Using Metrics and Keystroke Dynamics in Games to measure Player Experience

Michael Bauer

MASTERARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Interactive Media

in Hagenberg

im September 2014

© Copyright 2014 Michael Bauer

This work is published under the conditions of the *Creative Commons License Attribution–NonCommercial–NoDerivatives* (CC BY-NC-ND)—see http://creativecommons.org/licenses/by-nc-nd/3.0/.

Declaration

I hereby declare and confirm that this thesis is entirely the result of my own original work. Where other sources of information have been used, they have been indicated as such and properly acknowledged. I further declare that this or similar work has not been submitted for credit elsewhere.

Hagenberg, September 29, 2014

Michael Bauer

Contents

Kurzfassung (German)

Diese Masterarbeit befasst sich mit der Möglichkeit ob sich die Erfahrung eines Spielers mit Hilfe von Keystroke Dynamics messen lässt. Spielspaß ist ein essenzieller Teil von Videospielen, gerade im Bereich der Free to Play (F2P) spielen, die Geld von Einkäufen und Langzeitspielern erzielen. Verschiedene Metriken und Keystroke Dynamics zur Messung der Erfahrung von Spielern werden diskutieren und evaluieren.

Abstract

This thesis describes the challenges of adequately measuring experiences associated with playing digital games. Player satisfaction is central to computer games, especially in Free to Play (F2P) games, which gain money through long term playing and micro transactions. It will discuss and evaluate metrics on data, collected by keyboard input, to categorize experience of players, how long they have been playing computer games and at which level.

Chapter 1

Introduction

Player metrics are used in many ways to analyze players or games. For example to reconstruct bugs in the game, to proof that someone is using game hacks or to see how the game is used by the players. For startup companies using the free to play payment model with micro transactions and monthly payment it is more important to keep the customers playing over a long time period. Therefor other metrics have to be sampled, to find out what the main purpose of my customers is and how many features of the game are implemented for them. If the game is too difficult for players he will not stick to it very long. Even with multiple difficulty settings a player can choose a too hard level. By measuring the players experience level we can provide him the best difficulty level to entertain him.

1.1 Problem Statement

After a release the developer only gets limited feedback for his game, for example in forums, polls and the data how the playtime of the game is varying. Not much gamer use the forum to give positive feedback it is mostly used to find solutions for errors and report bugs or unsatisfied balancing values. If some users complain about design decisions you only get a small amount of your customers. According to Kristoffer Benjaminsson (*Easy Studios*) less than 20% are reading forum entries and less than 5% are writing in forums [23]. Also polls on websites are a possible means to collect feedback but it also varies on the amount of gamers attend it.

Interviews with the community are also a method to get an idea how the community thinks about your game and how to improve it. But these interviews get more and more expensive when the game wants to be played by many different personas(details in 2.3). Difficulties are that you have to make an appointment with them and also the geographical distance can increase costs. If you are a small company or develop a game as a student project you will not have the money to afford interviews on a large scale.

Games like *Battlefield 3*[17] collected more than 1TB of data per day [29] these data gets analyzed to find out if some weapons are unbalanced and what are the most popular maps. This information is used to develop new ones with a familiar setup in the DLC's[1]. In multi player games like *World of Tanks*[21] or *Call of Duty* where a match making system allocates a player to a server, some match making system use the metrics data to determine how good a player is and which server he would fit.

In single player games the gamer can choose the difficulty level in the beginning but he cannot imagine how difficult the game gets in higher level. When the game gets frustrating the gamer has often no opportunity to change the difficulty settings during the game and he doesn't want to start the game from scratch again.

1.2 Goals

Using metrics of your past game sessions can give the developer a hint how experienced a player is, but when a player wants to play the first session on a multi player game no data is there to calculate the experience. The gamer can either start with player with low experience or with high experience. But when he is a high experienced player on a low experienced server the enjoyment can be very low until he gets to the high experienced server. This can lead that the player is stopping the game. Like mentioned in single player games the difficulty settings are to choose in the beginning, which can force the player to quit in some stage because the game get to difficult for him. In *Call of Duty: Modern Warfare*[20] the gamer has to master a test level, depending on his time the game suggests a difficulty level.

The goal of this thesis is to find a way to use keystroke dynamics and metrics to measure the user experience with games at a very early stage to use it for difficulty settings in single player games or matchmaking systems for multi player games. For example in a short tutorial the keystroke dynamics can be calculated and compared to pre measured keystrokes, so the players experience can be identified and when he chooses a to hard difficulty the game can warn him that it could be too hard for him and the enjoyment could be too low.

1.3 Structure

The following chapter is about the definition of player metrics and also its subordinated fields of game and performance metrics. Additionally personas and player types will be described and how they influence the design and development process of video games. Not only the design and creation of

[1]Downloadable Content

player data is described, also the visualization and evaluation of this data is described. In chapter Use Cases 3 the history and definition of keystroke dynamics are described and how they can help calculating a user's experience. Keystroke dynamics features will be designed and with data from tester evaluated. At last the feature vector with normalization and the distance measurement are described in that chapter. Chapter Evaluation 4 is focused on testing the distance measurement and discussing the outcome of the test.

At the end in chapter Results and Discussion 5 sums up all the results of this thesis as well as from the user tests. Additionally possible improvements and extensions are being proposed.

Chapter 2

State of the Art

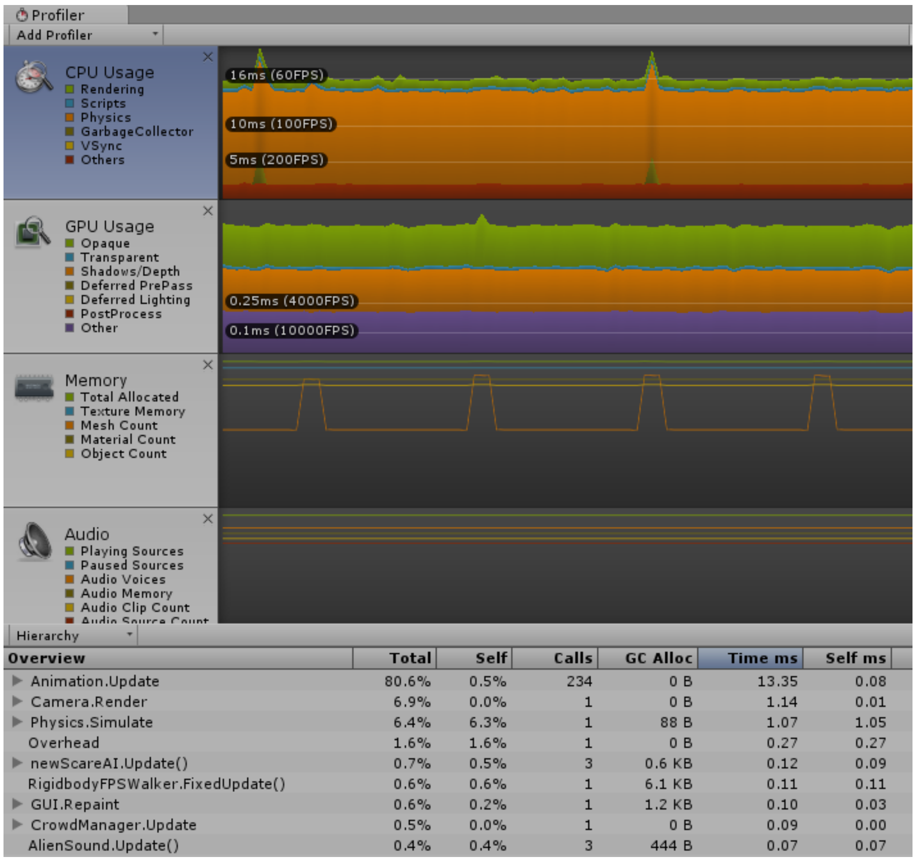
2.1 Game Metrics

Tracking player behavior via metrics so called telemetry is used over several years. For example in 1996 the game Quake[18] produced so intensive telemetry to restore the whole game session. This data was sampled for testing the game not for player analytics. Gathering player metrics become possible when peoples Internet connections increased. And in that time online Games started to rise. For this first simple games financed via Banner commercial tolls like Google Analytics was enough. By the time the games got more complex and in game shops increased the publisher needed their own metrics to analyze the player and to provide the right buy able stuff. Many game metric systems are industry secrets and only some are published. A lot of papers using metrics focus on the User Experience in term of Interaction and Interface design. Some of the principles in user experience design can be used to define metrics to measure player enjoyment in games.

Metrics in games can be categorized into three main types [7]. The main focus will be on 2.1.3 Player Metrics because this will be the most interesting ones to measure player enjoyment[8].

2.1.1 Performance Metrics

As Mark Graham Brown wrote in [4] performance metrics are related to technical and software infrastructure of a game. While consoles like the *Microsoft XBOX* or *Sony PlayStation* have predefined hardware, these metrics are more relevant at the development stage of a game. But the *PC* has a personal configuration of the hardware, and these metrics are relevant to see if the game performs well on different set of hardware. The more powerful hardware a game needs fewer people can play or buy the game. With this metrics a developer has an overview what type of hardware the community of an other game has and can set the minimum requirement for the new game, in a consensus that the game looks perfect and a lot of people can



Overview	Total	Self	Calls	GC Alloc	Time ms	Self ms
▶ Animation.Update	80.6%	0.5%	234	0 B	13.35	0.08
▶ Camera.Render	6.9%	0.0%	1	0 B	1.14	0.01
▶ Physics.Simulate	6.4%	6.3%	1	88 B	1.07	1.05
Overhead	1.6%	1.6%	1	0 B	0.27	0.27
▶ newScareAI.Update()	0.7%	0.5%	3	0.6 KB	0.12	0.09
RigidbodyFPSWalker.FixedUpdate()	0.6%	0.6%	1	6.1 KB	0.11	0.11
▶ GUI.Repaint	0.6%	0.2%	1	1.2 KB	0.10	0.03
▶ CrowdManager.Update	0.5%	0.0%	1	0 B	0.09	0.00
AlienSound.Update()	0.4%	0.4%	3	444 B	0.07	0.07

Figure 2.1: Screen shot of the Unity Profiler, with *S.C.A.R.E* as sample project

run it on their system. Performance metrics are also used to measure the changes of patches and updates to have a very quick feedback if something went wrong.

Software Profiler is also a form of performance metric, which analyzes the time each method or class uses. This gives the developer an estimation which parts of the code are slow and in times of low frames per second to re engineer. Like in figure 2.1 you can see a sample of the *UNITY Profiler*[1], in this example the animations take up to 80% of the calculation time.

2.1.2 Process Metrics

These metrics are related to the development process of games. Creative areas in IT and also game development have necessitated the use of agile development methods, where these metrics measure and monitor the process.

[1] http://unity3d.com/unity/performance

Larry Mellon wrote in [7]:

> “Metrics Driven Development (MDD) is not a process per se, but rather finding a way to qualify your tasks and risks to best focus your team’s limited resources on your project biggest problems.”

Teams can be more efficient when every developer can deal with important tasks. At *Scrum* you have user stories and the associated tasks, which are evaluated for its time used to finish the task. If some days later you find out that your time estimation are to low you can switch developer from an other task to this to finish it in time.

2.1.3 Player Metrics

User data or player metrics are by far the most source of data, derived from the users who play the game. This category is very general so it is spitted in three subcategory’s “Customer metrics”, “Community metrics and “Gameplay metrics” [26].

Customer Metrics

As a customer, users can download and install a game, purchase any number of virtual items from in-game or out-of-game stores and shops, spending real or virtual currency, over shorter or longer timespans. At the same time, customers interact with customer service, submitting bug reports, requests for help, complaints, and so on. Users can also interact with forums, official or not, or other social-interaction platforms, from which information about these users, their play behavior, and their satisfaction with the game can be mined and analyzed. We can also collect information on customers’ countries, IP addresses, sometimes even age, gender and email addresses. Combining this kind of demographic information with behavioral data can provide powerful perceptions into a game’s customer base.

Community Metrics

The community metrics are based on the game play or social related interaction between player, in-game or out-of-game of some combination of. User can interact with each other via in-game chat or voice or out-of-game via live conversation software like *Teamspeak*[2] or game forums.

