Latency Incorporated Game Mechanics

- a game design project

mea111036 - Tom Jensen

Masters Thesis - 2011 - Medialogy - Aalborg University

Tom Jensen

AALBORG UNIVERSITY

Department of Architecture, Design and Media Technology

Medialogy, 10th Semester

Title:
Latency Incorporated Game Mechanics

Project Period:
February 8th - May 31st 2011

Semester Theme:
Masters Thesis

Supervisors:
Martin Kraus

Projectgroup no.:
mea111036

Members:
Tom Jensen

Abstract:

This project proposes a novel method for achieving fairness for players of network games with high latencies.

Studies have shown that a player's performance falls as latency increases to the game server. Studies also show that the first person shooter genre is highly sensitive to this phenomenon.

Three hypotheses are constructed, based on results of other studies. These hypotheses seek to boost a player's characteristics as a function of the player's latency. Hence, Latency Incorporated Game Mechanics.

In order to evaluate these hypotheses a game is designed, as well as specific features and procedures needed in the evaluation process.

The game is then realized through a working prototype. This prototype is then used in an experiment that ultimately is inconclusive in its evaluation of the hypotheses, but does bring valuable information for future work.

Finally the project as a whole is evaluated. It is determined that while there is great need for future work on the project's main focus, a solid base of knowledge and prototype equipment has been accumulated for the next phases.

1. Preface

1.1. REFERENCING

When referencing an external piece of documentation it will be referred to via bracketed parentheses containing a number which can be found in the appendix chapter where a description of the external resource is listed.

1.2. INTERNAL REFERENCING

When referring to another chapter or section of the documentation it should appear as the section number followed by the headline of the appropriate piece of text referenced to.

Referring to images, tables or other charts will be referred to in the text as "figure" followed by the appropriate figure number.

1.3. APPENDIX

The appendix will contain a listing of the used references within the documentation as well as any other documents that is of some relevance to the main focus of the project. Relevant but space consuming things such as full data read out may instead be found on the DVD in their full length.

1.4. DVD

Accompanying this written documentation is a DVD containing files relevant to this project.

Videos

On the DVD you will find several videos. One of them is a audio visual production presenting the project as a whole. You will also find a two part 'making of' series of videos that highlight different aspects of the practical production. I recommend viewing at least the AV production before reading the documentation in order to get a quick overview of the project as a whole.

Source files

Most source files for the creation of the practical implementation is included on the DVD. Some files are not included such as Photoshop .psd's and 3ds Max .max files. The reason why they are not included is that they are work files that are not necessarily cleaned up and take up a lot of space. Their output, .tga and .fbx, for texture maps and models/animations respectively, are included in the source files for the project.

Sources and other files

Sources referred to in this documentation are included, if possible. In cases of web sites, a .pdf version of the web site of the state referred to is included as well.

Furthermore, other files relevant to the project, such as test data, process and development images as well as all images used in the documentation can be found on the DVD.

Contents

2. Introduction

The online experience of video games is becoming an increasingly important part of the product, even from a business perspective [8]. Some games have only an online component, completely forgoing a single player mode over competitive or cooperative human to human experiences.
A testament to the importance of the online experience are the communities build around games. Video casts of games, with commentators like a real life professional soccer match. Terms such as ranking, rating, achievements, prestige and leader boards have become common knowledge in the gaming communities, proving that player performance and skill is something that is becoming a priority for the players. No longer are video games restricted to being a way to relax and escape the troubles of everyday life - it has become a place where the gamer need to excel and perform too.

A player would expect to have a fair chance by being equal with other players at the start of a game [8]. However, playing with people from all over the world, all are not equal even at the start of a game.
Due to an array of factors, the most important of which is perhaps distance, there is a delay from the time a user carries out an action, to the time the server acknowledges and updates accordingly. In this window of time, another player who is located geographically closer to the server will be able to have his actions carried out faster than people further away, thus giving them an advantage.

It has been shown in numerous studies [1-3] [5] [7-10] that as this information delay, latency, increases, player performance drops.
The rate at which the performance falls is dependent on a number of factors, such as the genre and perspective of the game as described in [5] [10].

Solving some of this unfairness lies in having faster connections, but ultimately the sheer distances the signal has to travel and the restriction that is the speed of light, ensures that there will always be a delay between player action and server side reaction.
Even at the speed of light it takes approximately 0.13 seconds to travel around the world [9].

Thus mending this issue lies in methods of ensuring fairness through design of the games and network architectures.

> *"Latency determines not only how players experience online gameplay, but also how to design the games to mitigate its effects and meet player expectations."[5]*

This project introduces a novel method of tackling the fairness problem related to latency.

3. Latency Incorporated Game Mechanics

In this section I will describe three hypotheses that will seek to bring some fairness to those players documented to be at a disadvantage.
The hypotheses are based on, or inspired by studies in the field of user perception and performance in relation to network conditions.
Some of the hypotheses will later on be black boxed for the scope of this project, but they are stated and their implications explained here to put the idea into its full context and perspective.

Following the statement of each hypothesis there will be a short presentation of similar research or highlighting the problem area in order to uncover guidelines and draw upon knowledge useful for the design of a test platform for these hypotheses.

In their current form the hypotheses are aimed mainly towards the "first person shooter", FPS, genre, but if successful should be capable of being generalized.
The hypotheses are specific to the FPS genre because player performance has been documented to be particularly sensitive to increased latency [5].

3.4.1. HYPOTHESIS 1

"The player is able to maintain his performance level by adjusting game characteristics as a function of latency increase."

Several studies have confirmed that there is a relationship between latency and player performance. A typical example of such findings can be seen on Figure 1.

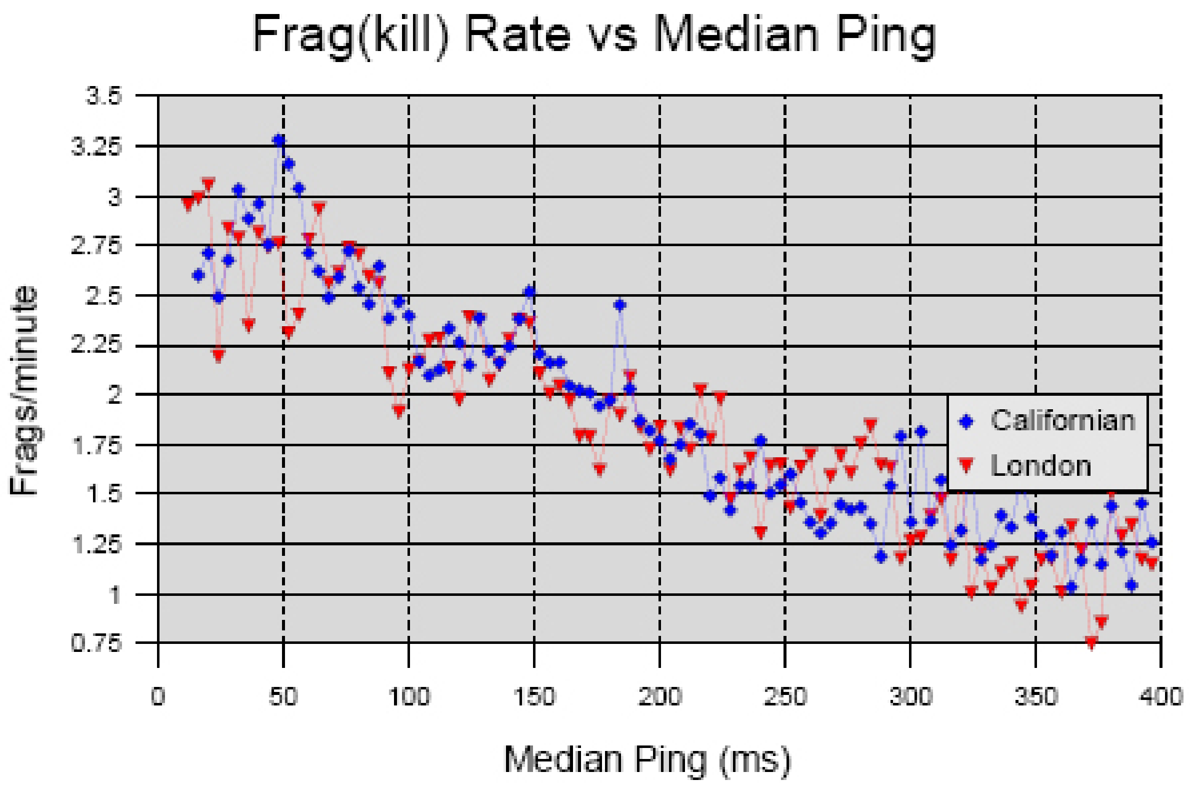


Figure 1: A typical outcome from a first person shooter game. The players performance is seen dropping as latency increases.

