison  .................................. 31
    6.1.3 Theory  .................................... 31
    6.1.4 Measuring data  ............................... 31
    6.1.5 Theory confirmation  ....................... 32
6.1.6 Repeating .................................................. 32
6.2 Possibilities of comparison ................................. 32

7 Initial bot ......................................................... 33
7.1 About GladiatorBot ........................................... 33
7.2 Reason of choosing GladiatorBot ......................... 33
7.3 Abilities ......................................................... 33
  7.3.1 Collect items objective .................................. 33
  7.3.2 Attack objective .......................................... 34
  7.3.3 Scan objective ............................................ 34
  7.3.4 Choosing objectives ...................................... 34
7.4 Disadvantages ............................................... 34

8 Examples of statistical comparison ......................... 35
8.1 Distance from enemy ........................................ 35
8.2 Distance from conflict point ............................... 35

9 Human-like bot 1 ................................................ 39
9.1 Results ........................................................ 39

10 Human-like bot 2 ............................................... 41
10.1 Idea ........................................................ 41
10.2 Comparison ................................................ 41
10.3 Theory ....................................................... 41
  10.3.1 Roam objective .......................................... 41
  10.3.2 Camp objective .......................................... 42
10.4 Theory confirmation ........................................ 42

11 Human-like bot 3 ............................................... 44
11.1 Idea ........................................................ 44
11.2 Comparison ................................................ 44
11.3 Implementation ............................................. 44
  11.3.1 Movement strategy selection .......................... 44
  11.3.2 Evade movement strategy .............................. 45
  11.3.3 Evade objective .......................................... 45

12 Human-like bot 4 ............................................... 46
12.1 Idea ........................................................ 46
12.2 Comparison ................................................ 46
12.3 Theory ....................................................... 46
12.4 Theory confirmation ........................................ 46

13 Human-like bot 5 ............................................... 49
13.1 Idea ........................................................ 49
13.2 Comparison ................................................ 49
13.3 Theory ....................................................... 49
13.4 Theory confirmation ........................................ 49
14 Human-like bot
14.1 Idea .......................................................... 52
14.2 Comparison .................................................. 52
14.3 Theory ......................................................... 52
14.4 Theory confirmation .......................................... 52

15 Study with human players ................................. 55
15.1 Method ......................................................... 55
15.2 Results ........................................................ 55
15.3 Comments .................................................... 55
15.4 Discussion ..................................................... 56

16 Conclusion ....................................................... 57
16.1 Results ........................................................ 57
16.2 Discussion ..................................................... 57
16.3 Future work .................................................... 57

Bibliography .......................................................... 60

List of Figures ........................................................ 62

List of Tables ........................................................ 63

Attachments .......................................................... 64
1. Introduction

The goal of this work is to create artificial intelligence based on Pogamut \cite{15, 14} for virtual agent (bot) playing first person shooter game Unreal Tournament 2004. The behaviour of this bot should be as close as possible to the behaviour of human controlled players. The bot will take part in the IEEE WCCI 2012 Human-like Bots competition \cite{8} and in The 2K BotPrize competition \cite{9}. I will now explain the basic terms which are required for understanding issues of the thesis.

1.1 Unreal Tournament 2004

Unreal Tournament 2004 is first person shooter game from Epic Games \cite{4} company. Player is controlling virtual body which is moving in the virtual environment called map. There are several types of games but the only relevant for us is a death match. The main goal of a death match is to kill as many other players as possible. The player can collect various types of items which make it easier to reach this main goal. For example weapons, ammunition, health, armor, damage amplifier etc. If the player is killed he is instantly reborn but he looses all collected items. To give a visual illustration of this game a screen shot from a game play is on the figure 1.1.

1.1.1 Death match

Death match is one of many game types where a player can choose in Unreal Tournament 2004 and generally in many other first person shooter games. Bots
in the both human-like bots competitions mentioned above will be playing the
death match, so I will focus exclusively on this type of a game.

There is set a certain limit of killed opponents which is called a frag limit. It
can be adjusted in settings but default value is 25. If the player kills opponent,
the number of his frags increases and the player who first reaches the frag limit
will win the match. If the player kills himself for example by falling into lava, the
number of frags decreases.

There is also a time limit for one match and if the match reaches this limit,
the player with the highest number of frags will win the match. The time limit
can also be adjusted in the settings.

All other players in the death match are the player’s enemies hence there are
no teams in the death match. If the player is killed, he re-spawns instantly on
some of specified places in a map. The health level after spawn is 100 points and
the player has only two basic weapons. It is a ShieldGun and a AssaultRiffle.

The player can collect items in order to make it easier to reach a frag limit.
Items are for example weapons, ammo and power ups as health packs, armour,
damage amplifier and so on.

1.2 Navigation in a virtual environment

The computer controlled agents do not have vision like humans do. They have
two possibilities for navigation in the environment.

The first less reliable is ray casting. The bot can send a ray in a given direction
to find out what the distance from an obstruction is. If the bot sends more rays
into different direction, it can be used for navigation in the environment.

The second possibility is using a navigation graph built in a virtual environ-
ment. It is composed from navigation points and navigation edges. Navigation
points are interesting points in the environment for example starting point of
a player, a pick up spot, a jumping pad, a sniping spot, an ambush spot or just
a point where a bot can safely stop. Navigation edges connect two navigation
points where a direct safe path exists. Navigation edges vary in types, which
means that there are for instance edges where it is sufficient to just run or where
the bot must perform some kind of a jump. Navigation graph makes it very
easy for a bot to move in a map. Also the standard graph based algorithms as
A* [1] or Floyd-Warshall [5] can be used. If the bot moves only on the navigation
graph, there is high probability of successful movement in the map. Unfortu-
nately, sometimes there are mistakes in navigation graphs which are made by the
creators of these maps. This is a source of problems with bots navigation.

1.3 Pogamut

Pogamut is a Java library that allows us to control virtual agent in the environ-
ment provided by a game engine. Pogamut simplifies the basic physical actions
of an agent, for example path finding and path execution, so I can focus on high
level agent’s intelligence.

The game play data are sent using TCP/IP by GameBots2004 text protocol.
Text messages from GameBots2004 are passed to Gavialib Java library and con-
verted into Java objects. Then Pogamut can work with these objects and provide a wide range of high level functions. The architecture of Pogamut can be seen in figure 1.2.

Figure 1.2: Pogamut architecture. Reprinted from [15] with permission from the authors.

1.4 The 2K BotPrize

The 2K BotPrize is a competition challenging developers to create bots for Unreal Tournament 2004 that behave like a human controlled players. The competition is modified Touring test for virtual agents. Bots and human controlled players are connected into virtual environment and human judges are supposed to distinguish bots from humans. They rank every player in a mean of their humanness and also make comments on them. The higher humanness means that the bot is more similar to a human and was mistaken for a human more times. The bot which reaches certain level of humanness will win the major prize. The 2K BotPrize takes place every year since 2008, and in the mean time, no one has won the major prize. Therefore the bot with the highest humanness wins the minor prize.

In [16] is described a modified Touring test used in the 2K BotPrize. The judges are using more natural judging system. They are part of the game and they are judging during the game play using a LinkGun.

1.5 Why do we need human-like bots?

The main question is: Why do we need human-like bots? It is because humans want to play computer games with other players behaving also like humans. It is true that computers can play computer games on a very high level. They have capabilities of fast computing so they can reach the main goal better. But the reason we made computers play games was to replace other humans. So we do not need other humans to play against us. We can play alone and computer will take care of the rest. Optimal solution is that we do not even recognize that we are not playing with humans.

1.6 Our method

Our approach to the problem of creating a human-like bot is following:

1. Collect game play data from humans.
2. Compute certain measures for humans.
3. Statistically process these measures.
4. Collect game play data from bots.
5. Computed certain measures for bots.
6. Statistically process these measures.
7. Compare results of humans and bots and determined the differences.
8. Make hypothesis why the particular measure different is.
9. Try to implement improvement to initial bot.
10. Collect game play data from improved bot.
11. Compute certain measures for improved bot.
12. Statistically process these measures.
13. Compare results of humans, initial bot and improved bot.
14. Prove or disprove hypothesis.
15. Continue with 8.

The result should be the bot which is in the improved measures closer to humans than the initial bot.

1.7 Structure of thesis

The second chapter discusses related work. What the advantages and the disadvantages are. And then, the differences between our work are described.

The third chapter is about collecting the game play data from both humans and bots. There are defined data which I am collecting. Next, there is a description of the software for collecting data. And also description of tournament for humans I have organized.

The fourth chapter describes the analysing part of our work. We can find there a definition of measures we computed. In addition, I describe the software for analysing data.

The fifth chapter is dealing with the statistical processing of computed measures. I examine there the software for computing statistical quantities, printing graphs and estimating distributions.

The sixth chapter is explaining in details my method used for creating a human-like bot.

The seventh chapter is describing initial bot I chose for the improvement. I am showing its capabilities and its weaknesses.

The eighth chapter is showing the possible statistical tools I have for comparing the different bots and humans. There are two examples of measures and I am examining them. I am also showing interesting facts following from these two measures.
The chapters nine to chapter fourteen describe improvements of the bot I made.

The fifteenth chapter informs us about testing the bot’s believability on humans.

The sixteenth chapter is at first summarizing results, and next there is a discussion and description of future work.

Attachment A is a first DVD medium containing all the software implemented for this thesis. It is Tournament for measuring data, Analyser for computing measures, Statistics for statistical processing and all six versions of Human-like bot. It also contains output from this software especially output form Statistics which contains histograms, distribution estimations, Akaike information criterion and statistical quantities everything for each tournament, for each player, for each map and for each player on each map. There are also comparison graphs comparing humans to bots and then t-test comparing humans and bots. At last, it contains electronic version of this thesis. There is also a user documentation, a programmer documentation and a javadoc.