Data-mining datasets derived from chat logs in an online game can reveal bugs or balancing problems. For example if some enemy is too strong a discussion in the forums or the chat will occur how to deal with that enemy. In some games the player can find some unsolved mysteries which are not explained in the game. Such Easter egg hunts are very popular and create a

[2] http://www.teamspeak.com/

strong fan base. Like in *Battlefield 4* after an update a new assignment was discovered but no one knew how to unlock it. So several people got together and searched for answers in-game and also outside. They discovered some hints for other Easter eggs, especially the Megalodon hunt. The hunt for this giant shark had more than 1000 forum entries, Youtuber like *Jackfrags* had millions of views on their videos and also some people rented Multiplayer server where everybody could help to search for the Megalodon.

Gameplay Metrics

This subcategory of the user metrics is perhaps the most widely logged and utilized type of game telemetry currently in use. Gameplay metrics are measures of player behavior: navigation, item and ability use, jumping, trading, running, and whatever else players actually do inside the virtual environment of a game. Four types of information can be logged whenever a player does something or something happens to a player in a game: What is happening? Where is it happening? At what time is it happening? And: Who is involved?

Gameplay metrics are usually used for informing game design. They can provide data to address key questions like if game features are used like intended, or if there a barriers in the game world or if some places in the game world are over- or underused. These metrics can be recorded during the development and after the release.

Players can generate hundreds of measures over a game session, depending on what you are measuring. As a game designer and also as developer of such metrics you have to consider that these data has to be transferred to the metrics database and the game experience shouldn't be affected. Especially in multi player network games you should consider this problem. If the data is too big to send during playing, it can be sent after playing. In most of the modern games the save game data is synchronized with a server and during that time the metric data could be sent to the database.

Drachen and El-Nasr subdivided in [8] the "Gameplay metrics" into three categories:

- **In-game:** Covers all in-game actions and behaviors of players, including navigation, economic behavior, as well as interaction with game assets such as objects and entities. This category will in most cases form the bulk of collected user telemetry.
- **Interface:** This metrics include all interactions between the player and the game interface and menus, such as game variables and hardware settings.
- **System:** Are all actions done by the game engines and their subsystems triggered by player actions. Like AI system, NPC (non player character) actions and automated events.

Game play metrics are the most important for evaluating user experience and game design, but in the chain of revenue in game development they are under prioritized. They are useful for design, user research, quality assurance or any other position, where the actual behavior of the player is of interest.

Many game genres share some metrics, that are the same but in different genres they have a different meaning. And also between two games in one genre there are data you can not compare, for example in one platform you have to shoot enemies and in the next one you have no enemies. It is possible to compare the same metrics like accuracy, rankings, position data and many more with other genres, but this comparison has no value. It is more important in which context you see your data. For example "Time per Level" for evaluating level design issues you need several metrics to identify problems. If you only measure time you see that they will need longer for this level, but if you add a death count and the reason of it you will see why they needed so long. Finding out how to read the data is the hard part.

2.2 Visualization

The data collected by metrics can even in small games grow to an unusable amount. In my test game 3.3.1 only four keys are used and in two minutes of playing an average of 150 keystrokes are made by the player, now imagine a much more complex game with 10 different keys and mouse control it become very difficult to read specific data out of the file. For better readability and visibility different Methods are used to model the data.

Charts

Charts are the basic and easy to read method to represent tabular numeric data, functions or quantitative data. Different types of graphs, like bar diagrams or line charts, can show data on a very way to see differences between the data in one look. Charts are often used to easy understanding of large quantities and relationships between parts of the data. In game play analytics charts are often used to visualize non positional data, like in figure 2.2 you can see a bar chart representing the percentage of completing the single player campaigns of some games. The exact numbers are not easy to see but differences are very fast to spot.

Heat Maps

As Jonathan Dankoff writes in [25], heat maps are used to represent data in the virtual world. Heat maps are usually overlays on a map where each color represents the amount of data happened on a position. For the color representation different color schemas are used, but mostly cold colors are used for low data areas and warm colors are in high data areas. Spatial

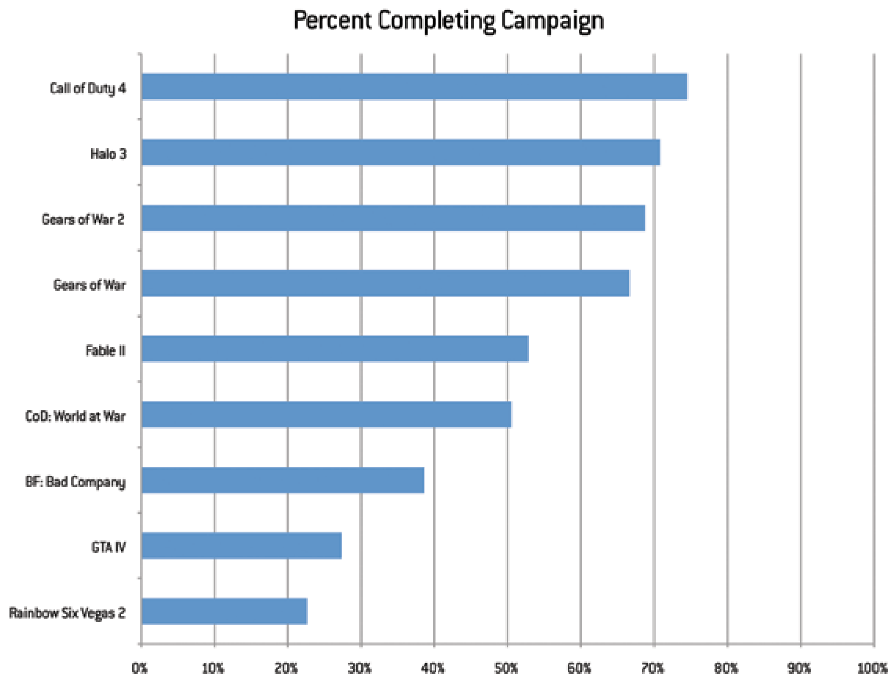


Figure 2.2: Bar chart from Bruce Phillips [28] displaying the percentage of completed single player campaigns.

gameplay metrics are used to create heat maps, because this metrics store the position in the game world.

For game development heat maps can be used in many situations like in jump & run games a heat map where the player dies can be helpful to identify problems in the level design. *Valve* wrote in [30], they used heat maps of 6.5 million bullets fired during a beta test of their game *Counter-Strike Global Offensive*[19] to visualize sight-lines for sniper or narrow points in the map. In figure 2.3 all positions where player fired sniper rifles can be seen, with such a map the level designer can identify if there are unwanted line of sights that can be an advantage for one team.

Trajectory Analysis

Trajectory analysis is the description of an object moving in specific time in space. In computer games this could be the navigation of a player through the game world. Along with the movement all other interactions with the game are stored, for example item pickup, fights or building. With this analysis the game developer can get feedback from witch position the player liked to attack enemies and what weapons are used in that situation. In racing games trajectories can be used to identify if there are paths in curves

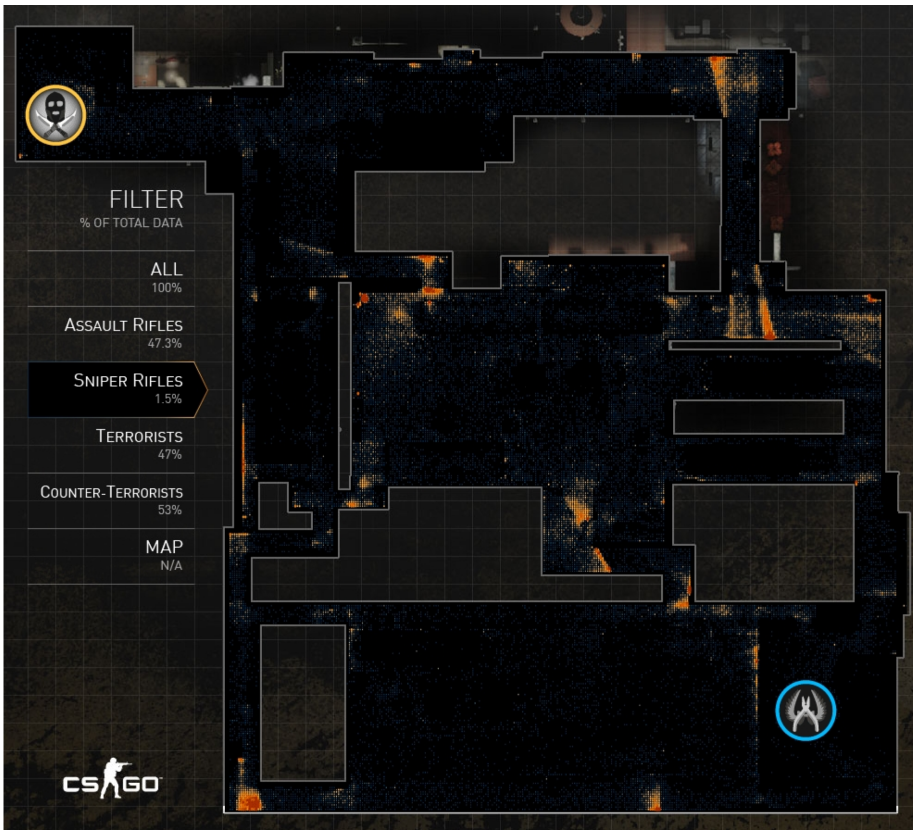


Figure 2.3: A heat map showing the positions where sniper rifles were fired[30].

that player use to be faster, than the original path the game and the track designer intended to be. Or the balancing of the cars can be reviewed, if some cars have a much earlier breaking point than others. Trajectory analysis can also be used to identify illegal bot programs in online games or to study player tactics.

In figure 2.4 the trajectories of 8 players playing Assassins Creed, the level designer intended that the player climb up the right tower and use their parachute to travel down to the left part of the castle. Two players climbed down the walls and went through the valley, which broke the tutorial and lowered their playing experience. So the level designers could find the problems and fix the issue.

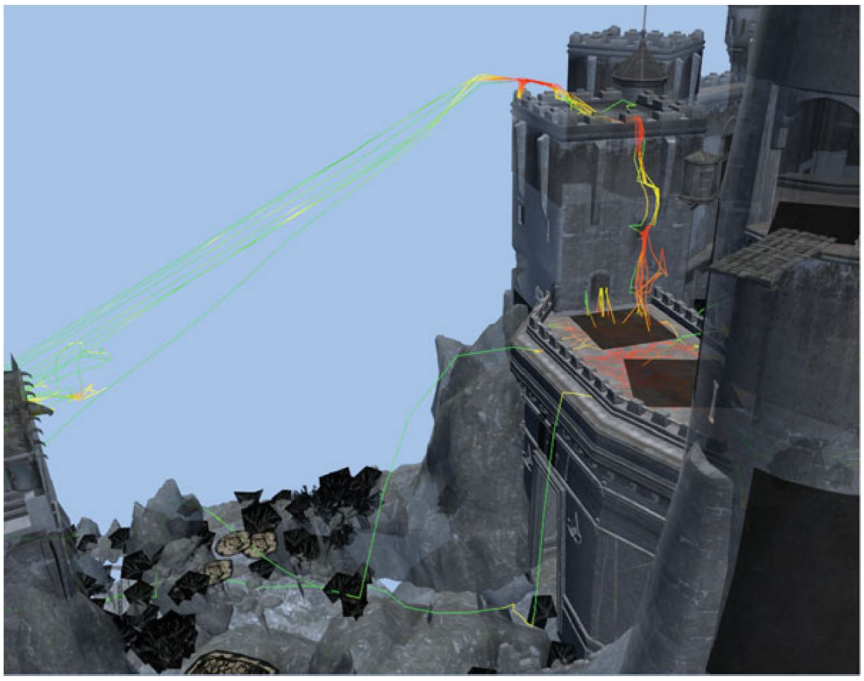


Figure 2.4: A trajectory analysis showing 8 people playing Assassins Creed from [25].

2.3 Personas

Alan Cooper is one of the pioneers in inventing personas. Personas also help prevent some common design pitfalls which may otherwise be easy to fall into. The first is designing for what Cooper calls "The Elastic User" by which he means that while making product decisions different stakeholders may define the user according to their convenience. Defining personas helps the team have a shared understanding of the real users in terms of their goals, capabilities and contexts. Personas also help prevent "self referential design" when the designer or developer may unconsciously project their own mental models on the product design which may be very different from that of the target user population. Personas also provide a reality check by helping designers keep the focus of the design on cases that are most likely to be encountered for the target users and not on edge cases which usually won't happen for the target population. According to Cooper, edge cases which should naturally be handled properly should not become the design focus [24].