What the first hypothesis proposes is that by altering the characteristics of the game for each particular player, the network experience can be made more fair for all participants.

This is exemplified on Figure 2 where the red line represents the performance drop observable in other studies, such as on Figure 1. The yellow line represents a some kind of boost to a players

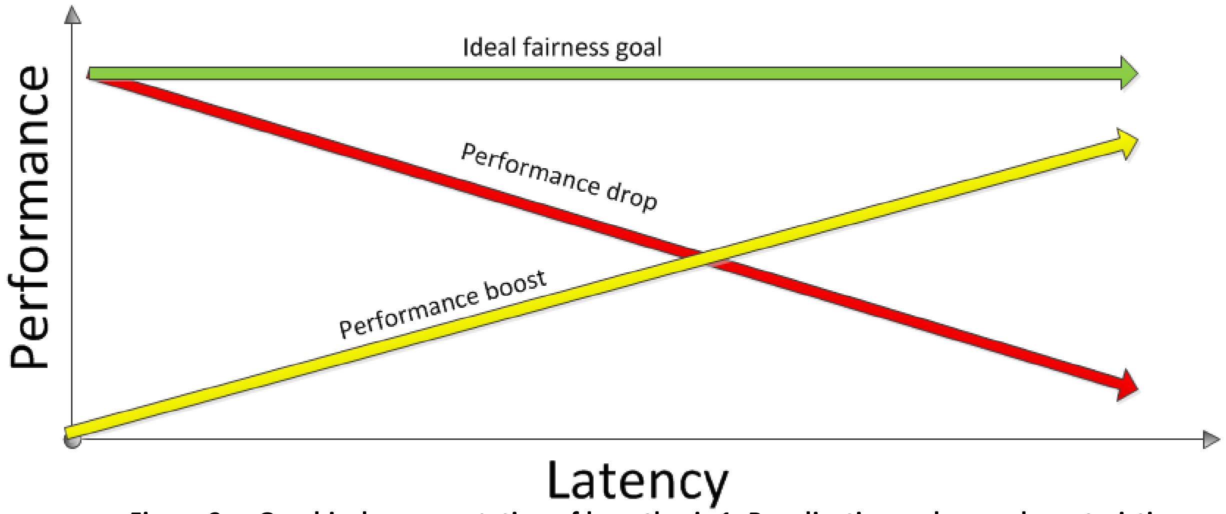


Figure 2: Graphical representation of hypothesis 1. By adjusting a players characteristics according to his delay to the game it is possible to maintain a constant performance output.

characteristics as the latency increases, and the green line represents the ideal goal of maintaining the same level of performance across all latency values. Essentially transforming the red line into the green one.

If successful the purpose of this hypothesis is to *maintain a players performance*.

3.4.2. HYPOTHESIS 2

"The player is able to maintain his regular role by non uniformly scaling weapon characteristics."

Many games have different roles or player classes depending on a player's preferences and play style. This often means that different player classes will have different equipment.

Studies have shown that not all activities in a game are equally affected by increasing latency [8] [5] [10].
An example of different game activities have been plotted on a Precision-Deadline plane as introduced by [5].

Figure 3 show how different genre's activities are differently affected by latency as the precision and deadline requirements of each component increases.

Precision refers to how precise an action must be in order to use it effectively, and deadline refers to the temporal precision, in other words how large the window of opportunity is with each action.

An example of a highly sensitive piece of equipment is a typical sniper rifle from a FPS game. Typically a sniper rifles damage is not spread out over any area, thus making the precision

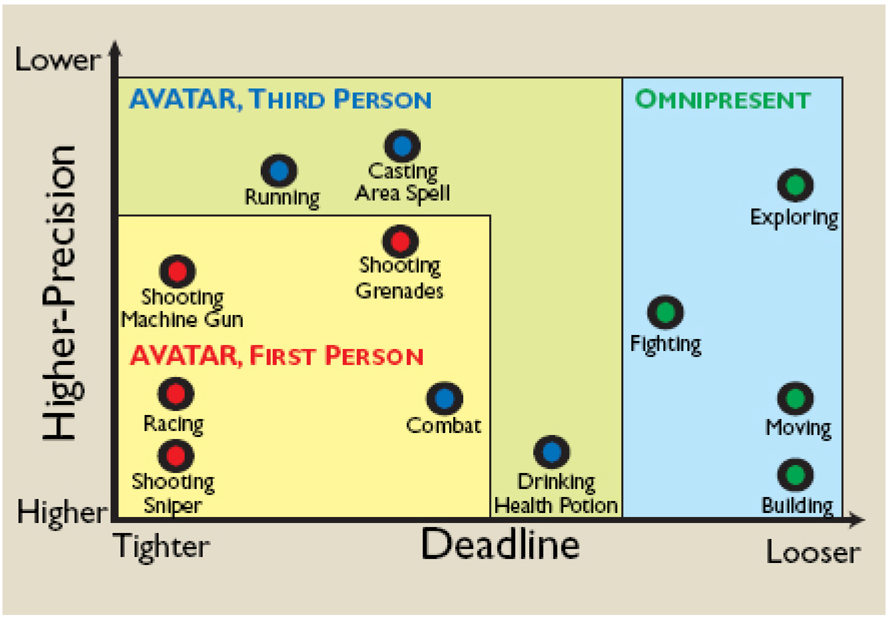


Figure 3: Various activities of different game genres plotted on a precision-dealine graph. The closer an activity is to the origin, the more it will be affected by increasing latency.

requirement high for it. A sniper rifle typically also has a slow rate of fire which means that there is a relatively long time before another shot can be taken should the first one miss, hence its deadline requirement is tight.

Another less affected example is a machine gun. It has a much higher fire rate which makes it less deadline dependent (it should be noted that this is not in agreement with the depiction on Figure 3 from [5]) and is typically spreading its damage over a larger area because it is somewhat inaccurate. Inaccuracy combined with a good fire rate gives it a greater chance to hit even though a player has moved during the time it took for the signal to reach the server. An illustration of this can be seen on Figure 4 from [10].

This is further cooperated by experimental results from [8] seen on Figure 6.

This second hypothesis suggests that scaling the characteristics of weapons as proposed in the first hypothesis, should not be done uniformly.