Attachment B is a second DVD medium containing data measured by Tournament application from all tournaments. There are also all reference bots used in bots tournaments. And we can find there a Pogamut library used by this project.
2. Related work

In the following chapter, I am going to analyse related works, compare them with my work and show advantages and disadvantages of their approach. I divided this chapter into two sections. The first one is other human-like bots and the second is measuring humanness of players.

2.1 Human-like bots

2.1.1 NeuroBot

NeuroBot [12] was a finalist of The 2K BotPrize competition. They are using biologically inspired approach. The architecture is based on theories about high level control circuits in the brain. They implemented spiking neural networks to control an agent through the Pogamut platform in Unreal Tournament 2004. NeuroBot achieves about 36% of humanness in the 2011 2K BotPrize. It was the second best result.

Their bot is not using built in navigation graph so the movement is definitely more human-like than the one of our bot. But it brings also certain problems with navigation in the environment. Current neuronal bot can not recognize cliffs and can easily fall down from them.

2.1.2 UT2

UT2 [19] reached humanness about 21% in the 2011 2K BotPrize competition. The UT2 is based on learning behaviour by multiobjective neuroevolution. It can observe opponents and learn from them. It also uses database of human traces from game play which are used when the bot gets stuck in the environment. The bot is controlled by neural network.

I am also using human game play data, but in a slightly different way. They are directly using and replaying human traces for the bot’s movement. I am not using human data by the bot. I am measuring these data and trying to make bot’s data fit them.

2.1.3 Knightbot

[21] is an article about the bot using inverse reinforcement learning [20]. The bot learns how to play from an expert human player. By reinforcement learning the bot tries to learn rewards for different game states.

2.1.4 ICE-CIG2011

ICE-CIG2011 [22] is a bot which is capable of learning from human judges and after that judge another humans or bots by itself. They believe that behaviour connected with judging other players will increase humanness of the bot in the 2K BotPrize competition. It is because in the competition humans compete also to be the best judges. This could lead in a little bit different behaviour.
The bot is learning tactics from judge logs. They are using a neural gas which is a type of neural network. The bot is trained by neuro evolution from the log. The neural gas is used for deciding whether an opponent is a bot or a human.

2.1.5 Long term memory bot

In they are describing the system used by the winner of the 2009 2K BotPrize. Most of the game bots are using short term reactive memory. It seem that long term memory known from humans is usually missing.

The bot uses the SQLite database for storing long term data from the environment. There are various types of information stored in this database. The bot stores hotspots where it killed someone and hotspots where it had been killed. Then, it tries to avoid those hotspots where it was killed and to reach the hotspots where it killed someone else.

Other useful feature is a use of visibility information gained before the match. The bot can later mimic human evade behaviour and hide behind the obstructions. Visibility information is stored before the match in the database. The bot also remembers the objectives which were unsuccessful and tries to avoid repeating them for a certain time in the future.

In my work I am also using a visibility library provided by Pogamut to mimic evade behaviour. Our evade conditions are a little bit different. I am not using just health information but also armor, damage amplifier and weapon information.

2.1.6 ICE-UT@RITS

The ICE-UT@RITS was on the second place in the 2008 2K BotPrize. In they are analysing videos from the 2008 2K BotPrize. They have determined five differences between their ICE-UT@RITS bot and human. After that, they tried to fix these differences and fixed their ICE-UT@RITS.

The differences were the following:

1. The first is item acquisition. The bot should not try to capture the item which was already taken by an opponent.

2. The second is attacking method. The bot should choose the weapon according to the situation.

3. The third is changing weapons. The bot should not change the weapon too often.

4. The forth is combat against multiple opponents. The bot should change target opponent when he loses the actual one from the sight and another is nearby.

5. The fifth is combat movement. The bot should not always approach enemy. So the movement around opponent was implemented and also the condition that health need to be 100 to approach was added.

Our bot meets all the factors except the last one. Our bot does not move around the enemy. It has only approach, flee and evade behaviours.
2.2 Measuring humanness

2.2.1 Game bot detection

In [18] they focused on automatic game bot detection. The on-line games are more and more popular. But there is a problem with using game bots for earning game awards without any efforts. The gaming community generally disapproves using of game bots.

This system uses traces of the player to recognize whether it is a bot or a human. It seems that human traces are very different from the bot’s one. Only a few hundreds seconds is sufficient to recognize bot from human. It can be used in every game where the player directly controls movement of avatar in the game.
3. Measuring data

My approach is based on game play data, so I need to collect this data from humans, from bots and after from our own bot. Initially, I will show our software for measuring game play data. After that I will describe tournament of humans and at the end tournament of bots.

3.1 Software for measuring data

I have used Pogamut platform to connect to the game server in order to measure game play data. I will now describe terms important to understand how this application works.

3.1.1 GameBots2004

GameBots2004 is a mod\[1\] to the Unreal Tournament 2004. It is written in UnrealScript\[2\]. It provides a text protocol to communicate with bots in the game. We can send text commands to the bots and at the same time we receive information from the environment. Hence, it can be used to control bots in the virtual environment.

There are three main types of connections to a server provided by GameBots2004:

- **Bot connection** used to control bot in the environment.
- **Control connection** used to control the game mechanism (for instance pause a game).
- **Observer connection** used to observe players.

I have already mentioned that GameBots2004 is used by Pogamut to communicate with the game engine of Unreal Tournament 2004.

3.1.2 Software design

I wrote an application in Java called Tournament which uses the observer connection to GameBots2004 server. Observer connection is intended to only gather information from the server because I do not want to control any agent.

At the beginning, we download a list of all players in the game. After that, we make an observer connection for every player. Observer connection allows us to register listeners on almost every message that is send by GameBots2004 server. There are many types of messages. The most important one is a self message

---

\[1\] Mods are game modification of the Unreal Tournament. They can modify internal classes written in UnrealScript. With mods it is possible to modify almost everything except of the engine. It is possible to create the new game with mods.

\[2\] UnrealScript is a scripting language especially made be Epic Games for Unreal Tournament series. It is similar to Java or C++. All Unreal Tournament game series is written in UnrealScript except the of the engine. The engine is written in C++.
which contains information about player. It is for example the health level or location.

If the particular message come, additional information is attached. It is the name of the player, then simulation time when the message came, type of the message and the message. Message with additional information is then saved in the text file to the following format:

<player_name>
$<time_from_beginning>
$<type_of_message>
$message

Every message is written on a separate line and all messages from one match are written to the one file. This file is then saved into output directory in the project root. For every new match the random name of the file is chosen because we do not want to overwrite a previous one. We must run the application after all players are connected to the environment because a player list is exported just once at the beginning.

3.1.3 GameBots2004 messages

I am saving the following types of GameBots2004 messages:

- **AddInventoryMsg** sent when the player get a new weapon or an ammunition.
- **AdrenalineGained** sent every time the player get a new adrenaline.
- **ChangedWeapon** sent if the player changed weapon.
- **ComboStarted** sent when the player used a combo.
- **FallEdge** sent every time the player is on the edge of something (for example cliff). If the player was running it is already falling. It the player was walking it can not fall of the edge.
- **GameInfo** send only after READY command. Contains information about game such as time, teams etc.
- **GlobalChat** send every time the global chat message has been sent.
- **HearNoise** send if the player hears a noise.
- **HearPickup** send if the player hears a pick up of an item.
- **IncomingProjectile** send if there is an incoming projectile that the player can see.
- **ItemPickedUp** send if the player picked up an item.
- **JumpPerformed** send if the player jumped.
- **Landed** send after the player landed on the ground after falling.
• *LostChild* send if the observed player left the game.

• *PlayerDamaged* send if the observed player hit a another player.

• *PlayerJoinsGame* send if the new player joined the game.

• *PlayerKilled* is send when another player then the observed one is killed.

• *PlayerLeft* is send when the player leaves the server.

• *PlayerScore* is a synchronous message and contains information about player score.

• *RecordingEnded* send if the demo recording stopped.

• *RecordingStarted* send if the demo recording started.

• *Self* is a synchronous message and contains information about a state of the player.

• *ShootingStarted* send when the player started shooting.

• *ShootingStopped* send when the player stopped shooting.

• *Spawn* send when the player spawn in the environment.

• *Thrown* send every time the player throw out a weapon.

• *WallCollision* send if the player collided with a wall.

• *WeaponUpdate* send when the player changed a weapon.

All messages except of Self and PlayerScore are asynchronous. It means that they come when the specific action takes place. Self and PlayerScore messages comes after certain constant period of time which is defined in GameBots2004 setting.

### 3.2 Humans data

The first task of this project was to find out how humans play Unreal Tournament 2004. I organized a tournament in Unreal Tournament 2004 which was held at the Faculty of Mathematics and Physics at Charles University in Prague in class SW1. I am now going to describe this tournament.

#### 3.2.1 Participants

All participants were asked to fill an on-line questionnaire before the match had started. In this form, there was all important information about them. Data from the form can be found in the Tournament directory in attachment B.

From the questionnaire, we can see a lot of information such as what the level of a player is or if a player had headphones. At the beginning of the tournament there were twelve participants. The average age was 21 with standard deviation 2. There was a range from experienced players to absolute beginners.
3.2.2 Game server

I was using dedicated GameBots2004 server, type of the game was Death Match and time limit to 999,999. I had to set an administrator name and password in order to be able to record server side demo. Server was run by bat script and I used separated bat file for each map. All scripts for running servers are in Tournament directory on attachment B.