You should create you personas for your product, the target audience should be your main focus. For interviews developer can invite people which math with their personas to verify if the decisions made for them are fulfilled.

2.4 Player Types

As we know everybody has different needs and every player plays a game for an other reason. First person shooter, racing games or role playing games are some genres where you can find mostly player only who are only interested in that genre. “Open World” games offer a lot of different player types the opportunity to play their game. Dr. Richard Bartle identified in [22] the following four main player types:

- **Achievers** are competitive and enjoy beating difficult challenges no matter they are set by the game or by themselves. The more challenging the goal, the most rewarded they tend to feel.
- **Explorers** like to explore the world - not just its geography but also the finer details of the game mechanics. These players may end up knowing how the game works and behave better than the game creators themselves. They know all the mechanics, short-cuts, tricks, and glitches that there are to know in the game and thrive on discovering more.
- **Killers** like to cause and provoke drama and impose other players in the scope of the game, especially when there is a lot to lose in the game. Killers know everything about the weapon stats in the game and how to use it.
- **Socializer** are often more interested in having relations with the other players than playing the game itself. They help to spread knowledge and a human feel, and are often involved in the community aspect of the game

In figure 2.5 you can see how the different player types act. The horizontal axis represents a preference for interacting with other players vs. interacting with the world and the vertical axis represents a preference for (inter)acting with something vs. (inter)acting on something. So, achievers prefer to act on the world, while socializers prefer to interact with other players.

The player types are very important for measurements with metrics, because you can not compare the enjoyment of different types with the same metrics.

2.5 Categorizing Player Types/ Personas via Metrics

In Open World games a lot of players can find their play style. But with this variety of play styles it gets harder and harder to compare players. As in 2.4 described you can categorize everyone into a play style group. People of one group are easier to compare because an explorer will travel much more than a socializer, who wants to get in touch with other gamers. Everybody

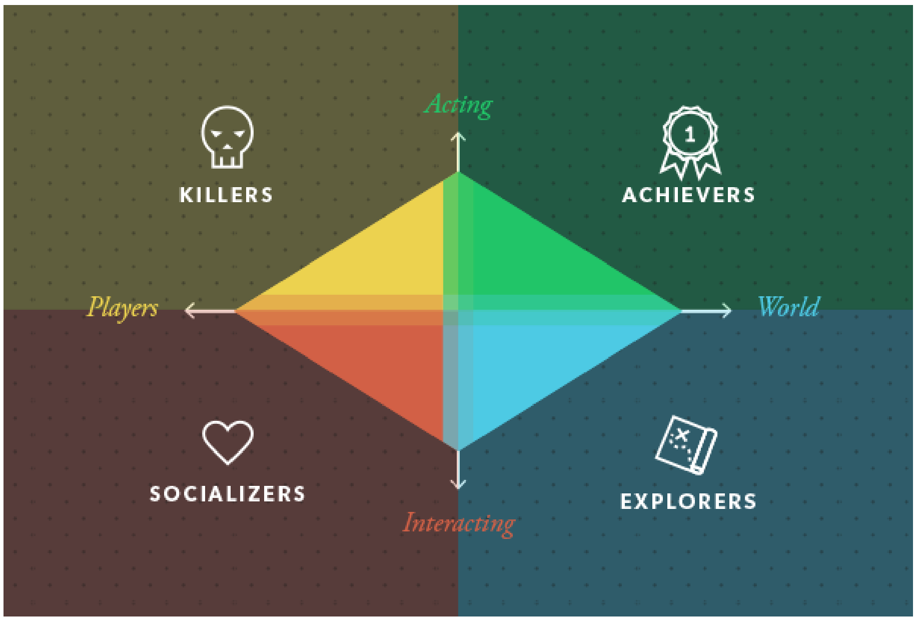


Figure 2.5: Summary of Bartle's player types [27].

has different factors pushing their player enjoyment. Tychsen and Canossa described in [14] metrics for identifying Player Types or Personas.

Navigation Metrics

Navigation can be recorded as frequency and also triggered metrics, caused by the player and the game environment. This metric should not only record the position and the speed, it should also record the movement modifiers. Depending on the game every movement style, like “still”, “crouch”, “walk”, “run” and so on, should be recorded. If the camera is moveable then their parameters should also be recorded, with that you can identify the spots in the game world where the player is looking at. In most cases the following parameters are captured as navigation metrics:

1. position (x, y, z) and rotation ω of the character,
2. horizontal λ and vertical φ camera rotation and
3. movement modifiers (crouch, walk, run, climb).

These metrics allows calculating heat maps or time base functions to show the different play style of the gamers. In figure 2.6 from Assassin's Creed [25] you can see the tracked navigation data of 138 player within 10 days. You can see that the lower part of the map is not that discovered than the upper left part, this can be an indication for level designers that in this region is too few to discover.

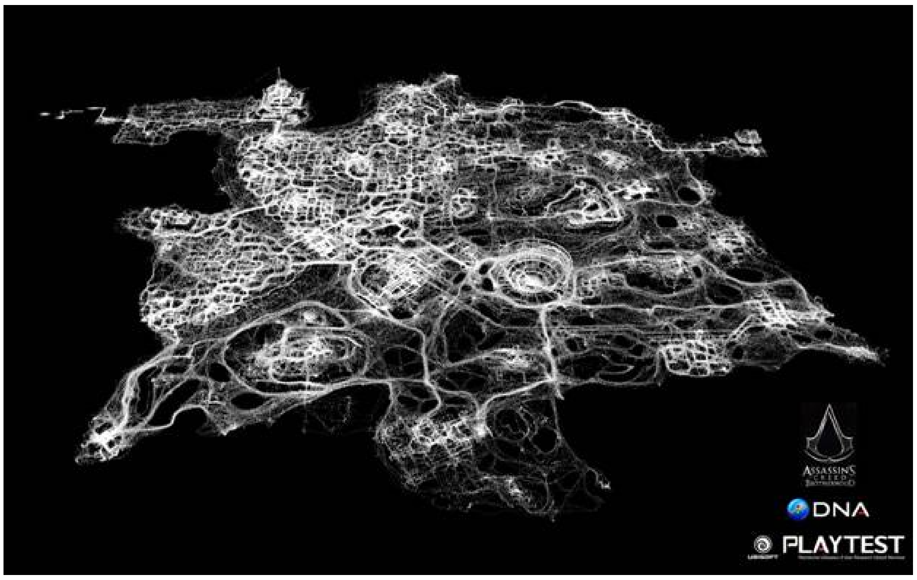


Figure 2.6: Navigation of 138 player in 10 day in Assassins's Creed.

Additionally Veron and Levasseur's [15] four user visiting styles can be calculated described in Chittaro and Ieronutti's article [5]. The ant visitor spends a long time to observe all exhibits, stops frequently and avoids open spaces. The fish visitor prefers open spaces and doesn't look at details. The grasshopper sees only things interesting to it and spends a long time observing it. The butterfly sees almost all things but the time observing it varies. In figure 2.7 the different navigation styles of visitors of a virtual 3D museum [11].

Interaction Metrics

These metrics store the interaction between the player and objects, the game world or with entities. For example firing weapons, picking up objects or in "Open World" games placing or destroying blocks of the world. For this metrics it is useful to save time stamp and position. A designer can also reconstruct the game world to every possible time and have a look how the world changed. In stealth video games like *Hitman* these metrics can tell the designers how the gamers played a level.

Interface Metrics

This category of metrics cover the use of the graphical interface, its menus and basic functionality. For example in Survival games you have a lot of crafting and building, so you get very large lists of items in you menus.

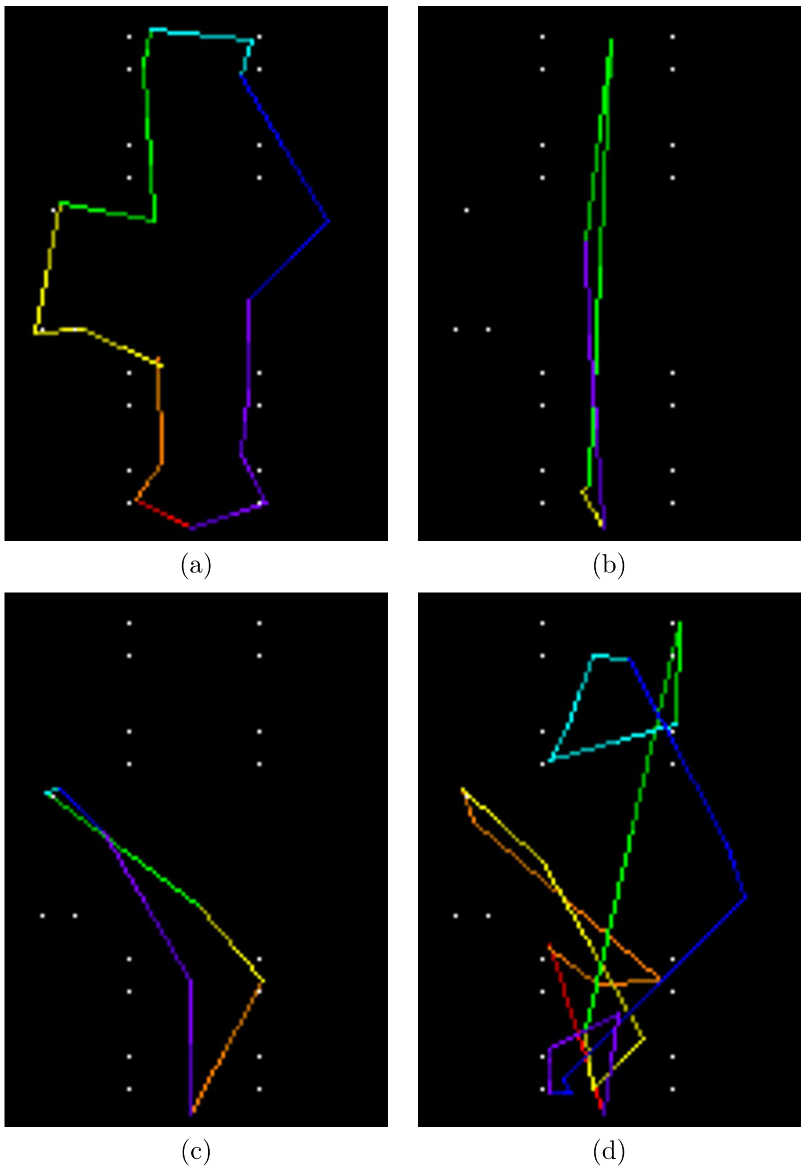


Figure 2.7: Navigation styles ant (a), fish (b), grasshopper (c) and butterfly (d).

Narrative Metrics

Narrative metrics deal with the game story and how the player navigates through it. These metrics store your choices and how you process with a given task, in communication with NPC's. Like in the action RPG *Mass Effect* you have different ways how to complete a mission and how to talk to NPC's.

Chapter 3

Use Cases

Keystroke dynamics of games are hard to compare because the players have a very high degree of freedom how they master a task. This chapter will focus on how the keyboard data can be used to make a decision if a player is experienced or a beginner in playing games .

3.1 Origin of Keystroke Dynamics

Keystroke Dynamics is a topic witch is older than computers, during World War II the rhythm of the operators Morse Code could be used to identify him and distinguish between allies and enemies . This methodology is called “The Fist of the Sender”[3].

In the early 1980’s the principle of “The Fist of the Sender” was uses to develop a user Id and password keystroke dynamics solution.

Security Keystroke Dynamics

The analysis of keystroke dynamics can be classified in static text and dynamic text. The static text analyzes the behavior of a person on a predefined phrase at a certain point in the system. Like when a user is logging in a system with an ID and password. Dynamic analysis does a continuous or periodic monitoring of keystroke behavior. It if first checked when a user logs in to a system and continues after that. For example the user browses the web he frequents certain websites. The typing behavior and a list of websites can be stored and used for identification. Dynamic analysis often needs a training phase to sample the identification data.

Features of a Keystroke

Of a single keystroke a lot of features can be extracted for the analysis, some features can be created by taking a look at multiple keystrokes in a timeline.