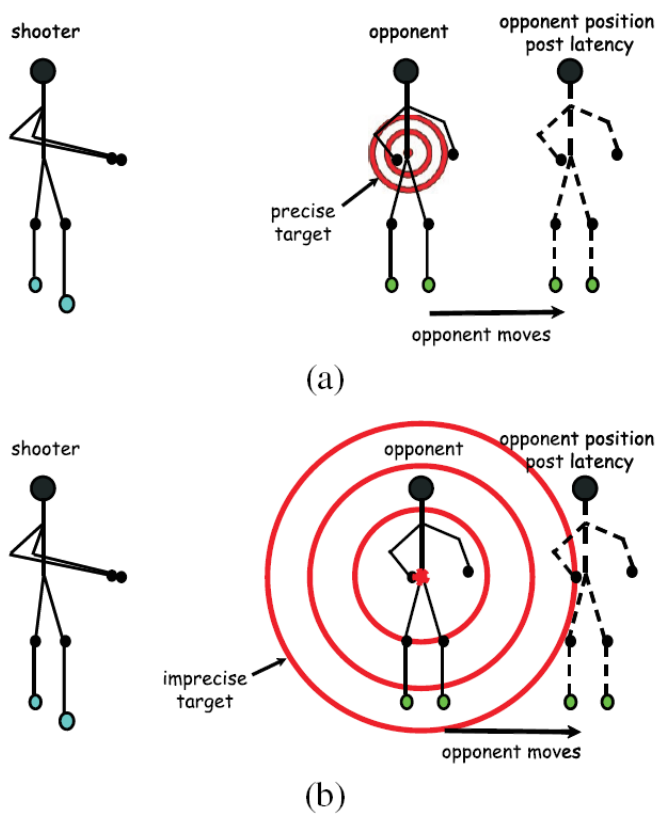


Figure 4: The advantages of having an inaccurate weapon is highlighted on part (b) of this figure showing how it is still possible to hit a target even though network delay has moved it.

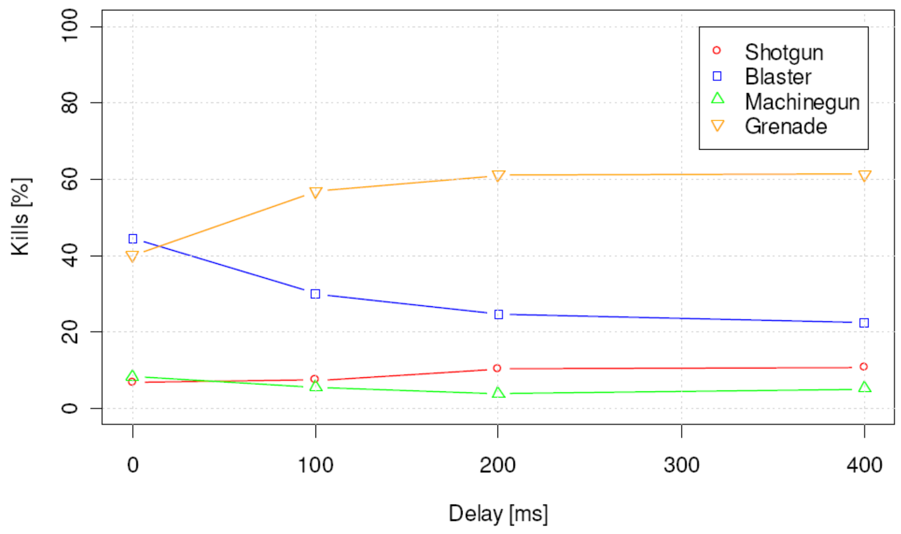


Figure 6: Data from a related experiment show that weapons with a greater area of effect are more effective as latency increases.

Meaning that weapons and activities that are more greatly affected by latency increase should be scaled more than weapons or activities that are less sensitive.

This is exemplified on Figure 5 where three different weapon systems have been plotted along with a proposed movement in the precision-deadline plane as latency increases. Essentially giving each weapon a vector in the three dimensional space (deadline, precision, latency).

What this hypothesis says is that the magnitudes in the precision-deadline plane are different so that in case of Figure 5: $S_0 > S_1 > S_2$.

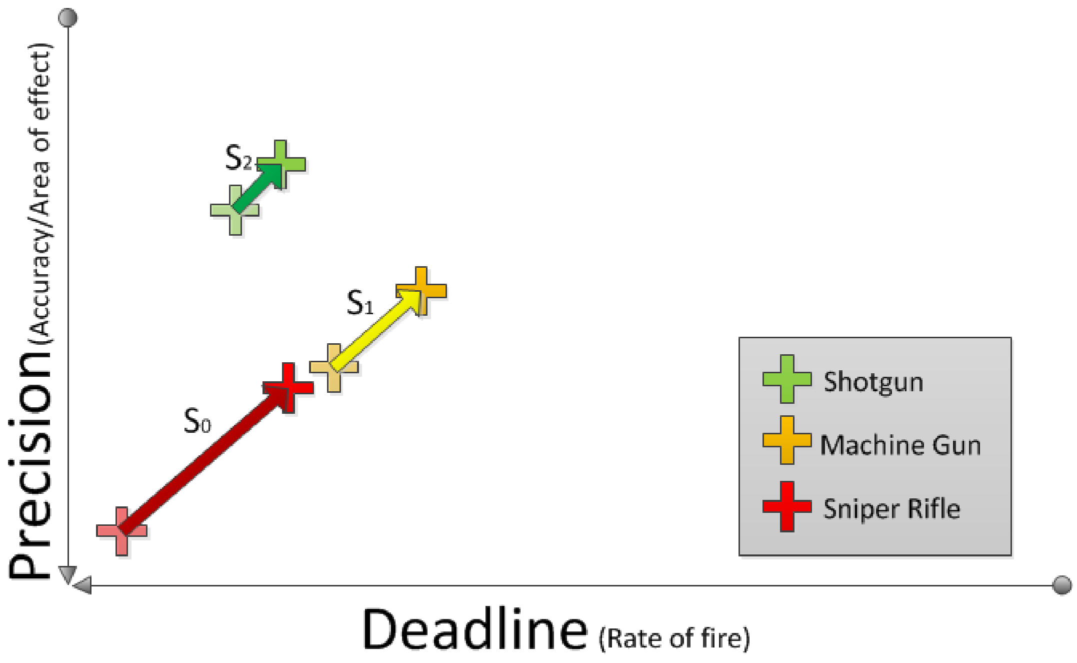


Figure 5: Graphical representation of hypothesis 2. Scaling latency sensitive items and activities more than those not heavily affected by it.

3.4.3. HYPOTHESIS 3

"There is a pattern to players' reactions to high latency that can be incorporated into the previous hypotheses."

It is hypothesized that players alter their behavior, consciously or subconsciously , when they encounter delays and inaccurate feedback to their actions when latency increases.
It is hypothesized that there is a common or a set of common behavior patterns that players will follow when encounter increasing latency, and that these patterns, or predictions, can be incorporated into the scaling methods of hypothesis 1 and 2.

An example could be that a player becomes more defensive and less active when encountering delayed actions and feedback, and as such is more vulnerable and is likely to run out of ammo because he does not want to risk the movement. Thus a player's ammo capacity or health could be scaled appropriately.

Very little research could be found on this subject, but instead an online survey was carried out to evaluate if there was enough validity for the hypothesis to be a part of the project.

The survey was largely unstructured and was carried out on an international gaming forum for mature and tactical minded gamers [11]. The survey was arranged as an open question about gamers' experiences with high latency and if they altered their behavior.
In total the thread got 25 replies with several patterns appearing in the unstructured stories and feedback.

In this section a series of quotes will be highlighted to illustrate the patterns, but a full version of the thread can be found on the DVD or by following the link in the references.

Quite a few responses indicated that players tend to change to a more defensive behavior as the following few examples highlight.

Most answers indicate that the player consciously attempts to be more defensive:

"Lag makes me:
Use stationary weapons/defensive tactics "

"Also I turn to more defensive behaviour, move less and camp more."

"I tend to sit back a bit and be more defensive, because I know I'll lose any reaction time duels when I go around corners and such. Might switch to less accuracy dependant weapons as well. Might even avoid combat completely if the game allows it."

"I try not to rush and instead move slow or camp for a while and look around better to still get the drop on the enemy (decreased movement, defensive behaviour)."

However a few examples indicate the exact opposite.

"I get more aggressive, making sure I'm the one rounding the corner and not some guy with a lower latency. If you just sit there, someone could potentially kill

you before you even see them.“

“I play very aggressively, taking risks, running right into close and melee range and rushing objectives with no real tactical plan.”

Another commonality in the answers it that players tend to switch to weapon types that has an area effect or high fire rate. In general also shooting more.

“The biggest difference i see from myself is when i notice the lag i shoot more.”