3.2.3 Game settings

All configuration ini files with settings of Unreal Tournament 2004 used during measuring the data are in the Tournament directory on attachment B. There was not any special modification of settings.

3.2.4 Players preparation

All players were asked to set up their names same as their surname in a lower case without diacritic. The match began only on a signal from the organizer. After connecting to the server, all players were not allowed to move and had to turn off all additional features of GameBots2004 server which are usually used for developing bots. The way to do it is to use keyboard shortcut Alt + h. On the screen, there will appear green and red lines representing particular features. Green features are active and red are inactive. Players had to turn off all active features by pressing short cuts presented at the begging of each green line. After that, the organizer started recording server side demo and launch software for saving data. In that moment, the organizer started the match.

3.2.5 Recording demo

Demo is a record of a match in audiovisual form. There are two types:

- Client-side demo records only from view of one particular player who started recording.
- Server-side demo record whole world including view of all players.

For recording server-side demo the Unreal Tournament 2004 server has to be run with the set administrator name and password. During the match I needed to enter console and then to log in as administrator. By simple command I started recording server side demo and after the match the demo was saved in the folder Demos in the installation directory of the Unreal Tournament 2004.

3.2.6 Data format

One file corresponds to one match and file name is composed from a number of a match and used map. The file is a plain text file which contains on each row one incoming message from GameBots2004 server with meta tags in following format:
Table 3.1: List of matches

<table>
<thead>
<tr>
<th>Match</th>
<th>Map</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DM-Flux2</td>
</tr>
<tr>
<td>2</td>
<td>DM-1on1-Albatross</td>
</tr>
<tr>
<td>3</td>
<td>CTF-Face3</td>
</tr>
<tr>
<td>4</td>
<td>CTF-FaceClassic</td>
</tr>
<tr>
<td>5</td>
<td>DM-DesertIsle</td>
</tr>
<tr>
<td>6</td>
<td>DM-Antalus</td>
</tr>
<tr>
<td>7</td>
<td>DM-Injector</td>
</tr>
<tr>
<td>8</td>
<td>DM-Flux2</td>
</tr>
<tr>
<td>9</td>
<td>DM-DE-Grendelkeep</td>
</tr>
<tr>
<td>10</td>
<td>DM-Deck17</td>
</tr>
<tr>
<td>11</td>
<td>DM-Leviathan</td>
</tr>
<tr>
<td>12</td>
<td>DM-Leviathan</td>
</tr>
<tr>
<td>13</td>
<td>DM-1on1-Albatross</td>
</tr>
<tr>
<td>14</td>
<td>CTF-FaceClassic</td>
</tr>
<tr>
<td>15</td>
<td>DM-Plunge</td>
</tr>
<tr>
<td>16</td>
<td>DM-Rrajigar</td>
</tr>
<tr>
<td>17</td>
<td>DM-TrainingDay</td>
</tr>
<tr>
<td>18</td>
<td>DM-TrainingDay</td>
</tr>
<tr>
<td>19</td>
<td>DM-1on1-Serpentine</td>
</tr>
<tr>
<td>20</td>
<td>DM-1on1-Roughinery</td>
</tr>
</tbody>
</table>

The beginning of the file could contain messages saved before organizer started the match. All measured data and demos are in the Tournament directory on attachment B. It contains folders with following names:

Match <number_of_match> <used_map>

Each folder contains two files with the same name as folder. One is in dat format and second in demo4 format. File in format dat contains measured data and file in format demo4 is server side demo.

3.2.7 Matches

The tournament in Unreal Tournament 2004 lasted about 5 hours. The initial number of people was twelve but people were leaving during tournament. So the number of players is not constant in all matches. We played twenty matches on fourteen different maps. Type of a game was DeadMatch and frag limit was 25. The list of all matches can be seen in table 3.1.
3.2.8 Website

I created a project page [11] on the Pogamut development wiki. There is the description of the tournament, all important information and also everybody can download data from the tournament and used them for own project.

3.3 Bots data

In order to determine differences between humans and bots, I needed also to measure data for bots. The conditions of bots tournament had to be as same as possible to the tournament of humans. I have chosen several reference bots from different categories. Those categories are the 2K BotPrize finalists, the epic bots from Epic Games and the Pogamut bots. We will now take a look at them.

3.3.1 The BotPrize bots

At first, I have contacted three the finalist of the 2011 2K BotPrize. The creators of ICE-CIG2011 bot, creators of NeuroBot and creators of UT\(^2\). I asked them to provide their bots to us. Creators of the NeuroBot and the UT\(^2\) sent us the bots. The creators of the ICE-CIG2011 offered, that they will connect their bot to our server. They did not want to send us the bot binaries or source codes because they are going to participate with the same improved bot in the 2012 2K BotPrize. Connecting the bot to our server would be difficult to organize because I needed to measure the bots many times.

So I had two bots from the 2011 2K BotPrize. The NeuroBot and UT\(^2\). I have to say that authors were very helpful with installing and running the bots and also solving some problems.

The UT\(^2\) was easy to run. I just needed to edit PogamutPlatformCustom.properties file in the bot root and set IP address to the server. Then I run the START.sh file and the bot then connected to the server. In the DATA folder the human data are contained. Bot uses human traces for movement.

The NeuroBot is a little bit more difficult to run despite the fact that it comes with an automatic Windows installer. If there is graphic card with support for CUDA drivers, the NeuroBot can utilize multiple cores of graphical processing unit concurrently. If not, it will use the CPU only. I am unfortunately using UNIX base system so I needed to run the NeuroBot from the virtual machine without support for CUDA drivers.

The NeuroBot is managed from graphical application. We can fill in the name of the bot, IP address of the server and press the button Login as a bot. After the two windows will pop up. We need to check Operate check box and press the Nemo button. This will run the simulation. The new window will pop up. Then we need to press space button on the keyboard. The bot will now operate in the environment.

3.3.2 The Epic bots

As the next reference bots I chose built-in bots (also called native bots) from Epic Games. They have ten levels of bots. The level one is the easier to kill,
and on the other side the level ten bot is the most difficult to kill. I chose level 1, level 5 and level 10 to cover all spectrum. Unfortunately, it is not possible to observe native bots using the Pogamut platform. So, the bots were used just as opponents of various skills for other bots.

3.3.3 The Pogamut bots

From the Pogamut platform I chose three reference bots. The HunterBot from the Pogamut tutorial, KnightHunter and GladiatorBot. From the HunterBot and the GladiatorBot I had both the jar file and source codes. From KnightHunter I got only jar file. Initially I needed to modify the HunterBot to do not change the name every time it changes objective since I needed the name to be constant all tournament.

3.3.4 Tournament

I had in total eight bots playing the tournament:

- UT
- NeuroBot
- NativeBot 1
- NativeBot 5
- NativeBot 10
- HunterBot
- KnightHunter
- GladiatorBot

It was not unfortunately possible to automatize tournament because the NeuroBot can be run only from graphical user interface. I contacted the author of the NeuroBot and he responded that this feature will be available in the future. So I had to do everything manually.

First, I started the server. After, I connected to it and started to record server side demo. Then, I connected all the bots to the server. I made sure that all bots are connected and ran the Tournament application. The data from the match were saved to the file. I repeated it for all twenty matches as in case of humans. The initial settings was the same it means frag limit also 25 and no special settings.
4. Analysing data

This chapter is about analysing data collected from tournaments mentioned in the previous chapter. At first, I will describe units used in the Unreal Tournament 2004. It is important for understanding the measures I am computing. After that, I will explain modes of a player I am distinguishing and measures I am computing. In the end, I will show my application for measuring data.

4.1 Units in Unreal Tournament 2004

In the virtual environment of Unreal Tournament 2004 are different units than in the real world. Hence I consider it important to describe them because they are used in measures I am computing.

4.1.1 Time

There are two different time units which can be used. We will now take a look at them.

At first, there is Unreal Tournament 2004 time. It is used inside a game and is quicker than real time. Exactly it is 110% of the our time. We can see an example on Damage Amplifier. It doubles a damage of weapon for 30 seconds of game time but in real time it is 27 seconds.

The second is the time used in Pogamut, Analyser and all our project. It is synchronized with Self message which is coming every certain amount of time. This depends on settings but in default it is 250 ms. So, one time unit is time between two self messages in default 250 ms.

4.1.2 Distance

There is not any exact formula for conversion between the game distance and real distance. But it can be imagined using following likening:

- 50 is a really small step.
- 100 is one normal step.
- 200 is height of a bot.
- 1,000 are ten steps.

4.1.3 Angle

One degree is 182.1 units so full angle is 65,556 units.

4.1.4 Coordinates

Coordinates are a little bit different. The x axis in the game is what we are usually calling y axis in the real world and in opposite y axis in the game is what we are usually calling x axis in real world.
4.1.5 Rotation

Rotation yaw is an angle from x axis in game coordinate system in positive direction. Roll is not used and is always zero.

4.2 Modes

I am distinguishing three modes of the player and two gaps between them. The first mode is the calm mode which is defined as a player does not see any other players and did not shoot in certain amount of time.

The second mode is an approach defined as the player can see enemy, shot before certain amount of time and is approaching the enemy. With approaching enemy I mean that the player is moving towards the enemy. I am only considering the movement of the players. So, if the distance between the player and the enemy is constant but the player is moving towards enemy and chasing him it is still approach.

The third is the evade defined as opposite to the approach. It means that the player can see enemy, shot before certain amount of time and is moving away from enemy. If the distance is same but he is running away from the enemy, it is evade.