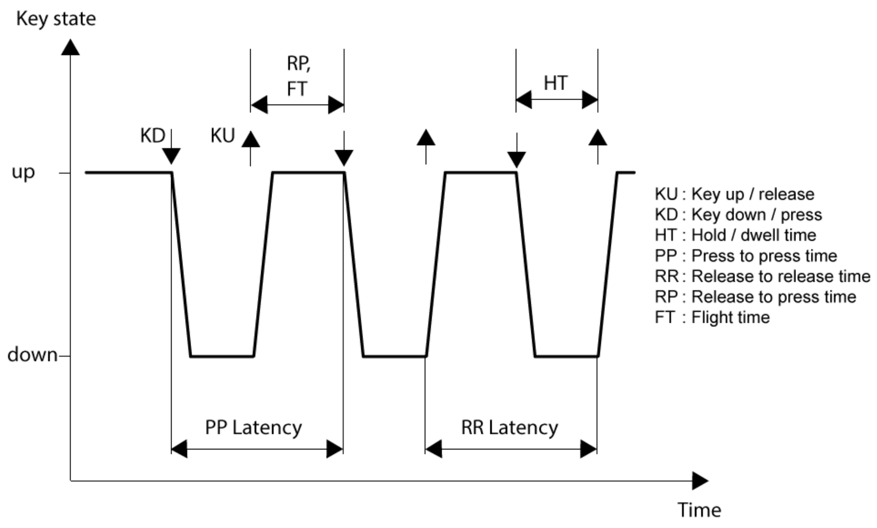


Figure 3.1: Keystroke timing information

In figure 3.1 you can see features that can be extracted out of keystrokes. Every key state change and its time is stored, **KU** marks the key up event and time and **KD** the key down event, the hold time (**HT**) can be calculated immediately. Latency is one of the most commonly used features in Keystroke dynamics. Three types of latencies are defined by Balagani and Phoha [1]: press to press time (**PP**) or Digraph, release to press time (**RP**) and release to release time (**RR**). In 2010 Stefan and Yao [12] called the release to press time flight time (**FT**). Bergadano [2] used Trigraph, the time interval between the presses or releases of alternate keystrokes.

3.2 Keystroke Dynamics in Games

Games are more complicated than login keystrokes, because in a game the player have a higher freedom how they can reach the goal of a game and especially when artificial intelligence is in a game a specific keystroke pattern can not be generated. But a lot of features and mechanics can be used to identify the player's experience. In Single player games the gamer can choose a difficulty level but when it is too hard and may get bored. Another example would be games like *World of Tanks*[21] where a matchmaking system chooses the player who compete at each other. Here the keystroke dynamics can be used to select player that have a similar experience to play in the same match, so that not experienced player don't have to compete in every game against better player. Like Jesse Schell wrote in [10] that keeping the player in the Flow Channel 3.2 is desirable.

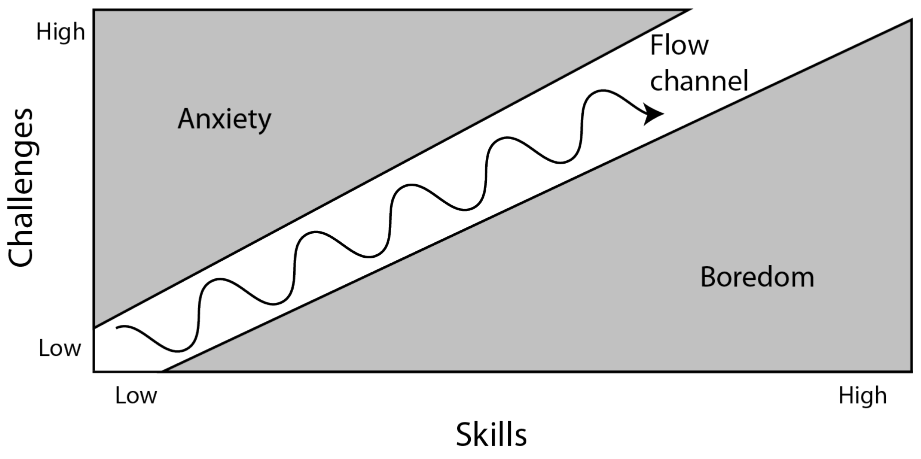


Figure 3.2: Flow Channel

For a publisher it could be interesting to see if someone lets other player test the game. Since the publisher introduced their distribution platforms like *EA's Origin* or *Ubisoft's UPlay* the gamer have to bind their games to an account. That is normally no big deal but with that a gamer gets no chance to sell the games. For selling games many gamer create a new account for only one game to sell the account. This is forbidden in the terms and conditions of the account but there are plenty of accounts on *ebay*. The keystroke dynamics could also be used by the publisher to get an indication when the account is used by a different person.

3.3 Keystroke Dynamics Evaluation

To evaluate the keystroke dynamics of different player to extract their experience with games I will use a feature vector of a players keystrokes and compare it with a distance measurement with pre-evaluated feature vectors. The vector and its detailed data will be described in section 3.3.3.

3.3.1 Test Game

To have a static environment a test game was developed. This game is a spaceship labyrinth where the player has to navigate through, see figure 3.3. *Unity3D* game engine was used to create it. The ship is controlled with a physic body with very low drag values to gain the experience of a realistic behavior. In previous space games, developed in university courses, this type of steering behavior showed that not experienced player had to get used to the steering and did more collisions than experienced gamer. During the development of *OpenMined* and later *SilentSix*, both games are open world space games where the player can craft and mine with the resources in the game world, the steering behavior caused a lot of design issues in the team and by the test user. In this development process we could see that depending on the games that were played different steering methods were preferred. For precise steering and aiming the rotation of the spaceship was captured by the mouse and the keys **A** and **D** were used as strafing. This method was preferred by players who are playing FPS games. Racing games fans liked the steering method where you could strafe and rotate with keys. Others had problems when the spaceship headed downwards to rotate in the right direction.

So I decided for the test game to use the classical racing steering without strafing. The spaceship 3.4 is controlled with the keys **W**, **A**, **S** and **D**, which adds each update cycle a force to the physic body. **W** and **S** are used to steer forward and backward and **A** and **D** are adding a rotating force clockwise or counterclockwise to the spaceship not depending on the direction where it is looking. To simplify the input dynamics I decided to skip the mouse input and only use the keyboard. Because the mouse is a more sensitive device, that produces an input at very little movement. Additionally a mouse device has can be configured in its driver so that it fits every player, which can cause features that not reflect the players experience.

During the game a capture file is produced, like in listing 3.1, which contains the start and end time of the level and collisions between the space ship and the labyrinth boundaries. To identify the player a unique identifier will be created and also stored. Also the keystrokes are stored in this file. Each key change is stored with the key itself, the value of the key and the time stamp with a resolution of 1µs. During testing on different operating systems the minimum frame time in the key capture process was 10µs on *Windows* systems and 100µs on *MacOS* systems. This accuracy will be high enough because the reaction time of keyboard and the human's eye and hand are normally 1ms and more.

```
Keyboard Capture PsychoINC
ParticipantNo|49e9eecd7f752afcfeb4ad536ba2ae4cc661e7a7
CaptureTime|06_08_2014_09:49:16-766647
CaptureStop|06_08_2014_09:50:34-085235
Collisions|14
W|-32768|06.08.2014 09:49:16-885725
W|0|06.08.2014 09:49:17-357041
W|-32768|06.08.2014 09:49:17-756354
D|-32768|06.08.2014 09:49:18-309073
W|0|06.08.2014 09:49:18-458527
S|-32768|06.08.2014 09:49:18-676568
S|0|06.08.2014 09:49:19-327675
W|-32768|06.08.2014 09:49:19-477776
D|0|06.08.2014 09:49:19-511798
```

Listing 3.1: Example of a keystroke dynamics file

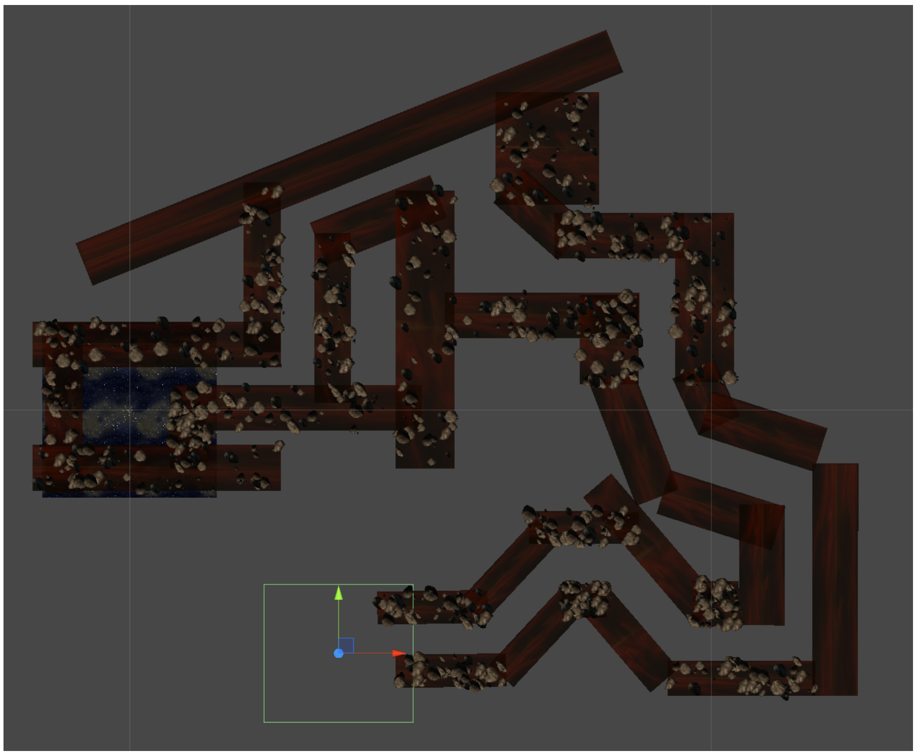


Figure 3.3: Labyrinth in the test game. Start is on the left side and the end of the labyrinth is in the bottom.

The key value is the raw value I get form Unity3D, while writing the capture file it is not important to get a good readable file. This file is used in only to transfer the data to the analysis software, which interprets this values as key up and key down. This would now not be a problem because the keyboard input is digital, but when it comes to analog input like a controller or a joystick the steps between on and off will get important.

The game does not have any type of dynamic obstacles or AI to keep the evaluation process consistent for all participants. The interpretation of random events in the experience will be hard because comparing one player that had to deal with an enemy with another without the enemy can lead to very different keystroke dynamics.

3.3.2 Test Gamer

Twelve people including myself tested the game to evaluate the keystroke dynamics. These people can be sectionalized in 3 main groups by their experience with games, especially games played with keyboard.

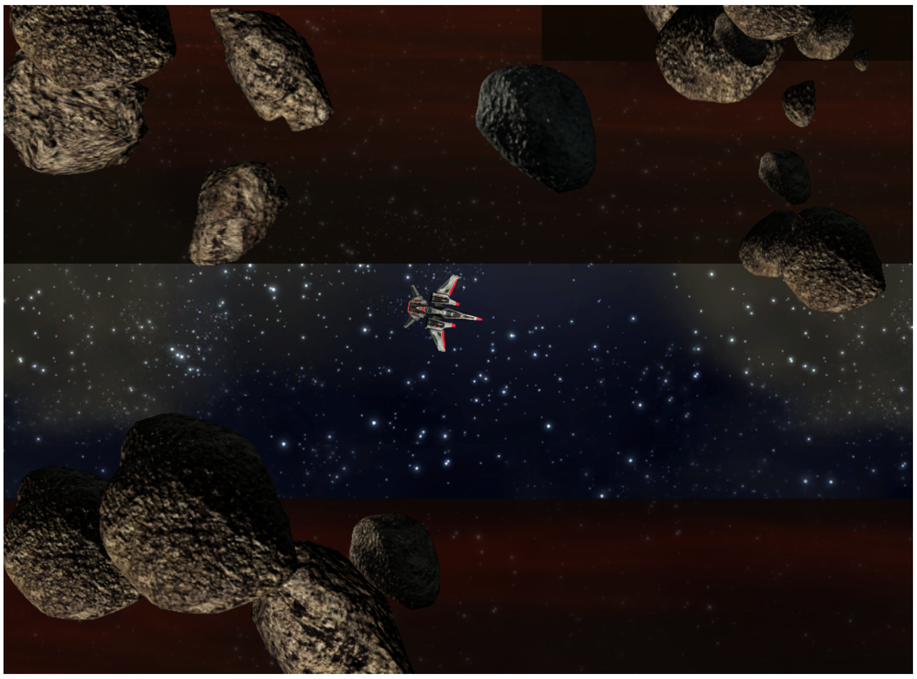


Figure 3.4: Screen shot of the test game with the space ship in the labyrinth.