“And when I shoot someone I use the magazine. No small bursts, just the entire thing at their direction until they die.”

With the answers given it is evaluated that there is enough merit to the hypothesis background for it to be a part of the project perspective.

3.1. PROJECT OVERVIEW

In this section an overview of the project plan is presented. What is to come in the following chapters and finally this chapter will conclude with delimiting some of the work and presenting the goals for the project period.

The rest of the documentation will be structured into 5 additional chapters; “4. Design” where the design choices and procedure for creating a testable prototype platform is described, followed by “5. Implementation” where the construction of the prototype is discussed.

The prototype is then put through a test session which will be discussed in chapter “6. Tests”.

Finally the project as a whole is evaluated in chapter “7. Evaluation”.

3.1.1. PROCESS AND PLAN

As part of an overview of the project an outline of the planning will be presented as seen on Figure 7.
This project was originally planned for two people, but following the DADIU production one of the participants was hit with some personal issues that prevented much further involvement in the project. However, it was not finally decided if he was going to contribute or not until the start of May, thus a lot of time was wasted and the project was ultimately overestimated for one person.
On Figure 7 the parts planned for, but ultimately not incorporated in the project, has been greyed out to give some idea of the overall planned scope of the project.

All in all this has some influence on the project, most profoundly being:

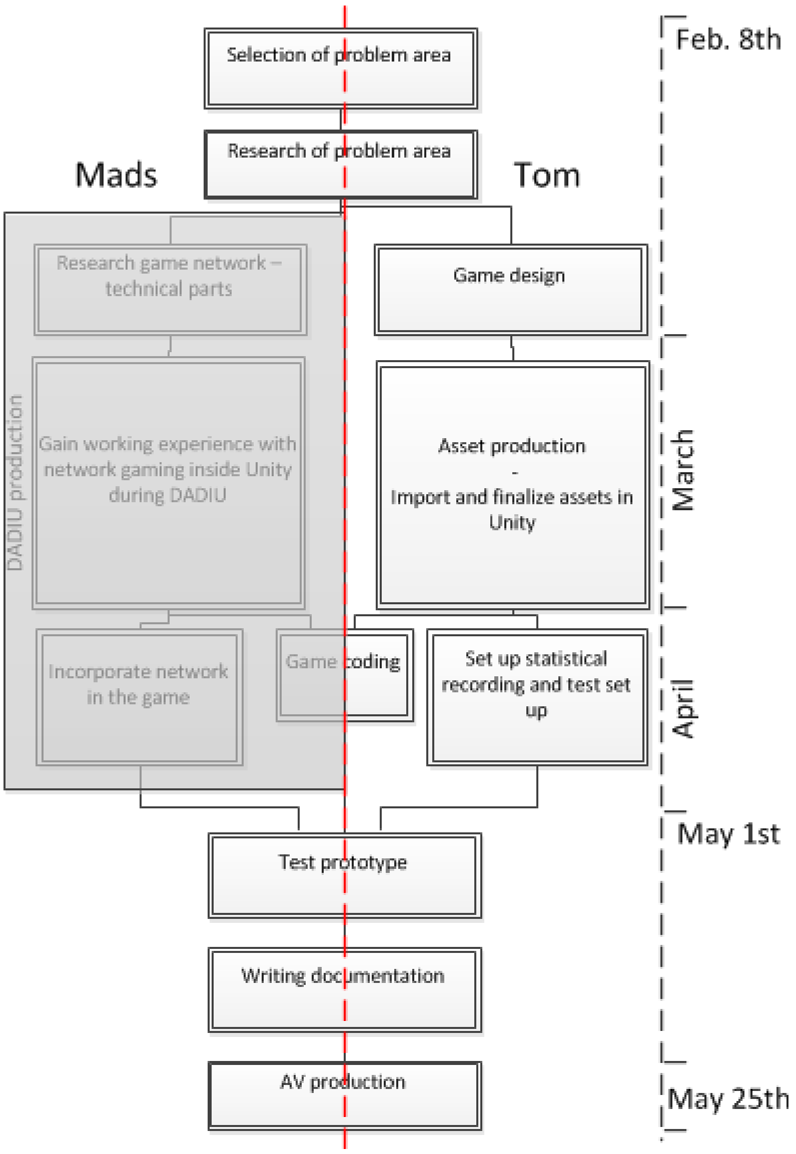
- No network in the prototype.
- The last three phases of the process was put a few weeks behind schedule.

3.1.2. DELIMITATIONS AND GOALS

For the purpose of the time frame of this project, even with two people, it was chosen only to evaluate hypothesis 1 for signs of validity, but still look for clues for hypothesis 2 and 3, that could be useful for further testing and planning.

For the span and scope of this project unit it has been chosen to set the following goals:

- Build a platform that can facilitate testing of the hypotheses.
- Evaluate hypothesis 1.



Feb. 8th
Selection of problem area
Research of problem area
Mads
Tom
DADIU production
Research game network – technical parts
Game design
Gain working experience with network gaming inside Unity during DADIU
Asset production - Import and finalize assets in Unity
March
Incorporate network in the game
Game coding
Set up statistical recording and test set up
April
May 1st
Test prototype
Writing documentation
AV production
May 25th

Figure 7: Project plan for both participants. Greyed out part represents work that was not done during the project time frame. Red line represents planned division of work load.

4. Design

Designing for the goals of making a game platform that can incorporate testing of the proposed hypotheses requires a lot of considerations in many different areas. Documenting all decisions in detail is an impossible task, thus this chapter will focus on the more important ones and leave the implementation sections practical insight to document some of the work done and describe the thoughts behind the decisions highlighted there.

The first part of this chapter will deal with establishing a base game design to work from. Defining genre, interactable content, features and player possibility for interaction.

The second part will deal with how to integrate a players latency into the game and equipment for the purpose of testing the hypotheses.

The third part will briefly deal with how to look for evidence of hypotheses 2 and 3.

The fourth and final part of this chapter will look at how to practically test it and catch up on subjects that was necessary to look into after realizing that there would be no network in this iteration.

4.1. CHOICE OF GAME

[5] and other studies conclude that the FPS genre is generally the most sensitive to latency when a players performance is in question. An example of this can be seen on Figure 9.

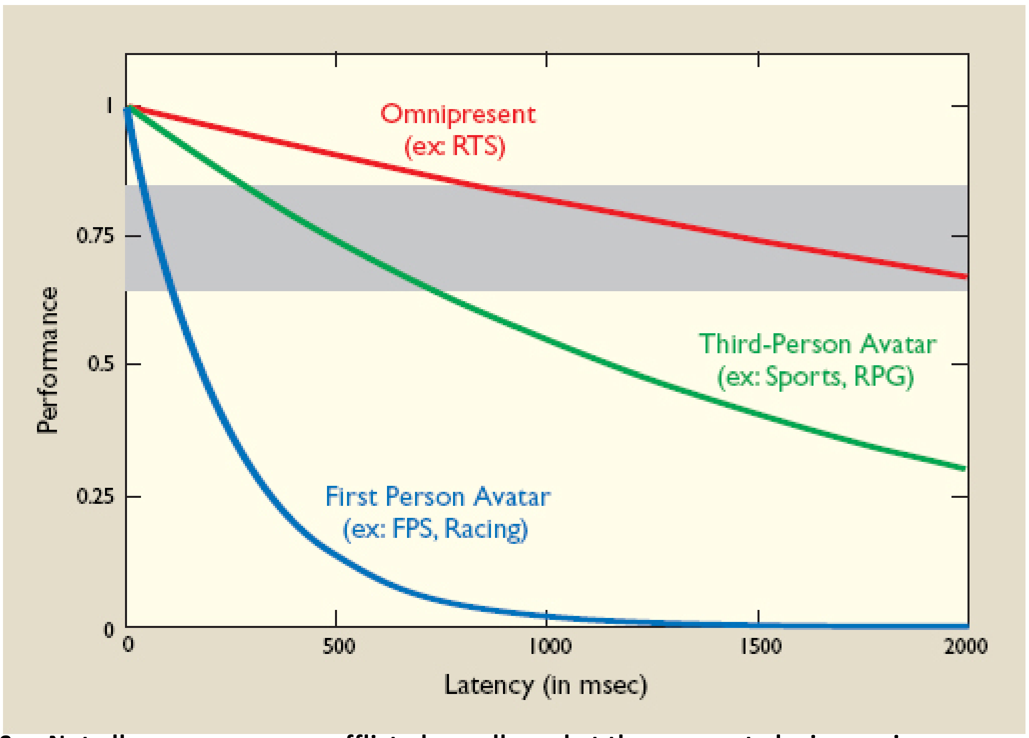


Figure 9: Not all game genres are afflicted equally and at the same rate by increasing latency.