The evade and the approach are properly defined as follows. If the distance between the player previous location and the enemy previous location is less than the distance between the player actual location and the enemy previous location the player is approaching. If it is more, the player is evading.

I am also distinguishing change from evade to approach and change from approach to evade as two gaps between approach and evade modes.

4.3 Measures

In each mode I am computing the same measures. We will now describe them all.

4.3.1 Adrenaline

Adrenaline is a power up which can be collected or earned by killing opponents. Minimum level of adrenaline is 0 and maximum level is 100. It can be used to perform adrenaline combos. Combos brings some advantage for a player for a limited time. It is for example an invisibility or higher speed.

Adrenaline measure is a level of adrenaline taken from synchronous Self message. So I save the adrenaline level with every incoming message.

4.3.2 Armour

Armour can be collected in the form of a Shield Pack or Super Shield Pack and it absorbs certain level of damage. How much depends on how much armour do the player have. Shield Pack gives the player 50 points of armour and Super Shield Pack gives 100 points of armour. Maximum armour from Shield Packs is 50 points and from Super Shield Packs is 150 points. Overall armour is a sum from
armour from Shield Pack and armour from Super Shield Packs. Minimum is 0 and maximum is 150. If the player has over 100 points composed from both Shield Pack and Super Shield pack it will absorb 100 % of damage. Below 100 points Super Shield Pack absorbs 75 % of damage and Shield Pack 50 % of damage.

Armour measure is a level of armour taken from synchronous Self message.

4.3.3 Distance from conflict point

The distance from conflict point is a measure telling how long from a position where the last fight started did the player move. When the player sees an enemy and start shooting, I saved actual player location as a start point of the fight. Then I measure how long the player moved from this initial location.

It is intended to measure dynamics of fight movement. If the player stops moving and start shooting on an enemy, the distance from conflict point will be zero. But if the player is for example chasing his enemy, the distance from conflict point will be higher.

4.3.4 Distance enemy

The distance from enemy is a space between a player location and his enemy location. It is saved with synchronous Self message every certain amount of time. The closer a player is the lower the distance will be. If the player is in the calm mode (it means he does not see enemy), it is distance from his last enemy.

4.3.5 Distance from previous location

The distance from previous location measures dynamic of movement. It is the distance between actual player’s location and a previous player location. The previous location is a location where the player was when the last Self message came (so by default before 250 ms). If the player is standing on the same place, this measure will be zero and if it is moving instantly away it will be higher.

4.3.6 Health

After spawn the player has 100 points of health. Health is indicating the amount of damage the player can take before he dies. With every damage made to the player the level of the health drops.

The health can be earned by collecting health power up. The first is Health Vital which adds 5 points to the player health up to 200 points. The second is Health pack which will add 25 points up to 100 points. The last is Big Keg O’ Health which adds 100 points up to 200 points.

The minimum health is 0 and it means that the player died and the maximum is 200. The damage amount depends on many factors. For example used weapon, distance and position of hit. For example head shot takes more health.

Health measure is a level of health. It is saved with synchronous Self message.
4.3.7 Number of rotations per time unit

Number of rotations per time unit is trying to capture the frequency of rotations. The player make one rotation if he consequently change his rotation and the angle exceeds the given threshold which can be adjusted. It can not last more than a certain time limit and it have to be in one direction. And the player can not look in the direction of movement during rotation.

Time unit length can be adjusted in the application settings. The measure is showing how many of these rotations the player did in the given period of time specified in setting.

4.3.8 Primary ammo

Weapons in the Unreal Tournament 2004 have two firing modes:

- Primary firing mode.
- Secondary firing mode.

Firing from a weapon consumes certain amount of ammunition. How many ammunition is needed for one fire and what are the maximum levels of ammo depends on the weapon. Some weapons do not need ammo at all. For example a ShieldGun one. Also in case of some weapons the ammo is shared between primary and secondary firing mode. Other weapons has different ammo for primary firing mode and different ammo for secondary firing mode.

The primary ammo measure is a level of ammo for primary firing mode for the weapon the player is currently holding.

4.3.9 Rotation difference

Rotation difference covers dynamic of rotations. It is defined as an angle from previous rotation so it is analogical to the measure distance from previous location. The previous rotation is the rotation of the player when the last synchronous Self message came. Then, I compute and angle between the previous rotation and the actual rotation. If the rotation difference is zero, it means that the player did not rotated at all.

4.3.10 Secondary ammo

The secondary ammo is a level of ammo for secondary firing mode for the weapon the player is currently holding.

4.3.11 Small armour

Small armour is a level of armour from Shield Packs. Armour from Super Shield Packs is not included. Minimum is 0 and maximum is 50.
4.3.12 Standing and rotating duration

Standing and rotating duration is a measure showing how long was the player not moving and rotating at the same time. Standing is defined as the state when velocity equals zero. Rotating is defined as a state when rotation is changing. Standing and rotating duration is how long was the player consequently standing and rotating.

4.3.13 Standing duration

Standing duration is almost the same as previous except of the condition for rotating. Standing duration is a period of time the player was not consequently moving it means the velocity was zero.

4.3.14 Time for switching picked up weapon

In the default settings of the Unreal Tournament when the player picks up an weapon and the weapon is better than the one the player is holding it will switch automatically. But this behaviour can be shut off. Especially some of bots do not switch new better weapon instantly but only when they needed.

Time for switching picked up weapon is a measure showing how long it took the player to switch the picked up weapon. So if the player picks up a weapon which he is not actually holding I measure the time till he switch to this weapon. If the player switches right to the picked up weapon, the time will be smaller otherwise it will be higher.

4.3.15 Time from last turn

Time from last turn is the time elapsed from the previous turn. Turn is defined precisely above. If it is high, it means that the player did not turn for a long time. On the contrary, it will be lower if the player rotated recently.

4.3.16 Ultra damage time

*Ultra Damage* (also known as *Damage Amplifier*, *Amplifier* or *UDamage*) is a very desired power up. It doubles the damage of all weapons for 30 game seconds. The player can not have more than 30 game seconds so the minimum is 0 and maximum 30.

Ultra damage time measure is showing remaining Ultra Damage time. If the player do not have Ultra Damage the time is 0.

4.3.17 Is alternative firing

It is a categorical measure where possible values are true or false. The information is contained in the incoming synchronous self message. If the player is firing witch an alternative firing mode, it is true and if the player is not firing with an alternative firing mode it is false.
4.3.18  Is crouched

Crouching is an action when the player lower to the ground. This make his body
less exposed but the speed of movement is than lower. Is crouched is a categorical
measure whose value is saved with every incoming synchronous message. It is true
if the player is currently crouched and false otherwise.

4.3.19  Is shooting

Shooting means that the player is firing either with primary firing mode or with
the secondary firing mode. Is shooting is also categorical saved with every Self
message. It is true if the player is currently firing and false otherwise.

4.3.20  Is standing

Standing is defined as a state when a velocity of the player is zero. If the velocity
is zero, it is true if it is more than zero it is false. Value is saved with every Self
message.

4.3.21  Is standing for a long time

Is standing for a long time is defined as a state when the players velocity is zero
longer than certain time limit. The time limit can be adjusted. Every time the
player is continuously standing longer than a certain time limit it is true. If the
player is not standing or is standing for a shorter period of time, it is false.

4.3.22  Is walking

Default style of movement is running but the player can also use walking. It is
slower movement than running but it has a few advantages. For example the
player can not fall over an edge. Is walking is true if the player is currently
walking and false otherwise. It is saved synchronously with every Self message.

4.3.23  Rotates to enemy after hit

Rotates to enemy after hit is defined as follows. The player is in calm mode. It
means he does not see any other players and he is not shooting. In the meantime
he is hit by another player. I measure if the player rotates more than certain
angle in less than certain time. Both can be adjusted.

If the player in the calm mode after being hit by an enemy behind rotates to
the enemy in certain time the measure rotates to enemy after hit is true. If he
does not rotate, it is false. The measure is intended to measure reactions on the
hit from an unseen enemy.

4.3.24  Rotating when running

Rotating when running is a categorical measure. It measures if the player makes
turns during running. The turn was already defined. Every time the player turns
the value is true and if he does not it is false. Disadvantage of this can be that
there are lot more false values since the true is saved only after one successful turn
but false is saved with every Self message excepts of the successful turn. I can
solve this by filtering false values and save only one after certain period of time.
The meaning then will be that false means that there was no turn in last certain
amount of time and true means that one successful rotation was completed.

4.3.25 Weapon

There are various types of weapons in Unreal Tournament 2004. When the player
is spawned he has two basic weapons. It is a ShieldGun and AssaultRifle. The
player can then collect weapons in the environment. Only one weapon can be
used at a time.

Weapon measure shows us weapons usage. With every synchronous Self mes-
sage the weapon information is used. Value of the measure is a name of a weapon
the player is holding right now. If the player has no weapon, the value is None.

4.4 Software for analysing data

I designed an Java application called Analyser for distinguishing the mods, com-
puting the measures and saving results to the CSV file in format for Statistics. I
am now going to describe this application.

From Tournament application we have a data files with saved messages. Our
application is loading these data files and messages one by one and every message
is then processed. Currently the Analyser is using Self messages, Played Damaged
messages and Item Picked Up messages. All others are ignored but if anybody
need them in the future, they can be easily added.

I omitted all matches on capture the flag\textsuperscript{1} maps because some reference bots
was not able to run on these maps since they are death match bots. This caused
problems in comparing data with humans because all humans were able to play
on the capture the flag maps. The problem was that on the histograms of almost
all distance measures for humans there were two peaks but the in the same
measures for bots there was only one peak. The second peak in distances was
present because capture the flag maps are usually bigger and the distances are
longer. This could be possible miss understanding because it can look that these
measures are different but they do not have to be.