High experienced player

These players are playing games with keyboard usage on a regular basis several hour per week. Christian B, Christian M and I are playing racing, shooter and RPG, so we are used with using the keyboard in games and have a high awareness how the game behalf's on their commands. Christoph is a game developer and plays different games on consoles. In the following figures this group is represented in red. The participants of this group should get very good scores in the feature selection.

Medium experienced player

This group contains a very broad spectrum of people. Either they play games on consoles and are not used to keyboard input, or playing games not so often, or playing other games like strategy games or puzzle games that are mainly based on mouse input. Medium experienced player have a good sense how their input effect the game, so they can predict when they have to push a button and for how long to get their character in the position they want. In the following figures this group is represented in blue. Testers of this group are Sandra, Patrick, Thomas, Cornelia and Michael M.

Low experienced player

Low experienced player mostly never play games where the input directly effects the character. They are mostly playing build u games or puzzle games on smartphones and tablets. Their sense of how a keyboard input effects the spaceship is not very distinctive. In the following figures this group is represented in green. Testers of this group are Ilse, Elisabeth and Gudrun.

3.3.3 Feature Selection

In games a specific pattern like in security keystroke dynamics can not be generated because a security pattern is learned by the user. In games a millisecond longer pressed button can lead to different situations and require different solutions. That's why a path in the same level of a game is rarely identical when a player makes a second attempt. In an early attempt personally known participants played the test game to identify the features. Every participant played the test game at least three times to get an average value of the keystrokes. A single feature gives only a hint on how good a player is, multiple features will show the experience of a player.

Feature: Difference Rotation

The labyrinth has seven left turns and eight right turns, the difference in keystrokes between the rotation keys should be balanced. This feature should show that more experienced player have a lower difference in the amount of the rotation keys usage, because they can image how long or how often they need to press the key to get the force they need to achieve their desired angle. A less experienced player gets some rotations in the first try and for some he needs more attempts and so the difference value will increase.

In figure 3.5 the difference between the rotation keys is shown in percent of all keystrokes. The difference between the keys in absolute numbers is not meaningful because it depends on how you are pressing the keys to rotate the space ship. For example if you are rotating in one long keystroke or like Thomas in short keystrokes the overall amount will not say anything about your ability. With the absolute value he wouldn't be in the middle section of this table. For that the difference in percent is used for the feature vector. This difference in percent is calculated as:

$$diff = \frac{|(\sum l - \sum r)| * 100,0}{\sum k}. \tag{3.1}$$

where r are the right rotation keystrokes, l are the left rotation keystrokes and k are all keystrokes.

The analysis of the keystrokes shows that the low experienced player have a significant high difference in using their rotation keys, this comes from using the wrong key and then switching to the right one. Especially when

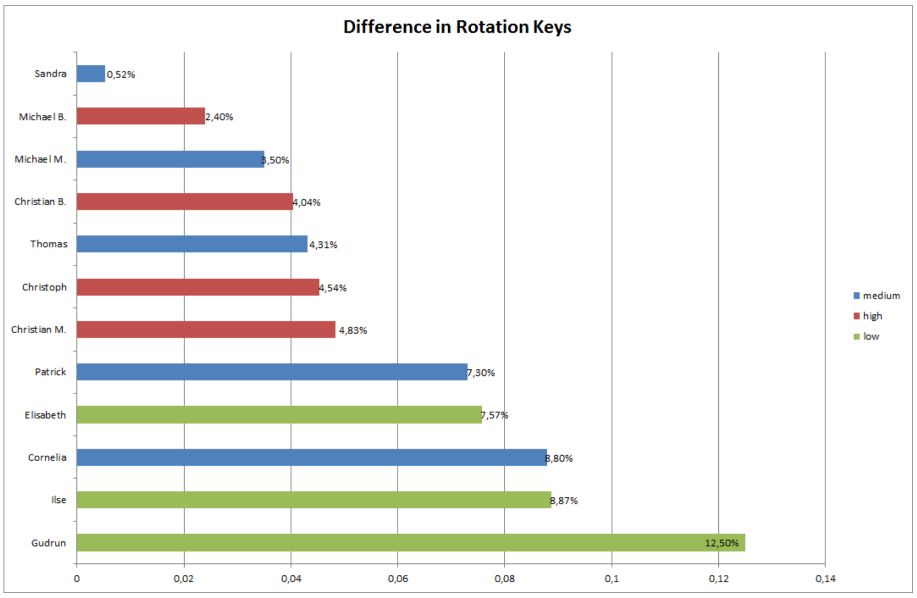


Figure 3.5: Chart that shows the difference of rotation keys in percent to all keystrokes, colored to the predefined experience groups.

they were facing downwards when the position of the key on the keyboard mismatches with direction on the screen. High experienced player have a low difference in keys, except Christian M who had also issues with the downward rotation. Medium players have as expected a low to medium difference.

Feature: Time without Keystroke

The time no key is pressed during gaming, especially in racing and space games is typically very low. In figure 3.6 you can see a chart of key value changes in time. A more experienced player will have a lower time when no key is pressed because he will at least fly straight. During supervising the participants during playing most of the time when no key was pressed was when the participant was having a look on the keyboard to find the key or when the spaceship moved to fast and he waited to slow down. The better player could imagine when they can lift the forward key to reduce the speed before a corner or they used the backward key to break.

In table 3.1 the average session length, the absolute average time no key was pressed and the average of the time no key is pressed divided by the session length is shown. This feature shows the players ability to switch between keys and the overall control of the space ship. For a fast time you have to fly on the limit so the acceleration key is pressed to the last point before reducing speed for a corner.

In the analysis of the data we can see that the high experienced player

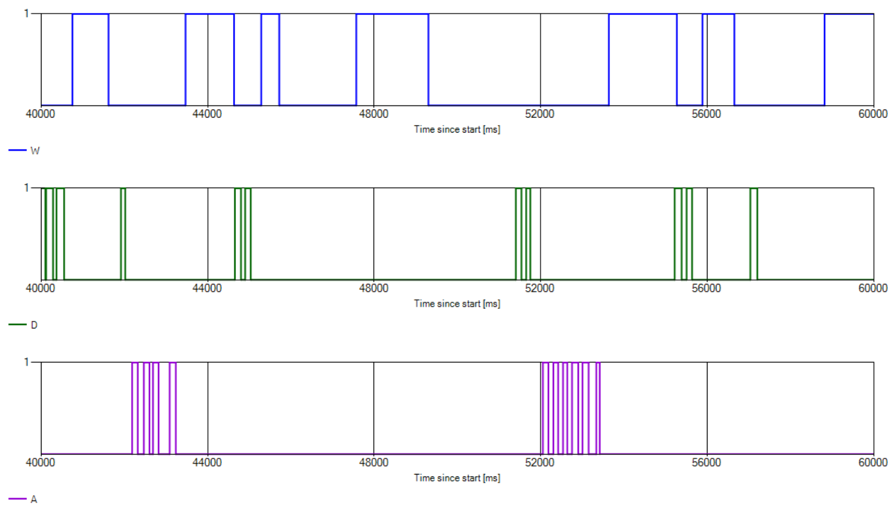


Figure 3.6: Chart of keystrokes in the test game.

Table 3.1: Time no key was pressed.

	Session [s]	*no key time [s]*	*no key time [%]*
Michael B.	66.40	1.374	2.067
Patrick	78.37	2.126	2.679
Christian B.	73.30	2.073	2.751
Sandra	79.67	2.289	2.773
Christian M.	85.23	4.376	4.492
Cornelia	90.65	4.823	4.823
Christoph	79.74	5.793	6.974
Michael M.	80.06	11.453	12.012
Gudrun	116.63	21.671	17.591
Thomas	92.00	20.408	18.438
Elisabeth	109.37	31.071	26.225
Ilse	123.88	50.984	41.093

have the lowest times without pressing keys. As expected the less experienced players have a significant higher time pressing no keys. This time comes from waiting what the ship is doing after they press a key and that the ship is flying too fast for them. The medium experienced players are spread over the table.

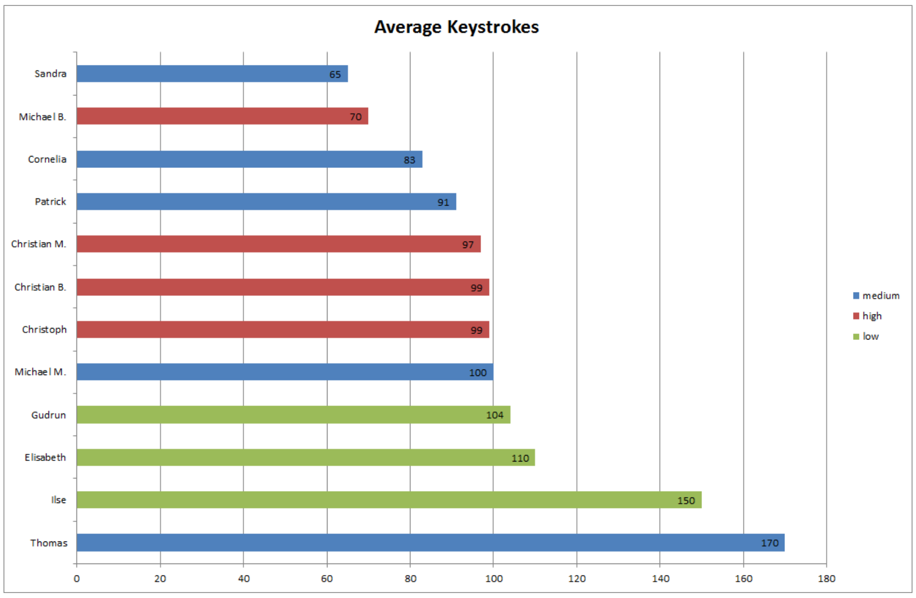


Figure 3.7: Average keystrokes per session.

Feature: Average keystrokes

The number of keystrokes shows how the player controls the space ship. Beginner tend to only tip the rotations keys multiple times, so that the space ship doesn't get a too fast rotation force and doesn't over steer. Also the forward speed can be too high for beginner and by lifting the key they can fly at a slower speed. Especially when they need to reduce speed like before a corner a beginner has troubles identifying the best spot where he should lift the key to get the right speed for the corner. Advanced player have a better feeling for than and so they don't get to slow for a corner and have to accelerate again.

The figure 3.7 shows the tester's average keystrokes over all their sessions. During three test games most players had a significant reduction of the keystrokes by getting used to the space ship controls. The low experienced players couldn't achieve such a high learning curve and their keystrokes stayed high. The analysis of the test data showed that some low experienced player have more than double the amount of keystrokes than the player with the lowest average keystrokes. The distinction between medium and high experienced player is very hard to make with average keystrokes, because they have similar keystroke usage.

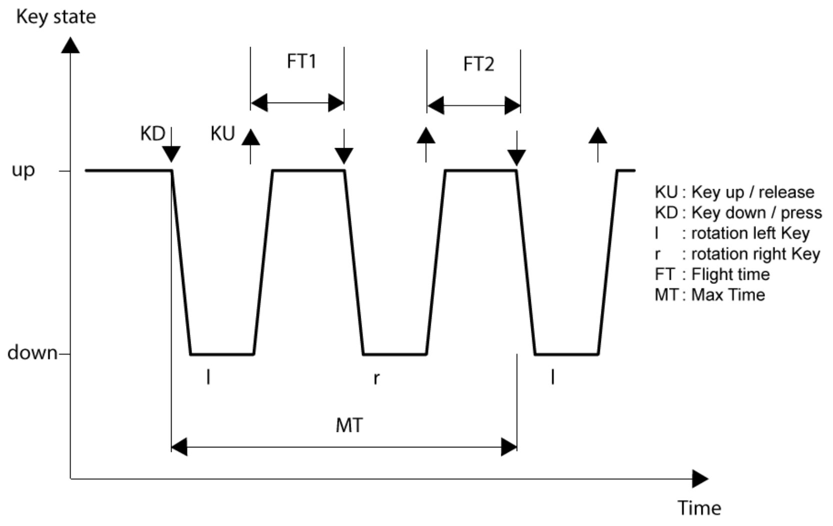


Figure 3.8: Graph shows the parameter of the pause between direction change feature.