It is favorable to choose something that is sensitive to the problem at hand because improvements should be easier to observe, thus it is decided to produce a FPS style game.

Figure 10 [10] breaks down an FPS game into the following primary components during play time:

- Movement
- Shooting

With only two primary components, both being simple in their nature, leaves a lot of the actual game play up to the type of enemies a player will face and the environment that is fought in. With the game being aimed at a human versus human multi player experience, there is little to do in designing opposition to a player.
Furthermore the simple nature of the game structure is favorable when making a prototype, so that not all production time is spent on the base structure of the game.

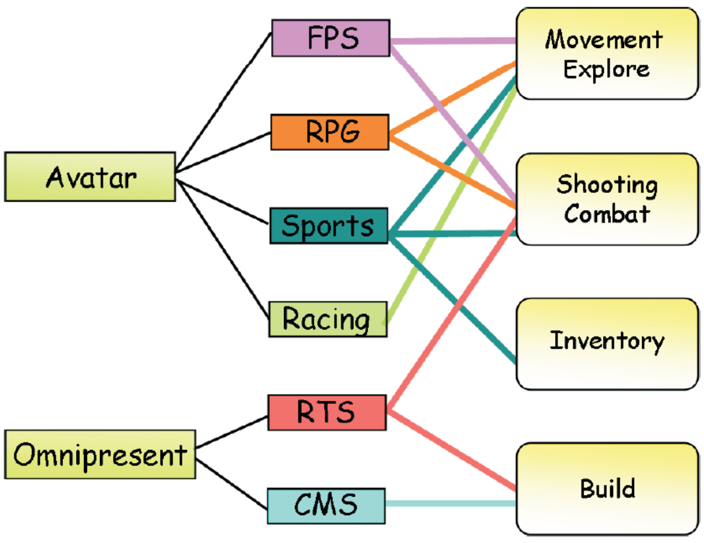


Figure 10: A break down of primary player activity components of popular game genres as set forth by [10].

4.1.1. GAME DESIGN

Game design involves making up and planning the contents of the game as well as the rules of a game. As the previous section highlighted, an FPS type game is mainly about facilitating shooting and movement. In terms of a multiplayer FPS the game play practically facilitates itself by giving the players a gun. Give a player a gun and he will want to shoot it, and then compare who was the best at it.

Nearly all games in the FPS genre function this way. Twists and alterations to the rules and way things are fought may occur through game modes, and the level of player decisions and tactics can also be altered through choice of items and characters, but in the end the game play is the same.

4.1.1.1. PLAYER SCORE

To make sure that players who are not as talented as others still have a chance of winning, and thus maintain their motivation to want to perform optimally, there must be a scoring system that facilitates this.
A scoring system that is based on how difficult each kill was is proposed for handling this. For instance if a player with a bad “kill to death”-ratio kills a player with a good ratio it should count as much more than if was the other way around.

A straight forward way of tackling this balance is to use the kill-death ratio of the player you just defeated and multiplying that ratio with some predetermined constant to make the value appear greater and then adding this as the point value.

In pseudo code this could look something like:

```
Points = ((defeatedPlayer.kills/(defeatedPlayer. deaths+1)))*value
```

The addition of 1 is there to make sure that a division with 0 does not happen.

4.1.1.2. GAME MODE

In order to eliminate some variables that might cloud the data it is important to have as simple a game mode as possible.

It is chosen to use a classic 'deathmatch' game mode so that events like friendly fire or blockading an enemy teams starting area will not influence the outcome of the games.

The deathmatch game mode is a free for all type of game, where all players are considered enemies and can kill each other. The win condition of a deathmatch type game is who can accumulate most kills and/or points.

4.1.1.3. CONTENT AND CAPABILITIES

Given hypotheses 2 and 3 which are related to behavior and roles, there must be some varied tools for the player to utilize so that alterations to behavior has a chance of being observed.

With hypothesis 2 it is apt to have weapons that are not equally affected by increasing latency, both to confirm what other studies have shown - that some weapons are more effective than others dependent on the latency situation, and to observe if there is a change in the users choice as his situation worsens i.e. if he is consciously aware of his worsened situation.

Finally features should not stray too much from the fundamentals of the genre. Making features as traditional as possible should ensure that the knowledge gained has the best chances of being a general representation for the genre as a whole.

With the above in mind a list of ideal features is constructed such that a player should be able to:

- Walk around an environment. While walking his weapon accuracy is worse than when stationary.
- Run, which will increase the players movement speed but worsen his weapon accuracy
- Crouch, which will make the player able to hide a larger amount of his character behind cover. It will also improve his weapon accuracy. The player is able to move around while crouched.
- Jump. Jumping will allow the player to clear short obstacles and gain access to higher vantage points. Weapon accuracy worsens while jumping.
- Fire his weapon. For each consecutive shot within a short period of time will increasingly worsen the players accuracy.
- Reload his weapon.
- Pick up items that will restore the players health, ammunition and boost weapon damage or player movement speed.

For the first iteration of this project 5 different weapons have been selected, 3 of them will be available to the player during play. Two of them are given:

- Melee attack.
- Grenade.

These two weapon types are classic to the FPS genre. The melee attack is often symbolized through a knife. Scoring a hit with a melee attack is often considered a killing blow due to the difficulty factor of getting close enough. The melee attack has limitless ammunition. Limiting the amount of work necessary to achieve a melee attack, it will be appear as the player smashing the buttstock of his gun at the enemy. This allows for a common animation to be used across weapons.

A grenade type weapon is very often standard issue in FPS games, giving the player the ability to cause an area of effect attack. This is often made as a slow moving object that is affected by gravity to enable it to make indirect attacks as well. To limit the number of animations and content needed for the first iteration, this weapon appears as a shoulder mounted grenade launcher on the player model.

The player then has to select his primary weapon amongst the following three:

- Shotgun.
- Machine gun.
- Sniper.

4.1.1.4. WEAPON MODELING

It is nigh impossible to find any practical guidelines or methods of achieving balance of game assets. A lot of it is perhaps down to immense amounts of play tests and tweaks that games go through during the Alpha and Beta stages of development. No guidelines to start from was found.

Thus there is a challenge in designing equally balanced weapons for the players, and distributing them differently across the precision-deadline plane.

It was chosen that in order to balance the weapons they must have the same output of damage per time.

This "damage/time" output common for the different weapons is chosen to be set equal to 100 damage/second. A players base value of health points is also set to 100, so this means that all weapons will be able to take down an enemy with 1 second of fire.

Some limitations and requirements are set to the weapon types such as the sniper rifle.
The precision of the sniper rifle is its greatest feature, and it should this be able to kill an enemy in one shot if placed correctly.

To facilitate this critical damage is introduced. Critical damage is achieved by hitting an opponent in the head and thereby scoring double damage.

With this requirement in place it is now possible to model the first weapon's characteristics.
To score a one hit killing shot the sniper rifle must deal at least 50% of a players base health so

that a head shot will deal 100% damage. This implies a damage value of 50, leaving the fire rate to be 0.5 seconds between shots.

Following this and using the precision-deadline plane as inspiration another route is taken with the machine gun. Starting by setting a fire rate of 0.1 seconds between shots and then arriving at 10 damage points per shot.