There is one shared data object with data structures for each measure in each
mode. One data structure object is holding a mode name, a measure name and
a structure for data objects. One data object is holding a map name, player
name and a value of measure. Also, there is one shared object which is holding
information about players. With every message the corresponding player object
is updated so we have actual information about all the players. With every Self
message the mode of player is determined, measures are computed and values are
saved to the particular data structure corresponding to the mode and measure.
The output is a CSV file in the following format:

\texttt{map, player, measure, mode, value}

\textsuperscript{1}Capture the flag is a type of game where the main goal is to capture opponents flag and
bring it home. It is a team type of game. Every team have one flag.
One record is saved on each line and the output are two files:

- *numericData.csv* holds numeric measures.
- *stringData.csv* holds categorical measures.

It is divided because of the later processing by Statistics application whose architecture require separated files. The Analyser is loading all data from the Data directory. One sub directory is one tournament and each file in sub directory represents one match. Data from one subdirectory are processed together and output is two files. For each subdirectory a different instance of Analyser is made. Data are saved into a Output directory each tournament in separate directory with a name corresponding to the input subdirectory.
5. Statistics

In this chapter, I will describe our application for statistical computation called Statistics. At first, this was a part of the previous Analyser application written in Java. But after I decided to use the R scripting language which is more suitable for statistical computations. Analyser computes all measures in all modes and the input to Statistics is output from Analyser. There are three main R scripts. I will now describe them.

5.1 Statistics

Statistics loads data in similar way as Analyser. One folder in Analyser output directory is one tournament and it is computed separately.

5.1.1 Histograms

At first histograms are printed. Histograms are grouped together to the one PDF file where is one histogram for each measure in each mode. This file is printed for all players and all maps in the tournament, for each map, for each player and for each player on each map.

5.1.2 Quantities

The second function is computing statistical quantities for each measure in each mode. Results are printed to a text file. I compute the following quantities:

- minimum
- maximum
- mean
- standard deviation
- variance
- median
- quantiles

In one file are all measures for each mode and this file is printed also for all, for each player, each map and each player on map.

5.1.3 Akaike information criterion

The third function is determining Akaike information criterion [2] for the following distributions:

- normal distribution
Akaike information criterion gives us a relative goodness of a fit of a statistical model on a data. It is only relative goodness so we can only tell which model is better but not how much. So, our software determine Akaike information criterion for all distributions listed above and select the one with the lowest one. We know that this distribution fits the best to our data.

For every measure in every mode, the best distribution is selected and results are written in file with statistical quantities. This file is generated for all tournament, for each player, for each map and for each player on each map.

Sometimes the Akaike information criterion can not be computed. This is because the model of given distribution can not be found. In this case the distribution is listed in separate list telling the fit was not possible.

5.1.4 Parameters estimation

The fifth function is connected with Akaike information criterion. It allows us to estimate parameters of given distribution. To better see how these distributions with estimated parameters look like our software prints a histogram with superposition of all distributions. This graph is saved similarly as histograms. All measures in all modes are in one PDF file and this PDF file is printed for all tournament, for each player, for each map and for each player on each map.

5.2 Comparison

Comparison script is determined to show differences between humans and bots. If we are trying to compare two histograms it is a little bit uncomfortable. Comparison script do a superposition of corresponding histograms so we can exactly see the differences.

The form is one PDF file for one comparison. In this file there is a graph for each measure in each mode. Comparison file is printed generally for all reference bots compared to human and then for each version of our improved bot compared to human. There is also one file comparing all improved versions of our bot to human so we can see the progress.

I will be calling these graphs comparison graphs.
5.3 T-test

T-test [8] is a method allowing us to verify a hypothesis that means of two random variables are the same. The most important is a p-value of the test. If the p-value is below the threshold I have selected the hypothesis is rejected. The threshold is called a significance level and can be adjusted. By default it is 0.05.

I performed t-test testing hypothesis that two same measures in the same modes have the same means. All measures in all modes are grouped together in one file. There is one file for each reference bots versus humans and one file for each improved bot versus humans.

5.4 Another scripts

There is also one file which is used by all the other scripts. It is called settings.R. It can automatically determine the number of measures and adjust the sizes of the PDF files. Also it makes list of players so we do not have to do it manually. In addition there is a script all.R which will run all the scripts. In the root of Statistics there is a read me file with requirements and instructions for running.

I also made a utility for performing t-test for any two data vectors we select. It could be for example two bots. It can be modified to either test a numeric or categorical data. The name of this file is a hypothesis.R.
6. Methods of implementation

Now, we have measured and statistically processed data from both bots and humans. This should be the base for our implementation to the initial bot. In the following chapter, I will describe my methods I use for this implementation.

6.1 Work flow

To start with, I will choose initial bot to implement our improvements. The initial bot should be from the group of measured reference bots since I want to have data from it.

6.1.1 Idea

I will come up with an idea why focusing on any particular behaviour of the initial bot. This idea could be very general.

6.1.2 Comparison

Now I will have two possibilities. The first option will be to compare statistically processed data of the initial bot with humans and choose some measure in some mode that is different and I want to improve it regarding an idea. With improve I mean that the new improved version of the bot will be closer to humans in given measure and in given mode.

The second option is that I will try to come up with a new measure showing the differences between the bot and humans according to our idea. This option proved itself as a more effective one.

6.1.3 Theory

After I will make a hypothesis why this is different. It is still a hypothesis because I do not know whether it is true. Considering this hypothesis, I will try to implement a improvement to the initial bot.

6.1.4 Measuring data

Then, I will measure a data for the improved bot. For measuring data I will need to make a new tournament of the bots. The conditions should be as similar as possible to the original one. So, I will do it exactly the same except of the fact that instead of the initial bot I will put there the new improved bot.

As I have mentioned the tournament will be similar to the previous one for the bots. It means twenty matches, frag limit 25 and no special setting. It needs to be performed manually. Then, I will analyse data with Analyser and statistically process with Statistics.
6.1.5 Theory confirmation

At this point I will again compare our improved bot, the initial bot and humans. If there is an evidence that the improvement make the initial bot closer to human the hypothesis is true. If there is no evidence, this does not have to necessarily mean that our hypothesis was false. In can be a mistake in the measure definition or in parameters setting or it can be because of conditions of a tournament. So, there will be place for discussion.

6.1.6 Repeating

After this, I will continue again with idea. At the end, I will have the bot with certain amount of improvements and data showing goodness of these improvements.

6.2 Possibilities of comparison

To compare data, I will use the output from Statistics application. We can use histograms, statistical quantities, distribution and parameter estimation, t-test and comparison graph\[\text{\textsuperscript{1}}\]. This gives us a wide range of tools for comparing measures. We will show an example of using these methods later.

If we are using a t-test for numeric measures, we just take results from Statistics application. But we are also using a t-test for categorical measures of boolean values. At first, we take a measure of boolean values which is actually a vector and we convert it to a vector of numbers. True is one and false is zero. Then, we can perform a t-test on two of these vectors. The meaning of mean of this vector is ration of true values to false values. In t-test we are always comparing measures containing data from whole tournament. It means from twenty matches.

\[\text{\textsuperscript{1}}\text{Comparison graph is a graph with a superposition of two same measures in the same modes but for a different subjects. We can for example see in one graph the compared bots with humans in particular measure and in particular mode. It is a comfortable tool for comparing two or more subjects with each other.}\]
7. Initial bot

For the improvements I chose a GladiatorBot from David Holaň. In the following chapter, I will describe the reason I chose this bot and also abilities of this bot. I will also point out weaknesses of the bot considering human-likeness.

7.1 About GladiatorBot

GladiatorBot was designed and programmed by David Holaň. David Holaň is a student of Faculty of Mathematics and Physics at Charles University in Prague. The bot is a death match bot and it is written in Java using Pogamut platform.

I contacted David Holaň and asked for permission to use GladiatorBot in our thesis and later in the 2K BotPrize competition. Permission was granted under condition that the new improved bot will have different name. The name I choose is simply Human-like bot. The binaries and source code of GladiatorBot can be downloaded from Pogamut repository.

7.2 Reason of choosing GladiatorBot

At first, I will explain the reason why I did not start from sketch and programmed my own bot. If I use the completed death match bot, I have more time to focus on my main job. It is to create a human-like bot. I can concentrate to the humanness point of view and I do not have to loose time with programming basic behaviour of the bot.

I chose GladiatorBot because it is the most advanced death match bot we have on our faculty. I was also considering the example HunterBot. But HunterBot is too simple and it would take a lot more work. From KnightHunter we do not have source codes.

7.3 Abilities

GladiatorBot is a objective bot with three following objectives:

- Collect items objective
- Attack objective
- Scan objective

It switches between them according to a situation. We will now describe them in detail.

7.3.1 Collect items objective

Collect items objective is designed for collecting items. It has advanced system for choosing which item is best to pick up according to priorities. Priorities are
based on needs of bots, on a distance, on a spawn chance etc. GladiatorBot do not collect items it do not need.

Very interesting is system for estimation of time a particular item will be spawned. The bot observes items in its surroundings and remember which are spawned and which are not. Then it computes spawn chance of the item a take it into account while determining pick up priorities. Good strategy of pick ups is important and it increases performance of a death match bot.

7.3.2 Attack objective

If GladiatorBot see any enemy the attack objective will be set. The first important part is a weapon selection. GladiatorBot has important information about all weapons and is able to choose the best weapon in given circumstances. It regards many factors in weapon selection. For example a distance, a range, a projectile speed, a head shot damage etc.