Feature: Pause between direction change

This feature should measure how fast the player can switch between the direction keys. For this feature the average flight time and the average minimum flight times are used. Only short changes should be consider, there for a maximum time is used. In figure 3.8 an example of such a feature is shown, it doesn't matter if the key sequence is left, right, left or right, left, right. **FT1** and **FT2** are the two flight times that are used to calculate. **MT** is the maximum time, the time span from the first down event to the third keys down event. This span is set as 1.5 seconds. When a session has not a fast direction change than 5 seconds are used as penalty value.

In figure 3.9 the average change time **aCT** and the average minimum flight time **amCT** are shown. **amCT** is the average of each sessions minimum change time.

The data show that people who are playing video games, which need some sort of fast key changes, on a regular basis perform better than people who are not playing video games or only puzzle and build up games where the keyboard is either not used or the speed of the keystrokes have little impact in the game.

Feature: Simultaneously pressed keys

This feature should show the ability to control the space ship in any situation. Better player should be able to master a corner with accelerating forward and rotate the space ship at the same time. Also on the straight

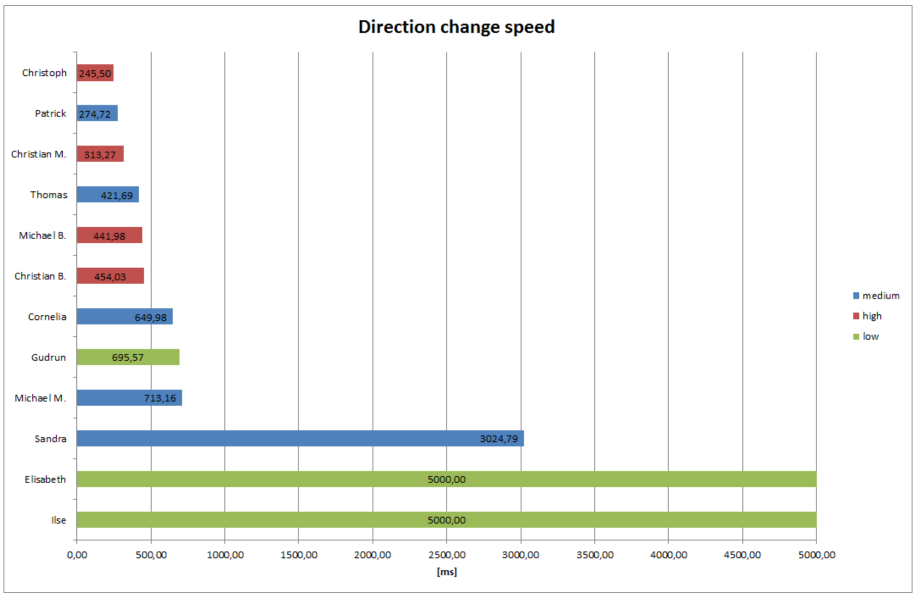


Figure 3.9: Time user needs between two direction keystrokes.

lines of the labyrinth this feature shows who needs to lift the forward key to steer and who can do it simultaneously. With the four available keys operating two keys at the same time is the maximum of simultaneously keys that are working. Either forward and rotation or backward and rotation are combinations that make sense. All other combinations have no effect in the game, so pressing three or all four keys at a time are not considered in the calculation. Console player can have a smaller rate of double pressed keys, because they are used to fly with analog sticks and where two axis are on one button.

In figure 3.10 the average of double pressed keys and the average pressed keys in relation to all pressed keys are shown. The total amount of double pressed keys are not representative because every player had a different amount of keystrokes. So the relation to all keystrokes comes into play. Here the data shows that all high experienced player have over 90 percent of their keys pressed at the same time than another. Medium player reach from 75 percent up to 95 percent. Low experienced player have a significant low rate of double pressed keys.

Additional Feature: Time

This feature is directly connected to the game, but can adapted for other games. It should show that the time a player needs to master the labyrinth is connected to their experience. At the start of the labyrinth the player is in big room and can not see the exit, this can lead to that the player

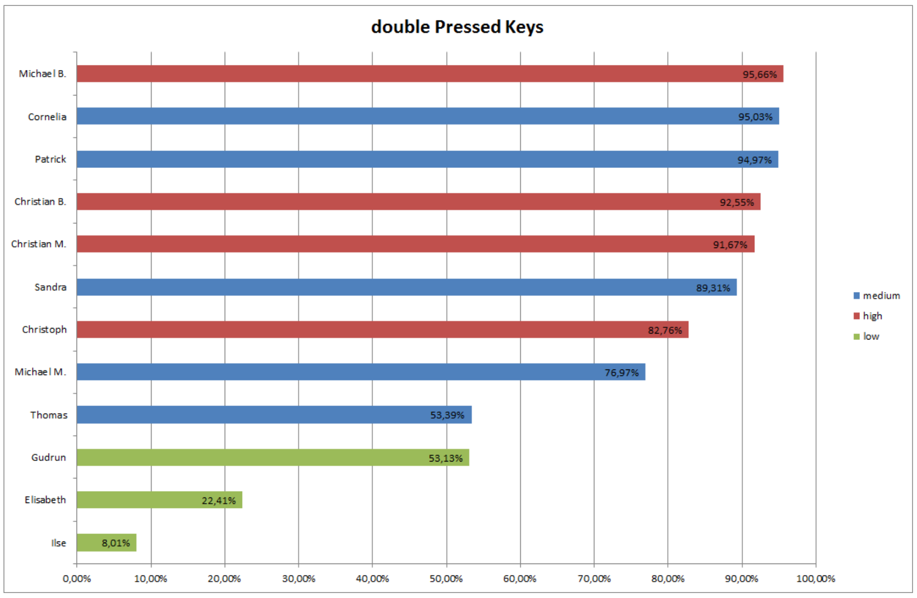


Figure 3.10: Simultaneously pressed keys.

has to orient where to go and this will cost time. With the learning over the attempts this orientation time in the first attempt should be erased by taking the average of all session times. The time feature will show that more experienced player are faster because they have a better awareness of the controls and fly faster through the labyrinth.

In figure 3.11 the average session time of each participant is shown. High experienced player are in the lead, but the medium experienced player have similar times. As expected the low experienced player need significant more time to complete the labyrinth.

Additional Feature: Collisions

The second feature that is directly connected to the game should show how precise a players steering is. High experienced player should have fewer collisions and low experienced player vice versa. A collision with the boundaries will cost speed of the ship and alongside with that time. It should also be an indicator if a player is steering to strong.

In figure 3.12 the average amount of collisions are shown. Here we can see the least experienced player have nearly double the amount of collisions as the more experienced players.

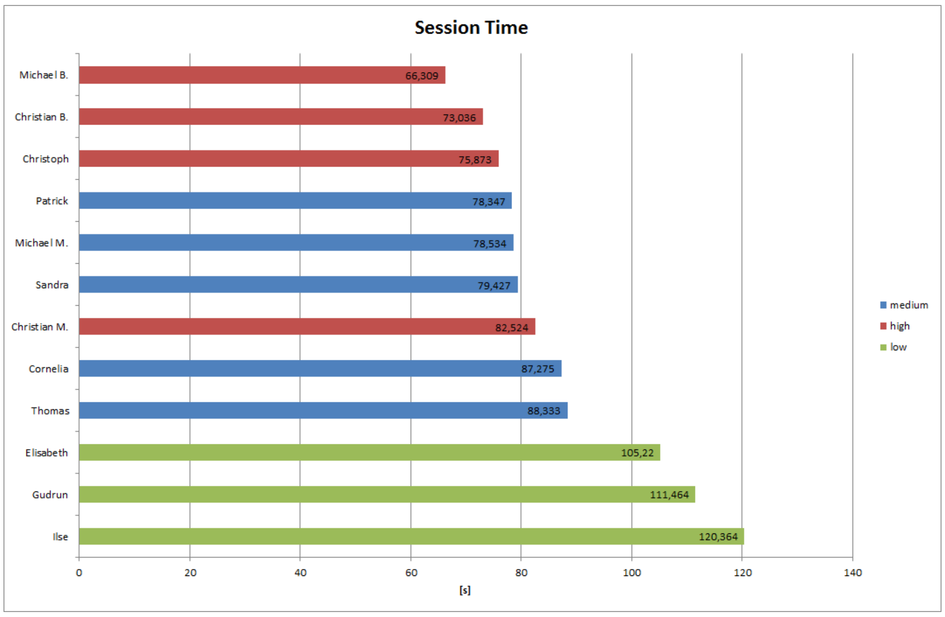


Figure 3.11: Session time.

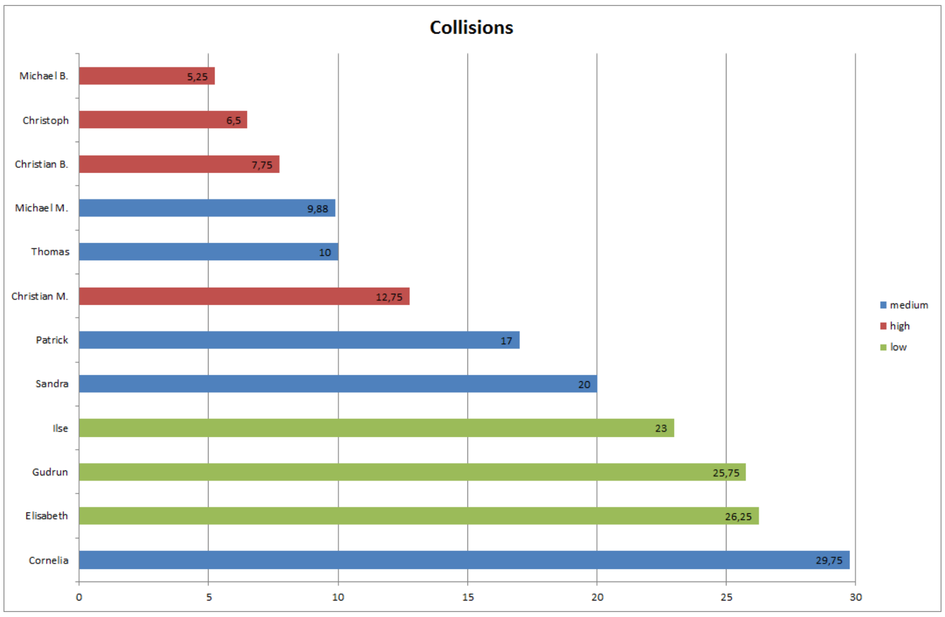


Figure 3.12: Amount of collisions.

3.3.4 Feature Vector

To compute the difference between two players the use of vectors is elementary because the distance calculation is very simple. The feature vector is composed of the seven described features, where each feature is represented as an individual element. The distance between two feature vectors is calculated by:

$$d = \sqrt{\sum_{i=0}^{6} (p_i - q_i)^2}. \tag{3.2}$$

where d is the distance, p is the feature of vector 1 and q is the feature of vector 2.

Normalizing

The elements of the vector have to be in the same range so that every dimension has the same impact on the distance. For example when one dimension has a maximum of 1 and the other one a maximum of 1000, than the higher scaled dimension will shift the distance into its dimension. For that each dimension has to be normalized. That means that every dimension value is between 0 and 1. All values used from percentage values can be used without normalizing because they are already between 0 and 1. Other values like average keystrokes, pause between direction, the time and the collisions have to be normalized. Average keystrokes are normalized by dividing by 200 because the highest value now is 170 and so there will be a reserve for higher keystrokes.

After doing the first clustering some percentage values also had to be improved because their value was too small to make an effect in the distance calculation. The clustering of the groups was mixed up so that two medium player formed each a separate cluster and all others where in one. So the difference between rotation keys has a maximum of 12,5% so it was multiplied by 2 to give it more impact, the same with the no keystroke time.