The final firearm to be handled is the shotgun. This weapon will need to distribute its damage across a number of individual projectiles per shot fired, thus spreading the damage over a larger area, like the machine gun, but even more so.
The number of projectiles is arbitrarily chosen to be 5. To make it effective in close combat it must also have a faster fire rate than the sniper rifle. With these requirements the remaining values, fire rate and damage per projectile are chosen to be 0.4 seconds and 8 damage points respectively.

Similar to having a constant damage/time value across weapons, each gun attempts to hold a constant potential damage per magazine as well as the total amount of ammunition the player is able to hold.

The only thing not determined by constants is the accuracy, or deviation a projectile has once it leaves the barrel. This is opted to remain tweak able such as to more intuitively define their precision component of their precision-deadline placement. After all the amount a projectile can deviate from the center of the aim is dependent on the size of the target and thus defining the effective range of the weapon, both of which are hard to anticipate in advance.

Figure 8 shows a summery of the preliminary values chosen for the weapons in an attempt to balance them. This first step is up for evaluation with the conclusion of the first iteration of the prototype.

Weapon	Number of projectiles per shot	Damage per projectile	Fire rate	Magazine size/ Total ammo
Shotgun	5	8	0.4 seconds	13/65
Machine Gun	1	10	0.1 seconds	50/250
Sniper Rifle	1	50	0.5 seconds	10/50

Figure 8: Shows the first iteration weapon base characteristics.

4.1.2. LEVEL DESIGN

In this section a set of requirements and guidelines will be established that should be followed in the execution of the practical level creation.

An important part of looking for evidence of player behavior change lies in designing an environment that facilitates different play styles and tactics.

- First off it is important to have a number of possible places a player can reenter the battlefield from so that a player cannot be harassed or taken advantage of when starting over.
- The battlefield should feature areas that allow for ideal long, medium and short range combat.

- Pick ups should be placed such that the player will have to leave the safety of cover in order to take them, or tempt the player to leave cover in some other way.

One reason why pick ups should be placed in the open is to observe a players willingness to take chances, and if this changes over time as indicated in answers to the survey in “3.4.3. Hypothesis 3”.

With these general guidelines established there is enough challenge in creating levels that facilitates all of them to a satisfying degree.

4.2. COUNTERING PERFORMANCE DROP

Other studies has attempted to make the game more fair for all by adjusting the game. In one experiment [8] the researchers attempt to balance the game by adding artificial latency to players who are at an advantage, thus attempting to equalize the latency difference between the players. They make an application “Self Adjusting Game Lagging Utility”, SAGLU, to do this.

The SAGLU application in [8] face some of the same issues and decisions that this project must also tend to:

> *“1. How to determine the additional artificial delay. The additional delay could be based on the highest player delay, a percentage of the highest delay, the average of the highest three delays, ect.*
>
> *2. How that delay should be added. It could be added immediately or ramped up linearly, exponentially, ect.*
>
> *3. How often a player’s network delay should be measured and the additional delay should be adapted. This is a trade-off between fairness and computational resources available.“ [8]*

While the proposed solution does not involve adding additional delay, it is still very similar in nature as it will be adjusting the player’s performance parameters instead.
Thus it is important to make good decisions on:

- How much to boost the player depending on his latency.
- When to measure and evaluate the player’s latency.
- When to adjust the players parameters.

These three issues will be commented on in the following sections.

4.2.1. MODIFYING CHARACTERISTICS

Figuring out exactly which player characteristics needs altering to boost a players output performance is difficult. Do you boost all characteristics, or just a few? Do you give increased movement speed? Damage? Health? Jump height? Jump speed? Where is the threshold?

When taking a deeper look at what components and variables a player consists of it is clear that there is a lot of them, and not all effects of boosting all variables can be anticipated without experimentation.

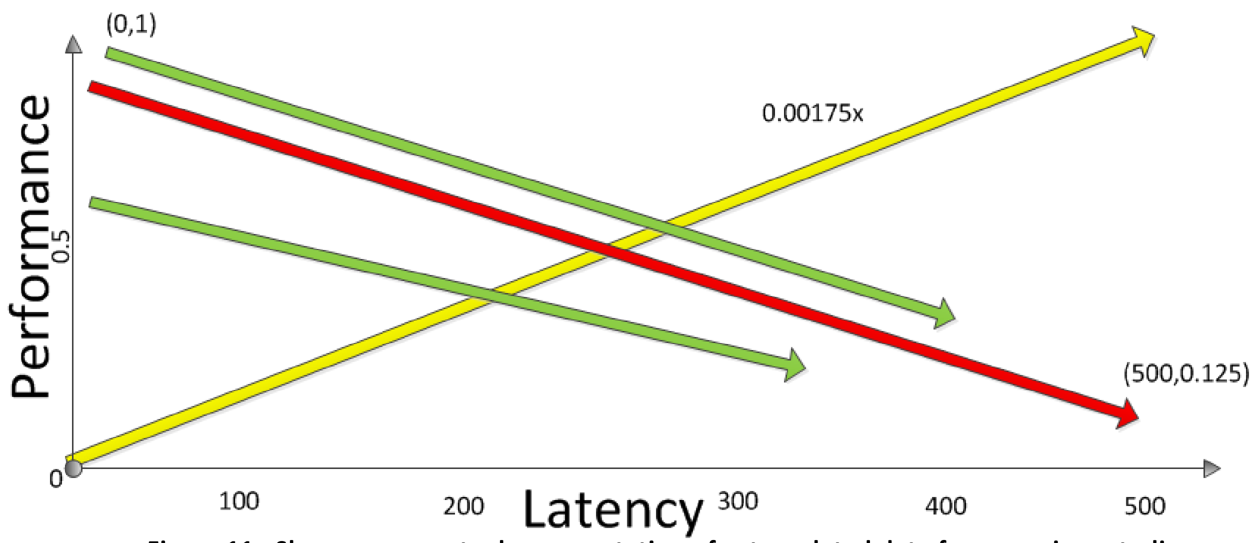


Figure 11: Shows a conceptual representation of extrapolated data from previous studies as well as the factoring coefficient chosen for first iteration of this project.

For the purpose of evaluating hypothesis 1, as set as a goal for the scope of the project period, it is chosen to only boost the most obvious characteristics of a player:

- Weapon damage
- Health

A weapon's accuracy and fire rate is not adjusted in this iteration as it will be evaluated through an investigation of hypothesis 2, which is not the focus of this iteration.

With the variables that will be boosted now selected, it is time to decide what model they will be boosted by.

Looking at graphs and data of previous studies on the subject of performance loss as a result of increasing latency, it is observed that the drop is practically linear all the way out to the 400-500ms marks. Planning beyond 500ms is not too important as several studies [3] [4] [7] have shown that players tend to stay below 200ms latencies to servers (depending on the particular FPS game of course).

It is obvious that the exact drop rate in performance is ultimately down to the specific game, however, a starting point is needed to experiment with.
The final boosting rate should of course be based on measurements from the control parts of the planned experiment, but with a time constraint it is chosen as follows.

By extrapolating data points from an array of graphs of previous work, such as Figure 1, resulted in a player performance drop coefficient of roughly -0.00175 performance/latency. Other results yielded varying but ultimately close drop rate such as -0.00117 or -0.0017. A conceptual example of this can be seen on Figure 11.

Since the drop is of a linear nature, it is chosen to counter the fall with an inverse proportional boost of 0.00175.
In other words, scaling weapon damage, health and movement speed with 1+0.00175 times the latency value in an attempt to make player performance constant and thus evaluate hypothesis 1.

4.2.1.1. WHEN ALTER? AND WHEN MEASURE?

With variables and scaling method now established, it is time to decide when the above changes to a players characteristics should occur.

It is assumed that in the following experiments that players will not attempt to cheat by spiking their own latency to gain advantages. However it would be very naive not to consider this when designing for when characteristics should update.