The second important part is attack movement. GladiatorBot has two attack movements. First is a approach when the bot is trying to get closer to its enemy. The second is a flee mode when the bot is trying to get farther from its enemy. There is a system for choosing strategy according to a lethality of the actual weapon in closer distance compared with farther distance.

7.3.3 Scan objective

The third objective is a scan objective. During this objective the bot is not moving and it is just rotating around. This objective is selected if no other objective is meaningful. It means that there is no item to pick up and no enemy to attack.

7.3.4 Choosing objectives

The objectives are chosen according to the priorities. The scan objective has the lowest priority and the priority of attack objective and collect items objective is computed dynamically considering many factors as distance, amount of damage required to kill, frag priority, pick up priority, health etc.

7.4 Disadvantages

All disadvantages in view of humanness follow from the fact that GladiatorBot is a death match bot. Its main goal is to kill as many opponents in the shortest time as possible. It can use all possible resources and it has abilities which humans do not have.

As an example we can mention that the bot every certain amount of time make a full turn or just half turn without even changing its direction of running. It is very useful for death match bot because it always knows what is around him. But to make a full turn while still running in one direction is very difficulty done by humans since they have only four direction keyboards.
8. Examples of statistical comparison

In this chapter, I will try to show what methods can be applied to compare two measures. It is also intended to show some interesting facts. There is no space for comparing bots and humans regarding all measures, so I will present one example of measure which is the same for both humans and GladiatorBot and one which is different.

8.1 Distance from enemy

From the Akaike information criterion follows that for both humans and GladiatorBot a measure distance from enemy in approach mode is closest to Weibull distribution with mean 909 units for humans and 930 units for GladiatorBot. As I have explained before a difference 100 units is one step so difference 21 is very little distance. So we can say that distribution of distance from enemy during calm mode is almost the same for humans and GladiatorBot.

Now, if we take a look on the results of two sample t-test testing hypothesis that a mean of a measure distance from enemy in approach for humans has the same mean as the same measure for GladiatorBot we can see that p-value of this test is 0.67. With significance level 0.05 we are confirming null hypothesis that the means are the same.

It is obvious on histograms. In figure 8.1 we can see distance from enemy in approach mode for humans and in figure 8.2 we can see the same measure for GladiatorBot. In those figures we can see also all distributions from which we were selecting using Akaike information criterion.

8.2 Distance from conflict point

We have found that distance from conflict point\textsuperscript{1} during evade is closest to logistic distribution but the same measure for GladiatorBot is closest to exponential distribution. A mean for humans is 532 game units and for GladiatorBot it is 167 game units.

If we perform two sample t-test testing null hypothesis that these two measures are the same, we get p-value $1.2 \cdot 10^{-22}$. So on a confidence level 0.05 we reject the null hypothesis that these two means are the same.

Now we will look at histograms. In the figure 8.3 we can see distance from conflict point during evade for humans and in the figure 8.4 the same measure for GladiatorBot. We can clearly see the difference which is that GladiatorBot is usually closer to a conflict point than human. This is interesting because it shows that the bot is more static during a fight in opposite to humans who are usually moving farther from a conflict point. Hence, we can assume that humans are more dynamic during fight.

\textsuperscript{1}To remind a conflict point is a location where the last fight started.
Figure 8.1: Distance from enemy in approach mode for humans.

Figure 8.2: Distance from enemy in approach mode for GladiatorBot.
I was just outlining the methods of statistical comparison which we are using. We did not implement any improvements getting the second measure closer to humans since we were focusing on other measures which we considered more important.
Figure 8.4: Distance from conflict point in evade mode for GladiatorBot.
9. Human-like bot 1

In the 2K BotPrize the Pogamut of version 3.3.0 is used but GladiatorBot was written for Pogamut 3.2.0. So I needed to refactor GladiatorBot to the newer version.

I created this chapter because Pogamut is changing very often and by refactoring GladiatorBot we got a little bit different bot than the initial one. The main changes were in a navigation of the bot. In the new version of Pogamut stuck detectors are registered automatically and also path executor\(^1\) is initialized automatically.

There is also a new listener for path executor state change. In GladiatorBot it was done manually by checking a path executor state in each objective. Now we do it centrally in the main class for all objective. We react on messages as path computation failed, stuck and so on.

9.1 Results

After I made these changes, I decided again to measure data. It is for the reason explained previously. The bot has change a little bit despite the fact it was only refactoring. The example of the change can be seen by comparing a figure\(^{9.1}\) showing measure standing for a long time during approach for GladiatorBot with the same measure for Human-like bot 1 in the figure\(^{9.2}\). The possible interpretation of this is different navigation in the new version of Pogamut which is then causing a little bit different behaviour.

\(^{1}\)A patch executor is responsible for navigation of the bot through an environment. The programmer can give it the target point and it will take care about getting a bot to it.
Figure 9.1: Is standing for a long time in approach mode for GladiatorBot.

Figure 9.2: Is standing for a long time in approach mode for Human-like bot 1.
10. Human-like bot 2

In this chapter, I will describe my first improved bot. I say first because Human-like bot 1 was only refactored version of GladiatorBot. I was improving the bot by replacing the scan objective with purpose to improve measure standing and rotating duration.

10.1 Idea

The first suspicious behaviour I mentioned during observing Human-like bot 1 is scan objective. It seems that this behaviour is not human-like. To remind during scan objective the bot is not moving and is constantly rotating.

10.2 Comparison

I have the following measure regarding this behaviour:

- Standing and rotating duration

As we can see from figure [10.1] there is a obvious difference between Human-like bot 1 and humans regarding the measure standing and rotating duration in calm mode. Humans seem to be standing and rotating less what is the fact we were expecting in the idea.

If we perform two sample t-test on this measure testing the null hypothesis that means for humans and Human-like bot 1 are the same we get a p-value $5.5 \cdot 10^{-6}$ so on confidence level of 0.05 we reject the null hypothesis. The mean for humans is 0.15 but the mean of the same measure for Human-like bot is 0.21.

10.3 Theory

I implemented two new objectives and deleted scan objective. Deleting scan objective should reduce the standing and rotating time. I called the new objectives free time objectives because they are chosen after the bot do not have anything to collect or anybody to attack. To simulate human behaviour we choose between these two objectives randomly. After a certain time, the bot will switch to the second objective. This time is also randomized to eliminate any patterns in the behaviour of the bot. These two objectives are the roam objective and camp objective.

10.3.1 Roam objective

Roam objective is just random movement around the map. At first the bot chooses a random navigation paint and then go there. After it reaches the navigation point, it chooses another one.
10.3.2 Camp objective

There is a type of navigation points called AIMarkers which marks some interesting point in the environment. This could be sniping spot, ambush spot etc. The environment provides us also recommendation for a rotation ideal on this spot.

Camp objective is composed from two main parts. First part is to get to any AIMarker. The AIMarker is chosen in half cases the nearest one and in half cases the random one. When the bot reaches the AIMarker it will rotates to the provided rotation and wait. More likely it will get a chance to snipe or ambush somebody.

10.4 Theory confirmation

As can be seen from the figure 10.2 our theory did not change the density graph. If we take a look on the two sample t-test testing a null hypothesis that means of the measure standing and rotating for the humans and improved Human-like bot 2 are the same we get the p-value $7.7 \cdot 10^{-9}$. It is also obvious from the means which are for humans 0.12 and for improved Human-like bot 2 0.22.

So unfortunately, according to the previous facts, we should disapprove our theory. But I think that the reason that replacing scan objective had no impact on the measures is not a wrong theory but wrong conditions of tournament. To be more precise I mean a number of players in the tournament. As I have mentioned previously the scan objective is executed only if the bot has nothing to do. It
means that it does not see any enemies and does not have anything to collect. But if there are a lot of players in the game, this state does not have to come. Simply the bot has always anything to collect or anybody to attack. But if the bot will compete in any competition where there will be less players (for example one on one) and this behaviour could possibly be activated and immediately reveal the bot.

To summarize the results I did not approved our theory. But, I cannot say that my theory was not right because of the given circumstances of our tournament.
11. Human-like bot 3

In the following chapter, we will look at Human-like bot 3. I was implementing an evade mode using visibility module.

11.1 Idea

Inspiration for me were two facts. First one was a brand new module to Pogamut written by Jakub Gemrot. It is a Visibility module and it is able to tell us for any two navigation points if the first is visible from the second. The possibilities are very useful because we now know where the bots enemy cannot see and we can hide there.

The second inspiration was a SQLite bot [7] which was the finalist the 2K BotPrize and has the same feature. But back then this module was not in the Pogamut so SQLite bot had to implement it by its own.

To summarize it, the behaviour of hiding is not typical for bot and is usual to human players.

11.2 Comparison

Here I had a little bit different approach. I did not have any special measures which I was trying to improve. It is because the information about visibility is not available in my measured data from tournament and it would be difficult to additionally recompute it.

But I wanted to implement the evade using visibility module anyway for the reason explained in idea.

11.3 Implementation

I decided to implement evade behaviour to the bot using visibility module. I needed to add a movement strategy to the attack objective. Also the important part was to edit a decision system for choosing movement strategy. I named the new strategy evade.

I needed to add a visibility library to our bot because this module is available only in Pogamut 3.3.1. We are tied by competition rules so we have to use Pogamut 3.3.0 where this module is not available. I just copied module source files to our bot project.

11.3.1 Movement strategy selection

I defined the evade condition where if satisfied, the evade movement strategy is used. My system for determining evade condition considers the following:

- *weapon advantage* is a difference between lethality of my weapon and lethality of a weapon of enemy

- *health advantage* is the bot health level
• armor advantage is the bot armor level
• ultra damage advantage is the remaining ultra damage time

Overall advantage is computed as a weighed sum of these advantages. The priorities of particular advantages can be adjusted. If overall advantage is below certain threshold the evade condition is satisfied.