Clustering

Clustering is the creation of groups (cluster) where each member has a higher similarity to each other than to the members of other cluster [9]. For the automated clustering K-Means algorithm [16] was used, which was invented by Steinhaus [13] and MacQueen [6]. K-Means algorithm is an unsupervised clustering algorithm that classifies the input data points into multiple classes based on their inherent distance from each other. The algorithm assumes that the data features form a vector space and tries to find natural clustering in them. At first three different random vectors, representing the three groups of experience, form the starting cluster. In the next step each vector is compared to the clusters, it is then assigned to the cluster with the least distance. After calculating and assigning all vector the cluster center is computed by the average of all feature elements. The reordering of the vector and recalculating of the clusters is so often done until no vector changes the cluster or a maximum of runs are done. The maximum runs should be so high that the clustering is only influenced when one vector is changing between two clusters because it is exactly in the middle.

The clustering should group the test gamer and later on the player based on their experience, like in section 3.3.2 described. After obliterating the initial normalizing errors the clustering seemed valid. The initial ordering in section 3.3.2 had some minor changes. Because some players have more experience with such games or couldn't get along with the steering.

High experienced cluster: Christian B., Michael B., Patrick, Christian M. and Christoph were assigned to this cluster. Patrick told me in a second interview that he had a good feeling for the steering and played racing games.

Medium experienced cluster: Michael M., Thomas, Sandra, Cornelia and Gudrun were collated to this group. Gudrun was the assigned to this group because some values were much better than the ones of the low experienced group, also in an additional interview she told me that she played years ago racing games on consoles, where she was not good but it could have an impact on how she handles games.

Low experienced cluster: Ilse and Elisabeth were assigned to this cluster as expected.

The clustering is based on a strict mathematical model, where personal perceive can be different. The changes were not surprisingly high and the distances to the surrounding cluster are plausible.

Chapter 4

Evaluation

This chapter is about testing the features of keystroke dynamics and metrics and verifying that distance measurement is an appropriate tool to identify the player's experience. The results are discussed in the next chapter.

The goal of this work is to determine whether keystroke dynamics and metrics can be used to identify a players experience with video games. Keystroke dynamics were designed to identify a person by his keyboard usage. Metrics are used to identify problems with games and level design. These two techniques are merged to get a value for the player's experience. Most of the tests were done during the GameStage Expo[1] in Linz, the others were done on the participants PC. The survey was done fully anonymous because the calculations didn't need personal information. All in all 90 participants played the game and were interviewed to identify their experience with games.

The probands had to player the test game, as described in section 3.3.1, where they had to fly with a space ship through a labyrinth.

4.1 Apparatus

The testing during the GameStage was done on a PC built for playing computer games, with an *Intel* i5 3570K, 16GB DDR3 RAM and 2 *NVIDIA* GTX 570 each with 1270MB RAM on a 22 inch monitor with 1920x1080 pixels. The vertical synchronization was activated and the frames per second drop below 60. For the input a standard Keyboard was used.

To get more variety of participants the testing was extended so that people could play the game at their personal PC or MAC. Since *UNITY*[2] has very low hardware requirements and the game doesn't use hardware intense features no tester had issues to run the game.

[1] http://lorti.github.io/gsexpo/

[2] http://unity3d.com/unity/system-requirements

4.2 Procedure

The study consisted of 3 blocks: the introductory block, playing block and the finishing block with a short interview and evaluating the playing.

First, in the introductory block, the participants were welcomed and given an overview about the project. Also the game itself, what they have to do and how they can interact with the game was discussed with the player.

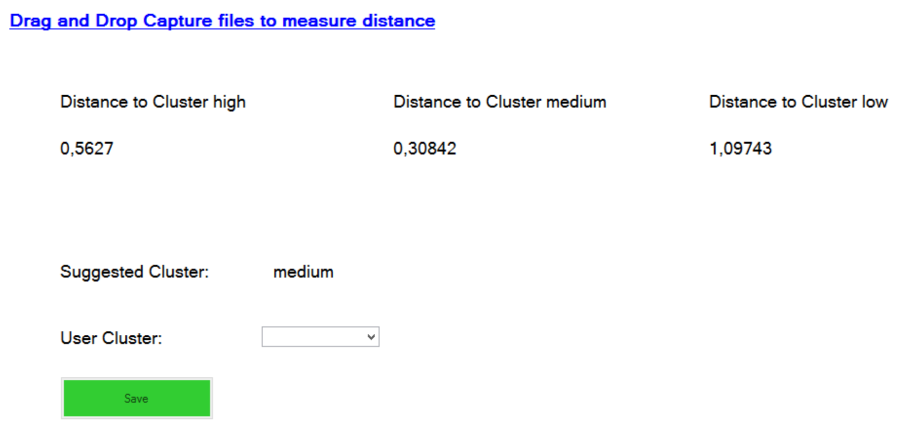
The playing block started with showing them the objects in the game world and to get a clearer view for the game. After that the participants could arrange the keyboard to feel comfortable and stated the game. During the gaming session the participants could ask additional questions about the game, here I could observe that participants that are playing video games very often had no problems with focusing on the game and in the same time talking to people. On the GameStage Expo mostly groups of friends or families were doing the tests and a rivalry came up to be the best of the group. When there was time I let them do the test a second time but this data was not stored in the database, so that every player has only one entry of dataset.

After playing the game the finishing block started, where I had a talk with the players to identify their experience with video games. In this talk we discussed how often he plays, what type of games, on which platform they are playing and how they would rate them self. After that the Capture file was analyzed by the MetricViewer software and the result was compared to the result of the talk. In the cases where the two results didn't meet each other I asked additional questions to find out why this could be the case. After that the study was over and some participants wanted to know more why and how did I do the calculation and some open discussions about this topic occurred.

4.3 Data Analysis

When a Capture file was imported to the software by drag and drop, the software automatically created a database entry with the keystrokes. Out of the keystrokes a feature vector was created and the distance was measured to the three clusters. In figure 4.1 an example of the result of the distance calculation is shown, the distances to the three cluster are shown and also the from the distance measurement suggested cluster is shown. The cluster from the talk can be selected from the list. After all the suggested and selected list and the feature vector get stored in the database.

Additionally to the single file analysis the MetricViewer software provides a tab where all evaluation data is captured to see an overview of all tests. This tab shows like in figure 4.2 all data that is stored in the database via the tests.


Drag and Drop Capture files to measure distance

Distance to Cluster high	Distance to Cluster medium	Distance to Cluster low
0,5627	0,30842	1,09743

Suggested Cluster: medium

User Cluster:

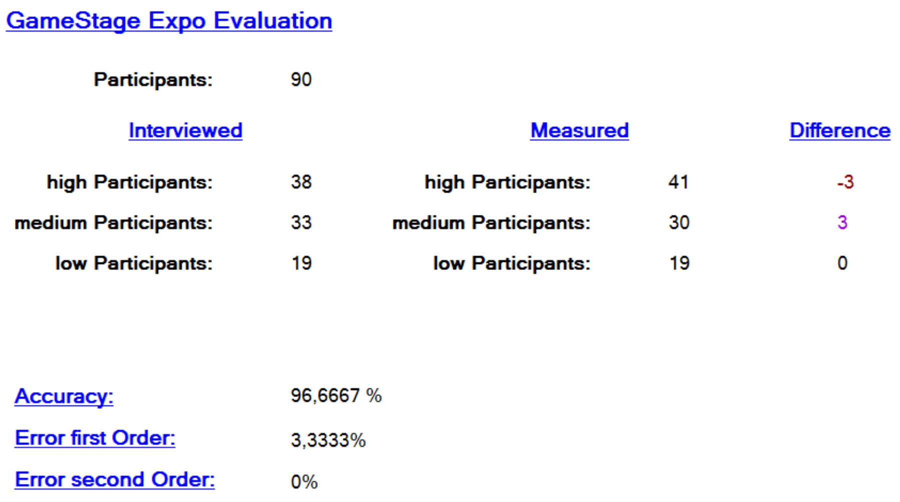
Figure 4.1: Screen shot of the MetricViewer software with an example of a distance calculation of a test person.

To identify issues with the calculations two errors are calculated, the first one is the error first order, which counts all wrong calculations regardless to what the error was. The error second order only counts wrong calculations, where the interviewed experience level and the measured level are more than on level apart. These errors are major problems and should show that the calculations have flaws.

4.4 Study Results

The MetricViewer software gives also a feedback to all participants by showing, like in figure 4.2, the number of participants, the breakdown of interviewed results by their experience, the breakdown of measured results by their experience, the difference between interviewed and measured results and the accuracy of the distance measurements. During all the test sessions 90 people participated in this study, with a range from 13 to 60+ years. Both female and male participants accomplished the tasks. As expected the age and the gender of the participants didn't show a tendency to an experience cluster. The only exception are *Digital natives* during the study it was hard to find people under 25 years who had no or very few experience with video games. *Digital natives* that told me they are very bad had at least medium experience because they are playing games on their smartphones and browser games, so they know some mechanics in video games, additionally they are used to working with a keyboard and had no issues with the input.

In all 90 tests of measuring the players experience with games only three participants had a different result than they where categorized by the interview and their self-assessment. The three false values were all interviewed as



GameStage Expo Evaluation

Participants: 90

Interviewed		Measured		Difference
high Participants:	38	high Participants:	41	-3
medium Participants:	33	medium Participants:	30	3
low Participants:	19	low Participants:	19	0

Accuracy:	96,6667 %
Error first Order:	3,3333%
Error second Order:	0%

Figure 4.2: Screen shot of the MetricViewer software with an example of the evaluation of all participants. Measured and interviewed results contrast with each other and the difference of both. Also the accuracy rate is calculated.

medium experience player but the data of the game calculated them as high experienced player, this is a normal false calculation but a miscalculation where a low experienced player is measured as high experienced player would be a sign that the calculation can have issues or the features are not correct. The accuracy is the value of correct measurements compared to the total of measurements, like in equation 4.1. The accuracy of the study is 96,66% or an error rate of 3,33% which is more accurate than expected. Due to it is not a measurement for security operations that error rate has a low impact in the gamers experience. All three participants that were categorized wrong were underestimated in the interview, their play style showed that they are better than they thought.

$$accuracy[\%] = \frac{correctMeasures * 100}{totalMeasures}. \tag{4.1}$$

Especially the calculations of the low experienced player are all correct which is a very important result because in single player games the low experienced player need help to choose the right difficulty for them. More experienced player have a sense to estimate the proper difficulty setting for them.

Chapter 5

Discussion and Future Work

This thesis shows methods that can be used to measure a players experience with video games at a very early stage. Also some visualization techniques like heatmaps and user orientated development methods are described.

The main goal was to use mostly keystroke dynamics and metrics to identify a players experience which can be used in a single player campaign to give the gamer a feedback which difficulty level he should choose. Especially when some extreme differences between the selected difficulty level and the players experience occur, the game can give feedback that with this constellation the desired enjoyment will fail.

5.1 Improvements

In general the keystroke dynamics and metrics work fine but there are some things that can improve the measurements. Like the feature session time 3.3.3 is only normalized on the upper boundary, to get values between 0 and 1, 50 seconds should be subtracted. The fastest players needed 60 seconds so with a reduction of 50 there is enough space when a faster time is to evaluate.

In section test game 3.3.1 the controls and problems with flying downwards was described, this would be a great feature for the identification of the experience but it was not track able from the keystrokes alone. For this more information of the game, especially the orientation of the track segment, would be needed. A metric created in the game would fit perfectly for it.

Currently the keystroke dynamics and metrics are stored in text files, manually added the database and the evaluation only starts when a file is dragged in the MetricViewer software. This should be fully automated and the user of the MetricViewer should get a window with the new added data.

5.2 Discussion

The shown metrics and keystroke dynamics to identify a players experience are very strong connected to the game, in this case a 2D space game with only keyboard usage. That means that adaptations have to be made for other types of games. In this thesis a simple game was developed to have an overview over the features and the more complex a game gets the more complex are the features and calculation errors are harder to identify. For example when you extend the game with a mouse for rotation you get two axes to control the space ship, but the raw mouse input gives you no hint what the player is doing, here a stronger connection to the game is needed.

The clustering with K'means algorithm can accomplish as much cluster you want, but for a satisfying result enough samples or participant data should be available. The more samples you have for clustering the better the cluster center are calculated. 12 participants was more or less the minimum of participants to define three clusters. As seen in section 3.3.4 some testers were switched into different cluster, this could lead to problems where too few samples are in one cluster or a cluster gets no samples. To achieve a good clustering enough samples should be available or in situations where empty or too few samples the amount of tester should be increased.