It is tempting to continuously update characteristics by simply linking the immediate latency to the above scaling, however spikes and general noise will make it hard for a player to play get a feel for the situation, not to mention the confusion it is bound to give when attempting to walk at a constant speed.
Thus consistency is something to strive towards.

Another disadvantage to the above is how a players health should be handled if it is constantly shifting. Again consistency is of importance, both for handling the internal workings of the game, as well as player perception.

A proposal for a solution is that the changes stick with the player throughout a life, thus applied when entering the battlefield anew. This is of course assuming that the game is fast phased with a relatively short life expectancy of a player.
It is furthermore proposed that latency should be evaluated periodically, at a rate that keeps game performance as a high priority. These samples should then be filtered using some variation of a weighted running average, thus making it harder to gain an advantage by momentarily spiking your own delay to the server.

4.3. MEASURING PLAYER BEHAVIOR

It is hard to anticipate exactly what to look for when searching for player behavior change. A players behavior can change in so many ways, so it is important to look at everything a player can do and then afterwards look for differences in the data sets.
Establishing a player's regular behavior, by observing his various statistics without any significant amount of latency and then comparing his statistics with games where he is exposed to high latency values and see if there is a difference in some areas.

4.3.1. STATISTICAL LOGGING

As described above, it is wanted to quantifying and logging all player actions available to him such as:

- Shots fired.
- Hits.
- Number of headshots.
- Damage done.
- Kills.
- Deaths.
- Jumps.
- Amount of time spend running.
- Amount of time spend crouching.
- Distance traveled.
- Number of reloads.

- Number of pick ups gathered.

And many more conceivable statistics one could gather about a player and his choices and performance, such as:

- How long the player has spend in cover.
- What weapons he chooses.
- The range to the kills made.
- What weapon was used to successfully kill.

In the end one can break these parameters into more and more variables such as damage done per weapon, or amount of sideways walk versus forward walk, or where a player has been.
Such detailed statistics is not sought after in the first iteration of development, because it is hard to anticipate exactly what patterns might emerge, even with the indications of "3.4.3. Hypothesis 3".

Therefore it is chosen to implement as many different statistics in order to search for evidences of patterns. Should such patterns start to emerge, one can focus further tests and action logging on this.

Alternatively one could also log a player's input - his key and mouse presses and see if there is a behavior change in his interaction - unclouded by the game statistics themselves. However this is beyond the scope of this iteration of the project.

4.4. HOW TO TEST IT

Like an array of the different studies that this project is based upon, an experimental set up is needed where it is possible to simulate different levels of latency to particular users. Other studies have used NISTNet [13] or a windows version of it, DummyNet [12] to simulate an array of network conditions to their test subjects.

It is favorable to have all test subjects placed in a LAN environment where the base latency is as low as possible so that individuals can be singled out and their latency increased and thus make comparison of players easier with the control.

As suggested earlier, it is also important the keep all players motivated in order to get more consistence in player performances. This part is handled by arranging the experiment as a tournament, and having the score balancing take care of the prospect of winning. It is also important to keep players motivated so that they are willing to play for a long time since a huge amount of data is needed to say anything conclusive with as complex an environment as human versus human play.

4.4.1. WHEN AND HOW SHOULD THE ARTIFICIAL LATENCY BE ADDED?

If the latency is chosen at random for each life then the player will no idea about it before his choices have been made and is unable to anticipate it. Thus a latency value - if a behavior change is what you are looking for, should remain constant for a fair amount of time or increase or both.

If it is simply a matter of looking at the raw performance output a random amount of latency can be chosen.

However, if looking for evidence of hypothesis 3, it is important to have the players play for a significant amount of time before adding any latency, in order to get a good picture of their normal behavior.

4.4.2. NO LATENCY. WHAT NOW?

When a working network version was no longer an option within the time frame of the project, alternatives had to be reconsidered. With such an important part of the project not being feasible, the entire focus of the project was in jeopardy, ultimately a choice had to be made.

Since latency in its rawest form is a delay, then a local delay could be introduced if the player was given something else to fight than human players.
While latency and delay refers to nearly the same thing, there is a few practical differences that will likely be influencing the players perception and thus the experience and performance.
Some of these differences lie in the latency compensation techniques applied by a lot of contemporary games.

Most FPS games operate with a authoritative client-server structure where all player input is sent to the server, which then executes the input. This is a good way to minimize cheating in the way that one player cannot send a message where he tells the server to kill another player. All input is interpreted by the server.
However, waiting for the input to be processed and the feedback send back to the player is disruptive and unplayable for a fast paced game. Therefore a prediction component is added to this structure so that it allows the player to carry out his actions immediately. The server is constantly predicting the position and action of the player and if the prediction and player input does not match, the player's view is corrected.

With this prediction in place it makes sure that the player is not practically delayed the equal amount of his latency. However, with increased latency the number of prediction errors, and thus corrections to the players actions is increased, making the player stutter or "lag" around the environment.

While latency and delay is not practically the same, it is chosen that for the purpose of searching for clues of the hypotheses validity, making the assumption that the two are the same is acceptable.

With this assumption two things are needed:

- Delay the players input by a variable amount
- Give the player something to fight against

Many games feature AI characters to play against, also known as bots. Bots are essentially capable of carrying out the exact same actions as a player, but are ultimately (in most cases) not able to learn and adapt tactics. In fact, because of this consistency of behavior, bots have been used as stand ins for human players in several of the studies referenced to in order to gain large amounts of data quickly.

With a single player version, the plan has now changed to a survival type of game, where a player has a certain amount of lives and has to defeat as many bots as possible. All bots are allied and will attempt to kill the player.

4.5. DESIGN SUMMERY

This chapter has addressed a number of the issues faced by other studies and attempted to make choices that will aid in the evaluation of the proposed hypotheses.

A game has been chosen that will be both sensitive to change in the desired field, as well as be relatively simple in its structure - thus allowing for a quick production.

Primary content for the game has been decided, the rules and basic parameters set forth.

More importantly, the problem of how to when and how much adjust that should be made to the player has been defined as well as proposing a few alternatives for further experimentation.

Furthermore, preliminary guidelines on how to approach an experimental set up and gather data has been proposed.

Finally last minute adjustments was made to the plan of the duration of this project after a network game was no longer an option.

5. Implementation

Due to the spatial and interactive nature of much of the practical work, it has been chosen to document it in a video format as it is both time consuming and difficult to properly portray many of the practical problems, solutions and techniques on paper with static images.

These videos will highlight some of the different aspects of the game production as well as practical problems and chosen solutions - documented in a "making of" - style.
These can be found on the accompanying DVD.

All in all the videos will cover the following areas of the prototype production:

AV-Production.

This video will give and introduction to the project as a whole

Asset production

Production and design choices for the assets of the game. This video is rather long and is focusing on a lot of different techniques using the production of the game, while this is interesting to me, it is of relatively little importance in relation to the focus of the project i.e. it is not necessary to watch if not of interest to the reader.
The video will cover the following subjects:

- Concept and style
- Different asset production techniques
- Interactable assets
- Characters
- Rigging and animation
- Environment
- Sounds
- Effects

Game features and presentation

This video will highlight what work has been done on the prototype, what features it possesses and partly how it was achieved. In contrast to the "Asset production" video, this is of a more technical nature such as:

- Game features and core components.
- Levels.
- User Interface.
- Weapons.
- Statistical logging.
- Character control.
- Delay and LIGM.

The written part of this section will be very lightly documented on paper, though it will attempt to give an overview of some of the key aspects of the production that has relevance to the projects focus.

5.1. IMPLEMENTATION PROCESS

With a network game no longer a solution in the time frame available, another solution was needed. The lack of network has an impact on several areas of the project, but most important is perhaps the latency that the project relies on.

Creating the network was at this point no longer a possible solution as I had not focused my readings on it, nor was a week a realistic time frame to do so while finishing up the remaining game logic.

Alternatives were needed. Making the game single player would require several additional tasks, but in comparison it felt as the more realistic.