11.3.2 Evade movement strategy

The evade movement strategy is to get to the nearest navigation point where the bot can hide from its enemy. If the enemy is chasing our bot, it will continue evading. After the bot gets to the point where it is hidden from the enemy, the attack objective is abandoned. If the bot now switches to the collect items objective, there is a very high probability that it will return to its enemy. So I needed to create a new objective.

11.3.3 Evade objective

The evade objective is pursued only after abandoning attack objective where the last movement strategy was evade. The main reason is to continue evading to the safety. So, the bot will choose the farthest navigation point which is not visible by the enemy and go there. It will choose the navigation point where the path is not near the enemy. If the bot is cornered it must choose path near the enemy.

The priority of this objective is decreasing with time. It is similar to cooling down. So, with time there is a higher probability the bot will switch to the another more important objective. For example to collect a health pack.
12. Human-like bot 4

This chapter is concerning the next version of Human-like bot. I was improving a measure rotating when running.

12.1 Idea

I focused on rotations of the bot during running. This behaviour is very useful for death match bot but is very hard to be done by humans. The bot is running in the same direction during making a full or half rotation. Because humans usually controls the avatar witch only four direction keyboards it is very hard to make a smooth full rotation while running still in one direction.

12.2 Comparison

The measure reflecting this behaviour is:

- Rotating when running

The difference is apparent considering the measure rotating when running in a calm mode of the bot. If we look at figure for humans 12.1 and on figure for previous Human-like bot 3 12.2 we can see a difference between them. Human-like bot 3 is more rotating during running.

The mean of the measure rotating when running in a calm mode for humans is 0.346 and the same measure for Human-like bot 3 is 0.402. We performed two sample t-test testing a null hypothesis that the means are equal. We got a p-value 0.022 so we reject the null hypothesis on a confidence level of 0.05.

12.3 Theory

I had a hypothesis that if I remove the rotating during a run, the measures above should be closer to the humans. So, I completely removed this behaviour. I did not erase parts of source code but I added option for approve or disapprove rotations. So, it could be easily switched on by one parameter in case of future use.

12.4 Theory confirmation

We can now compare humans in figure 12.1 with our improved Human-like bot 4 in figure 12.3. Human-like bot 4 is not exactly same as humans but is closer than the previous version.

The mean of the measure rotating when running for humans is 0.346 and the mean of the same measure for improved Human-like bot 4 is 0.366. I performed two sample t-test testing a null hypothesis saying that these two means are equal. The p-value of this test was 0.400 so we accepted null hypothesis on a confidence level 0.05.
Figure 12.1: Rotating when running in calm mode for humans.

Figure 12.2: Rotating when running in calm mode for Human-like bot 3.
Figure 12.3: Rotating when running in calm mode for Human-like bot 4.

The null hypothesis for previous version of bot was rejected and for improved bot it was accepted. From the histograms, we can see that the measure for improved bot is now closer than for the previous one. So, we confirmed our hypothesis and improvement was successful.
13. Human-like bot 5

We will now look at the next improved version of Human-like bot. I was improving a measure that rotates to enemy after hit.

13.1 Idea

I read the transcripts of comments made by judges from the 2009 2K BotPrize [3]. In this document one of the judges (Judge 1) commented one of the bots (anubot) saying that the bot did not turned to him after being shot. This bot get a rating zero which is the lowest possible. It seems that it is expected from human players to turn back after being hit by unseen player behind.

Our bot also has not this functionality. If a opponent shoot our bot in the back and our bot does not see him, it will not turn to him. There is no procedure treating information about being shot by another player.

13.2 Comparison

There is one measure concerning this behaviour:

- Rotates to enemy after hit

If we compare humans in figure 13.1 with Human-like bot 4 in figure 13.2 we can see that humans rotates to its enemy after being shot in more cases than Human-like bot 4.

After performing two sample t-test testing null hypothesis that the mean of this measure for both Human-like bot 4 and humans is the same we get a p-value 0.251. The mean for humans is 0.650 and the mean for Human-like bot 4 is 0.444.

13.3 Theory

I improved my bot to achieve the behaviour. If the bot does not see any other players and suddenly feels a hit, it makes a half rotation for a certain time. If there is an enemy, the bot should switch to the attack objective. I had a hypothesis that adding this behaviour will get our bot closer to humans concerning the measure above.

13.4 Theory confirmation

At first, I performed two sample t-test testing null hypothesis that means for humans and improved Human-like bot 5 are the same. We got a p-value 0.604 which is higher than for humans and the previous bot. The difference is obvious also from means. The mean for improved bot is 0.714 which is closer to the humans with mean 0.650 than the previous Human-like bot 4 with mean 0.444.
Figure 13.1: Rotates to enemy after hit in approach mode for humans.

Figure 13.2: Rotates to enemy after hit in approach mode for Human-like bot 4.
Now when we look at this measure for humans in figure 13.1 in comparison with our new Human-like bot 5 shown in figure 13.3 we can see that our improvement got this measure closer to humans. According to these histograms, comparison of p-values and means we see that the improvement was successful and we confirmed our theory.
14. Human-like bot 6

At this point, I will be concerned with the sixth version of Human-like bot. I was improving the measure time for switching picked up weapon.

14.1 Idea

I noticed that in the default setting of Unreal Tournament 2004 is selected option to automatically switch picked up weapon if it is better. So, human players with default setting will have this behaviour. Our bot do not switch the pick up weapon until it needs it. This could be potentially suspicious for judges because there is no reason in not changing weapon if it is better. The approach with a changing weapon when needed is not very good because switching weapon takes certain time and can slow down the player.

14.2 Comparison

There is a measure showing how long did the player wait before switching last pick up weapon:

- Time for switching picked up weapon

On the figure 14.1 we can see that there is a evident difference between humans and the previous Human-like bot 5. Humans seem to change a weapon immediately more times than Human-like bot 5.

After performing two sample t-test testing null hypothesis that the mean of this measure for both Human-like bot 5 and humans is the same we get a p-value 0.126. To be more precise the mean for humans is 288.6 logic iterations and for Human-like bot 5 it is 383.0 logic iterations.

14.3 Theory

At first, I added a system for deciding whether one weapon is better than the other one. I factor default setting into the system. Then I added behaviour telling the bot to change a picked up weapon immediately if it is better than its current weapon. This should make our bot more human-like.

14.4 Theory confirmation

I performed two sample t-test testing null hypothesis that means for humans and Human-like bot 6 are the same. We got a p-value 0.664 which is higher than p-value of the test for humans and the previous bot. We can also see the difference from means. Mean of tested measure for improved bot is 264.2. To remind for humans it was 288.6 and for previous bot it was 383.0.
Figure 14.1: Time for switching picked up weapon in calm mode for Human-like bot 5 and humans.

Figure 14.2: Time for switching picked up weapon in calm mode for Human-like bot 6 and humans.
From the figure 14.2 we can see that improvement made this measure much closer to humans. The improvement according to the comparison of mean, p-values and density graphs was successful so we confirmed our theory.
15. Study with human players

In the following chapter, I will describe study of the bot’s believability on human players. I will describe how the study was made and what were results. Then I will mention comments by humans to the bot and at the end I will discuss the results and these comments.

15.1 Method

The believability of the bot was tested during a study with human players. One human player was playing a death match against the bot on ten different maps. To ensure anonymity both players has a default skin and their name was set to different random number before every match. In another room, there were three judges who were observing the game by connecting to the server as spectators. Their task was to decide in every match who had been a human and had been a bot and also to write a reason why they think so.

15.2 Results

The results were not very satisfying because they recognized the bot in every match. But the important fact is that in most cases they recognize the bot because it got stuck. With getting stuck there is always a problem and it is extremely difficult to completely eliminate it. It is partly because this problem rises from Pogamut itself. In this project we were not focusing on low level movement of bot. I was just using internal Pogamut methods for moving from a point A to a point B. Sometimes the problem is even with maps where could be mistakes in navigation graph. In the description of future work we will outline possible solutions.

15.3 Comments

The reasons why did judges think that it was a bot are following (sorted by frequency of comment):

1. The bot got stuck on the wall or another obstruction. This composed from a few particular situations:
   - The bot was trying to run through an obstruction.
   - The bot was not moving for a longer period of time
   - The bot was not moving for a longer period of time and was only jumping.
   - The bot stopped for a while then jumped and continued.

2. The bot was stopping for a short period when collecting items.

3. The bot was running in a cycle between two close locations.
4. The bot got stuck on the lift. This compose from two particular situations:

- The bot was waiting for a lift under the lift platform when the lift was up.
- The bot took the elevator up then jumped down and then took the elevator again.

5. The bot was static during the fight.

15.4 Discussion

As it was already mentioned, all these problems except one were connected with low level movement. When the judges were asked about another reason why they recognized the bot there were none. They spoke well about hiding, weapon selection and good aim. At the end, I want to highlight the fact that nobody ever won the main prize in the 2K BotPrize competition so we should not suspect that making a human-like bot is an easy challenge.

1 With low level movement we mean the basic movement from one point to another point.
16. Conclusion

I will now summarize and evaluate my work. Initially, I will start with results of my project. Then, I will discuss my thesis. And at the end, I will sketch a direction of a future work towards participation in the 2K BotPrize competition.

16.1 Results

The result of my thesis is on the first place human-like bot and also a massive background for measuring, analysing and statistically processing data and comparing differences between players. I used it during improving the initial bot I have selected.