This measurement method with defined cluster of preselected participants is very strong coupled to their experience level. The predefined groups, like in section 3.3.2 described, form the cluster of the measurement. In this work the low experienced group consists of people that are normally not playing video games, except puzzle games where the time is not important. The selection of the right participants is very important, for example when I want to produce a game what is very hard to pass and my target audience is very experienced player I have to select participants that are known to games. Player with no experience can influence the groups and the measurement can be shifted. In the study it was very hard to find low experienced player, because nowadays it is hard to find people that have no or hardly any experience with video games. The more preselected participants you have the more groups can be made.

For verifying that it is possible to measure a players experience 90 people played the game and their keystrokes were measured to verify the calculations. Of this 90 measurements only three were wrong, which is a first order error rate of $3,33\%$, which is a very good value. More important is that no errors second order were measured, which means that no player with low experience had a high rating.

Two of the three false measures were player that played a lot as teenager but for the last 10 to 15 years they didn't play that much. In the interview a medium experience was set but the evaluation of their keystroke dynamics showed that they are in the high group, they told me that when they played they did a lot of 2D racing games and from their experience they were pretty good. So it can either be that playing video games and its mechanics will not be lost over the years and can be fetched from the memories or the calculation is wrong. Due to lack of long term studies this can not be verified.

All in all keystroke dynamics are a good use to measure a player's experience at an early stage, with a very fast calculation time. It can be used in several fields to improve the player's enjoyment. Like in multi player games with a matchmaking system, in a small tutorial where the player can adjust all options of a game the keystroke dynamics are created. When the player than starts a match these dynamics can be used to find a server where the other player have a similar experience. After some matches the game statistics and metrics can be used to recalculate the player. In single player games there are multiple ways to use keystroke dynamics. First it can be used in games where the difficulty levels are permanent to give the player a feedback when he chooses a level which is too different from his experience. As in section 1.2 the game *Call of Duty 4: Modern Warfare*[20] uses a trainings level so that the player gets used to the controls and weapon handling, after finishing this level the game suggests a difficulty level based on the time the user needed. In this example keystroke dynamics and metrics can be used for more detailed analysis.

5.3 Future Work

This study was designed as an initial study to find out, if the idea of providing additional detail is beneficial for measuring a players experience with keystroke dynamics and metrics. Furthermore, study results were expected to point out areas requiring further investigation.

In the future the system of keystroke dynamics and metrics should work on different games, but unfortunately every game is different and for that new features have to be identified. Especially when the game uses other input devices the features has to be adapted.

For an easier use the recordings and calculations should be done in the game directly, as described in section 5.1. It can be used in finished games and also in the production stadium, where tester can have an additionally survey to confirm the calculation results.

The initial study only showed for one type of game that calculations of a players experience can be done. For extending the idea of keystroke dynamics and metrics for such a use with new features, the calculations have to be verified that they meet the intended goals.

Appendix A

CD Content

Format: DVD-ROM, Single Layer, ISO9660 + Joliet-Format

A.1 Thesis

Path: /

Thesis_Michael_Bauer_2014.pdf	Master's thesis as PDF file.

A.2 MetricViewer (C#)

Path: /MetricViewer

ReadMe.txt	Instructions to use the project.
/src	Source code of the MetricViewer
/build	Executable of the MetricViewer

A.3 Mongo DB

Path: /MongoDB

/dump	Dump file of the database

A.4 Capture Files

Path: /CaptureFiles

/Preselected	Capture files of the 12 preselected game testers
/Evaluation	Capture files created during the study

A.5 Test Game

Path: /TestGame

/Preselected	Test game for the preselected tester
/Preselected/Build	Executable for MacOSX and Windows
/Preselected/Master thesis Game	Unity3D project source code (Version 4.3.2f1)
/Evaluation	Test game for the evaluation
/Evaluation/Build	Executable for MacOSX and Windows
/Evaluation/Master thesis Game - Eval	Unity3D project source code (Version 4.3.2f1)

A.6 Images

Path: /Images

2_1_Chart.png	Percentage completing campaign
2_1_HeatMap.jpg	Heatmap in Counter-Strike
2_1_PathMap.jpg	Trajectory in Assassins Creed
2_1_Profiler.png	Screenshot Unity3D profiler
2_4_PlayerTypes.png	Player Types
2_5_NavAnt.png	Navigation style ant
2_5_NavButterfly.png	Navigation style butterfly
2_5_NavFish.png	Navigation style fish
2_5_NavGrashopper.png	Navigation style grashopper
3_1_Keystroke_Timing.png	Graph keystrokes
3_2_FlowChannel.png	Flow channel
3_3_1_AverageKeystrokeChart.png	Evaluation average keystroke
3_3_1_CollisionChart.png	Evaluation collisions
3_3_1_DifferenceRotChart.png	Evaluation difference rotation
3_3_1_DirectionChangeSpeedChart.png	Evaluation direction change speed
3_3_1_DoublePressedChart.png	Evaluation double pressed keys
3_3_1_SessionTimeChart.png	Evaluation session time
3_3_GameWorld.png	Game world
3_3_KeyChangeSpeed.png	Graph keychange speed
3_3_KeyTimeChart.png	keys in time
3_3_TestGame.png	Test game
4_3_DistanceCalc.png	Evaluation screenshot

4_3_GSEval.png	Evaluation of the study

A.7 Online Literature

Every folder consists of a PDF and a URL to the Website

Path: /Literature

/Bartle	Richard Bartle - Who plays MUAs?
/BFTelemetry	Benjaminsson Kristoffer - Presentation of Telemetry data in Battlefield Heros
/Cooper	Allen Cooper - The origin of personas
/AC3DNA	Jonathan Dankoff - Game telemetry with DNA
/DrachenMetrics	Anders Drachen - What are game metrics?
/PlayerTypes	Kyatric - Bartle's Taxonomy of player types
/StayingPower	Bruce Phillips - Staying Power: Rethinking feedback to keep players in game
/EANumbers	Dean Takahashi - Goofy EA numbers that show how deeply players are engaged in games
/Valve	Valve - The science of CSGO Heat Maps

References

Literature

[1] Kiran S. Balagani et al. “On the discriminability of keystroke feature vectors used in fixed text keystroke authentication”. In: *Pattern Recognition Letters* 32.7 (May 2011), pp. 1070–1080 (cit. on p. 17).

[2] Francesco Bergadano, Daniele Gunetti, and Claudia Picardi. “User Authentication through Keystroke Dynamics 1”. In: *ACM Transactions on Information and System Security (TISSEC)* 5.4 (2002), pp. 367–397 (cit. on p. 17).

[3] Biopassword. “Authentication Solutions Through Keystroke Dynamics”. 2008 (cit. on p. 16).

[4] Mark Graham Brown. *Keeping Score: Using the Right Metrics to Drive World-Class Performance*. Productivity Press Inc., 1996, p. 224 (cit. on p. 4).

[5] Luca Chittaro and Lucio Ieronutti. “A visual tool for tracing users’ behavior in Virtual Environments”. In: *Proceedings of the working conference on Advanced visual interfaces - AVI ’04*. New York, New York, USA: ACM Press, 2004, p. 40 (cit. on p. 14).

[6] J.B. MacQueen. “Some Methods for classification and Analysis of Multivariante Observations”. In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*. 1st ed. Berkley: University of California Press, 1967, pp. 281–297 (cit. on p. 30).

[7] Larry Mellon. *Applying metrics driven development to MMO costs and risks*. Tech. rep. 650. Redwood City: Versant Corporation, 2009, pp. 1–9 (cit. on pp. 4, 6).

[8] Magy Seif El-nasr, Anders Drachen, and Alessandro Canossa. *Game Analytics*. Ed. by Magy Seif El-Nasr, Anders Drachen, and Alessandro Canossa. London: Springer London, 2013 (cit. on pp. 4, 7).

[9] Charles Romesburg. *Cluster Analysis for Researchers*. Lulu Press, North Carolina, 2004, p. 340 (cit. on p. 30).

[10] Jesse Schell. *The Art of Game Design: A book of lenses*. CRC Press, 2008, p. 520 (cit. on p. 17).

[11] Kingkarn Sookhanaphibarn and Ruck Thawonmas. “Visualization and Analysis of Visiting Styles in 3D Virtual Museums”. In: *Digital Humanities 2010*. Intelligent Computer Entertainment. London: King’s College London, 2010, pp. 1–6 (cit. on p. 14).

[12] D Stefan and Danfeng Yao. “Keystroke-dynamics authentication against synthetic forgeries”. Chicago, IL, 2010 (cit. on p. 17).

[13] Hugo Steinhaus. “Sur la division des corps matériels en parties.” In: *Bull. Acad. Polon. Sci* 1 (1957), pp. 801–804 (cit. on p. 30).

[14] Anders Tychsen and Alessandro Canossa. “Defining personas in games using metrics”. In: *Proceedings of the 2008 Conference on Future Play Research, Play, Share - Future Play ’08* (2008), pp. 73–80 (cit. on p. 13).

[15] Eliseo Veron and Martine Levasseur. *Ethnographie de l’exposition: l’espace, le corps et le sens*. Centre Georges Pompidou, Biblioth{\‘e}que publique d’information, 1983, p. 220 (cit. on p. 14).

[16] J. A. Hartigan Wong and M. A. “Algorithm AS 136: A K-Means Clustering Algorithm”. In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 28.1 (1979), pp. 100–108 (cit. on p. 30).

Films and audio-visual media

[17] Digital Illusions CE. *Battlefield 3*. PC-Game. 2011 (cit. on p. 2).

[18] id Software. *Quacke*. PC-Game. 1996 (cit. on p. 4).

[19] Hidden Path Entertainment Valve. *Counter-Strike: Global Offensive*. PC-Game. 2012 (cit. on p. 9).

[20] Infinity Ward. *Call of Duty 4: Modern Warfare*. PC-Game. 2007 (cit. on pp. 2, 38).

[21] Wargaming.net. *World of Tanks*. PC-Game. 2010 (cit. on pp. 2, 17).

Online sources

[22] Richard Bartle. *Who plays MUAs?* 1985. URL: http://mud.co.uk/richard/wpm.htm (cit. on p. 12).

[23] Benjaminsson Kristoffer. *How data rules the world: Telemetry in Battlefield Heroes*. 2010. URL: http://dice.se/publications/how-data-rules-the-world-telemetry-in-battlefield-heroes/ (cit. on p. 1).

[24] Allen Cooper. *The origin of personas*. 2008. URL: http://www.cooper.com/journal/2008/05/the_origin_of_personas#more-286 (cit. on p. 11).

[25] Jonathan Dankoff. *Game Telemetry with DNA Tracking on Assassin's Creed*. 2014. URL: http://www.gamasutra.com/blogs/JonathanDankoff/20140320/213624/Game%5C_Telemetry%5C_with%5C_DNA%5C_Tracking%5C_on%5C_Assassins%5C_Creed.php (cit. on pp. 8, 11, 13).

[26] Anders Drachen. *What are game metrics?* 2012. URL: http://blog.gameanalytics.com/blog/what-are-game-metrics.html (cit. on p. 6).

[27] Kyatric. *Bartle's Taxonomy of Player Types (And Why It Doesn't Apply to Everything)*. 2013. URL: http://gamedevelopment.tutsplus.com/articles/bartles-taxonomy-of-player-types-and-why-it-doesnt-apply-to-everything--gamedev-4173 (cit. on p. 13).

[28] Bruce Phillips. *Staying Power: Rethinking Feedback to Keep Players in the Game*. 2009. URL: http://www.gamasutra.com/view/feature/4171/staying%5C_power%5C_rethinking%5C_feedback%5C_%7B%7D.php (cit. on p. 9).

[29] Dean Takahashi. *Goofy EA numbers that show how deeply players are engaged in games*. 2013. URL: http://venturebeat.com/2013/02/26/goofy-ea-numbers-that-show-how-deeply-players-are-engaged-in-games/ (cit. on p. 2).

[30] Valve. *The Science of CS:Go Heat Maps*. 2012. URL: http://blog.counter-strike.net/science/maps.html (cit. on pp. 9, 10).

Messbox zur Druckkontrolle

— Druckgröße kontrollieren! —

Breite = 100 mm
Höhe = 50 mm

— Diese Seite nach dem Druck entfernen! —