My background system consists of three applications. Tournament for measuring game play data which allows us to save a whole match into a text file. Then Analyser for computing measures I have defined. And in the end Statistics determined to statistically process measures from Analyser. Output from Statistics make it possible to compare measures among different tournaments, players, maps or even players on a particular maps.

Human-like bot was improved six times. First was refactor to a newer version of Pogamut used in the 2K BotPrize competition. The second was removing scan objective and replacing with two new objectives. The third was adding a evade mode using hiding from enemy. The forth was removing of unnatural rotation during running. The fifth was adding a behaviour of turning around after being hit to the back. An the last sixth was improving changing better weapon immediately instead of keeping worse weapon.

I have also measured and analysed a lot of data for humans and bots which can be used for another projects where is needed information about how humans or bots play first person shooters.

16.2 Discussion

The main goal was to make the bot as similar to human controlled players as possible. I measured similarity using a statistics tools and also I did a study with humans. If factor were only in improvements, I did three from total number of five improvements were successful and made our bot closer to humans. However the study with humans player showed that it will be still a long way to create truly believable bot. I will have to make more improvements in order to outwit judges especially in the field of movement.

16.3 Future work

I would like to continue with proposed method of improvements towards the 2K BotPrize competition held in September 2012. I will be looking for more differences and I will try to improve our bot to be more similar to humans.

The first think to do would be automatizing whole system of measuring, analysing and statistically processing data to be easier to use. At first it would
be good to automatize Tournament application. An ideal way would be to just give it a list of maps and bots and run it. It would run all matches on given maps with given bots and for each match record a server side demo and save messages. Currently it is not possible because of NeuroBot but the authors promised to add this future to the new version. This could definitely save so much time because now it has to be done manually. The second think would be connecting Analyser and Statistics to be able to run at once. It could be solved by running R scripts from Java. To sum it up, it would require only two actions to get demanded output. To run Tournament and then run Analyser. To connect Tournament and Analyser would not be a good idea because if the measure will be added all tournaments would have been repeated. Also it would be nice to optimize Analyser to be more flexible for adding new measures.

Regarding the Human-like bot the first think which has to be done is to solve a problems with movement. First problem which could easily reveal the bot is stuck. This problem is based in Pogamut platform and it is very difficult to completely remove it. However, it can be solved either by editing Pogamut classes or using special libraries determined to solving this problem. One example of this library is Jungigation library which can be used to recompute navigation graph by exploring it with sampling bot and then use it for bot’s navigation. The principal of sampling bot is to erase those edges in navigation point where it got stuck. Usually the bot is stuck on the same place so this could possible make this problem less obvious. This library cannot be used in this thesis because it was not fully completed yet.

The second biggest problem is that the bot is stopping while collecting items. This is something what people definitely do not do and it has to be solved in order to compete in any competition. This problem is also based on Pogamut but it can be solved either by overriding part of Pogamut or adding a new parts. To give a direction, it can be solved by extending or editing path executor.

The next possible improvements are not so important as the previous one but it could be interesting to take a look at them. Initially there is an idea of adding attack movement as dodging, jumping or moving around enemy. Also the bot could from time to time look around during a fight to see if there is not another enemy which is closer. The next idea is to use the visibility module for planned ambushing. The bot can wait for someone hidden behind obstruction and surprise enemy. The visibility could be also used for covering during a fight. There should be also implemented behaviour for projectile avoidance off course considering reaction time of humans. Maybe it could be interesting to try implement crouching. The using of crouching was not confirmed during executed experiments but we can look at professional players. Also some people are using double jumping or just jumping for faster movement and for lowering probability of being shot. Very important and interesting point would be to implement movement besides navigation graph. Next suspicious behaviour is that the bot do not collect weapons which has been thrown out by another players usually after their death. Then to get the bot closer to professional player it has to be using various types of combos witch adrenalin combos beginning and weapon combos ending. The last think which worth testing a hearing. This is very typical for humans and it can possibly reveal the bot.

There are also other small bugs in the bots behaviour that will have to be
fixed. It is for example wrong navigation on certain maps\footnote{DM-DesertIsle, DM-Rrajigar and DM-1on1-Serpentine} or the fact that the bot sometimes open fire through wall or is shooting totally away from enemy. And besides, the bot is at some point shooting after re spawn what is not good. Then, the priorities especially during evade should be tuned up to achieve plausible behaviour. It means for example when is it human-like to collect certain item during evade and so on. In the end, the bot is sometimes stuck on some item which is not present but the bot probably computed that there is high chance of spawning. But then, it sees that there is no item and make another plan. After that, it computes that there is a high chance of spawning again a goes back. This leads to cycle in the bots behaviour and it is a clue for judges.
Bibliography


List of Figures

1.1 Unreal Tournament 2004 from Epic Games . . . . . . . . . . . . . 5
1.2 Pogammut architecture. Reprinted from [15] with permission from
the authors. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
8.1 Distance from enemy in approach mode for humans. . . . . . . . . 36
8.2 Distance from enemy in approach mode for GladiatorBot. . . . . . 36
8.3 Distance from conflict point in evade mode for humans. . . . . . . 37
8.4 Distance from conflict point in evade mode for GladiatorBot . . . 38
9.1 Is standing for a long time in approach mode for GladiatorBot . . 40
9.2 Is standing for a long time in approach mode for Human-like bot 1. 40
10.1 Standing and rotating duration in calm mode for Human-like bot
1 and humans. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 42
10.2 Standing and rotating duration in calm mode for Human-like bot
2 and humans. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43
12.1 Rotating when running in calm mode for humans. . . . . . . . . . 47
12.2 Rotating when running in calm mode for Human-like bot 3. . . . 47
12.3 Rotating when running in calm mode for Human-like bot 4. . . . 48
13.1 Rotates to enemy after hit in approach mode for humans. . . . . 50
13.2 Rotates to enemy after hit in approach mode for Human-like bot 4. 50
13.3 Rotates to enemy after hit in approach mode for Human-like bot 5. 51
14.1 Time for switching picked up weapon in calm mode for Human-like
bot 5 and humans. . . . . . . . . . . . . . . . . . . . . . . . . . . 53
14.2 Time for switching picked up weapon in calm mode for Human-like
bot 6 and humans. . . . . . . . . . . . . . . . . . . . . . . . . . . 53
List of Tables

3.1 List of matches ........................................ 17
Attachments

Attachment A: DVD 1

On enclosed DVD 1 there are the following directories:

- Applications contains all applications and their output.

In addition there are the following files:

- Thesis.pdf is this thesis in electronic PDF version.
- Programmer documentation.pdf is a programmer documentation in PDF.
- User documentation.pdf is a user documentation in PDF.

I will now describe content of Application directory.

Applications

In Applications directory can be found the following:

- Tournament contains a Java application for measuring game play data.
  - src contains a source code of the application.
  - target contains compiled code, jar and javadoc.

- Analyser contains a Java application for computing measures and its output.
  - Data contains input data from Tournament. Each subdirectory is one tournament and each file is one match.
  - Output contains a output which is as well input to Statistics. Each subdirectory is one tournament and contains two files. First is a numericData.csv containing numeric measures and the second is a stringData.csv containing categorical measures.
  - src contains a source code of the application.
  - target contains compiled code, jar and javadoc.

- Statistics contains R scripts for statistical processing data from analyser and output.
  - readme.txt is a manual how to run the Statistics application.
  - run.bat is a bat script to run all the R scripts.
  - src contains a source code of the application.
  - target contains an output from application.
    - Comparison contains a comparison of all measures in all modes.
      It compares bots to humans.
* **Statistics** contains histograms, estimation of distributions, statistical quantities and Akaike information criterion for each tournament, player, map and player on map.

* **T-test** contains results from t-test between bots and humans.

- **Human-like bot 1** contains source code from the first version of Human-like bot.
  - `src` contains a source code of the application.

- **Human-like bot 2** contains source code from the second version of Human-like bot.
  - `src` contains a source code of the application.

- **Human-like bot 3** contains source code from the third version of Human-like bot.
  - `src` contains a source code of the application.

- **Human-like bot 4** contains source code from the forth version of Human-like bot.
  - `src` contains a source code of the application.

- **Human-like bot 5** contains source code from the fifth version of Human-like bot.
  - `src` contains a source code of the application.

- **Human-like bot 6** contains source code from the sixth version of Human-like bot.
  - `src` contains a source code of the application.
  - `target` contains compiled code, jar and javadoc.

**Attachment B: DVD 2**

On enclosed DVD 2 there are the following directories:

- **Tournament** contains data from all tournaments.

- **Bots** contains all reference bots.

I will now describe a content of each directory.
Tournament

Contains measured data from all tournaments and everything connected with tournaments. There are following subdirectories:

- *Bots* contains measured data and demos from bots tournament.

- *Human-like bot 1* contains measured data and demos from tournament with Human-like bot 1.

- *Human-like bot 2* contains measured data and demos from tournament with Human-like bot 2.

- *Human-like bot 3* contains measured data and demos from tournament with Human-like bot 3.

- *Human-like bot 4* contains measured data and demos from tournament with Human-like bot 4.

- *Human-like bot 5* contains measured data and demos from tournament with Human-like bot 5.

- *Human-like bot 6* contains measured data and demos from tournament with Human-like bot 6.

- *Humans* contains measured data from humans tournament.
  - *Data and demos* contains measured data and demos.
  - *INI files* contains INI files with settings used during the tournament.
  - *Questionnaire.xls* contains a questionnaire everybody had to fill in before the tournament.

- *Servers* contains scripts for starting servers used in all tournaments.

Bots

Contains all used reference bots. One subdirectory corresponds to one bot.