Additional tasks include:

- Creating opponents to play against
- Introducing delay

5.2. FEATURE DESCRIPTION

Of the basic requirements of the FPS genre proposed in the design chapter, the player have the following capabilities:

- Walk around an environment.
- Run.
- Crouch.
- Jump.
- Fire his weapon.
- Reload his weapon.
- Pick up items that will restore the players health, ammunition and boost weapon damage or player movement speed.
- Perform a melee attack.

Notable features that did not make it in time are:

- The ability to perform a grenade attack.
- A weapon's deviation is not dependant on his state (stationary, running, jumping, walking, crouching, ect.).
- The weapon's deviation does not get worse for each consecutive shot.

The ability to make a grenade attack is not necessarily needed in the first iteration, but would have been nice to have to look for evidence of increased usage or effectiveness at higher latency values.

While the game does not differentiate between the different states the player is in when it comes to weapon accuracy, it does feature it in a way. The virtual camera is linked to the player avatar's upper body and thus the shaking of the camera, and thus the cross hair is intensified as the player walks and runs compared to when the player is stationary.

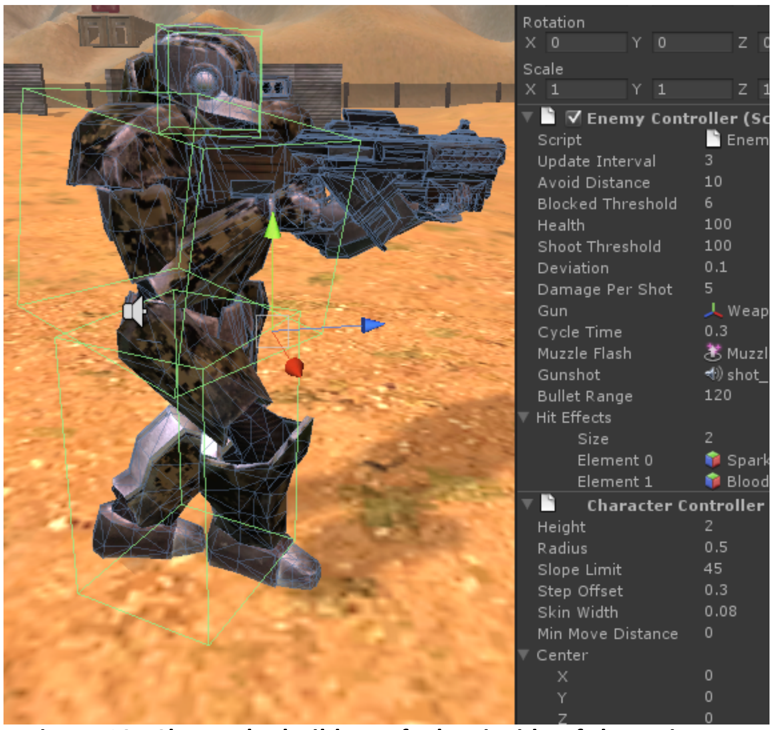
5.2.1. BOTS

An AI player contain three components, a behavior module, a movement module and a weapon.

Behavior module

The behavior module is responsible for feeding input to the movement module. Given it a direction to move towards. The behavior module controls the other two components, and it is updated at a predefined interval.

The enemies will always attempt to move towards the player, unless they are about to collide with another object that is not the player. In this case they are instructed to move to their right for a period of time and then attempt to follow up on their pursuit of the player. While this is a unexciting and simple approach, it works very effectively in chasing the player.



```
Rotation
X 0          Y 0          Z 0
Scale
X 1          Y 1          Z 1
▼ ✓ Enemy Controller (Sc
  Script                 Enem
  Update Interval        3
  Avoid Distance         10
  Blocked Threshold      6
  Health                 100
  Shoot Threshold        100
  Deviation              0.1
  Damage Per Shot        5
  Gun                    Weap
  Cycle Time             0.3
  Muzzle Flash           Muzzl
  Gunshot                shot_
  Bullet Range           120
▼ Hit Effects
      Size               2
      Element 0          Spark
      Element 1          Blood
▼     Character Controller
  Height                 2
  Radius                 0.5
  Slope Limit            45
  Step Offset            0.3
  Skin Width             0.08
  Min Move Distance      0
▼ Center
    X                    0
    Y                    0
    Z                    0
```

Figure 12: Shows the build up of a bot inside of the Unity editor.

Furthermore the behavior module is also responsible for determining when and who to shoot at. Since the player is the only enemy in the eyes of the AI players, they will only have the player as their target.

The AI players will start shooting at the player when the target is within a predefined range and there is a clear line of sight to the target.
Some of these predefined values can be seen in Figure 12 with the settings used for the test part of the prototype.

Movement module

The movement module is responsible for moving the player around the battlefield. For this instance the default character motor that comes with Unity is used. Once given a vector, the character motor will move in that direction using the physics engine inside of Unity.

Weapon

The AI players all have the same weapon for this prototype in an attempt to make the opposition offered by the enemies as consistent as possible. The weapon functions much like the player's does, but it is less powerful and slower firing than most of the players weapons. It is however fairly accurate over short and medium ranges.

On each map there is an object that creates new enemies. It keeps track of how many enemies are alive, and will periodically instantiate new enemies to maintain a predefined number of opponents for the player to battle. Enemies will be instantiated on an arbitrary index of a predefined set of locations.

Besides these modules an AI character has a health value, that when it reaches 0 the enemy die. Like the human controlled player, the AI players has 3 hit boxes; head, torso and legs. If hit in the head, they it will loose double the amount of health compared to being hit in either of the other two spots. These 3 hit boxes can be seen on Figure 12.
Once an AI player dies it will tell the player that he has made a kill.

5.2.2. INDUCING DELAY AND ADJUSTING PLAYER CHARACTERISTICS

In order to simulate the latency that this project revolves around, local delay was partly incorporated into the player's control of his character.

Very simply, when the player presses a button the game waits for the desired amount of time before calling the appropriate function such as firing or reloading the weapon.

Unfortunately no reliable solution was found for delaying the movement input of the character given the nature of the Unity character motor. This is unfortunate, but with a players offensive capabilities functioning as intended it was acceptable enough for the first few experiments.
It should be noted that, due to an oversight, the melee attack is not delayed, which results in some source of error in the test chapter.

5.2.3. LOGGING PLAYER ACTIONS

Logging the players actions and statistics was done by writing usable variables to a .txt file each time the player dies.
The text file has the following naming convention:

SessionID_PlayerName_life_DelayStatus.txt

The SessionID is a random number given to the file to minimize the risk of it being overwritten. The player name is the name the player has chosen for his character. The life refers to how many times the player has died. Finally the DelayStatus is whether the player is delayed, not delayed or delayed and with LIGM applied.

This file contains working statistics of the following parameters:

- Kills (cumulative value of the whole session).
- Deaths (cumulative value of the whole session).
- Accuracy (for the particular life).
- Shots fired (for the particular life).
- Shots hit (for the particular life).
- Headshots (cumulative value of the whole session).
- Weapon type (0 = shotgun, 1 = machine gun, 2 = sniper).
- Number of melee hits (for the particular life).
- Time alive (for the particular life).
- Pick ups gathered (for the particular life).
- Time spent in cover (for the particular life).
- How much the player is delayed (for the particular life).
- The multiplication he receives from LIGM (for the particular life).

- What map he is playing on.

An interesting part of the above statistics is the "time spent in cover". This variable is achieved by defining areas of cover on the map with a trigger box and noting when a player enters and leaves such a box. An example of areas defined as cover can be seen on Figure 13.
Figure 13 also gives an overview of 3 unique and playable levels made for the game.
Each have different sizes, amount of enemies, theme, layout of cover and pick ups.
Another interesting tool is the player tracker. The player tracker periodically writes the position component of the player to a .txt file. This file can then later be viewed using an game object with a line renderer attached to it. This is implemented so that it is possible to observe some of the session should there be anomalies in the data of the statistics files.

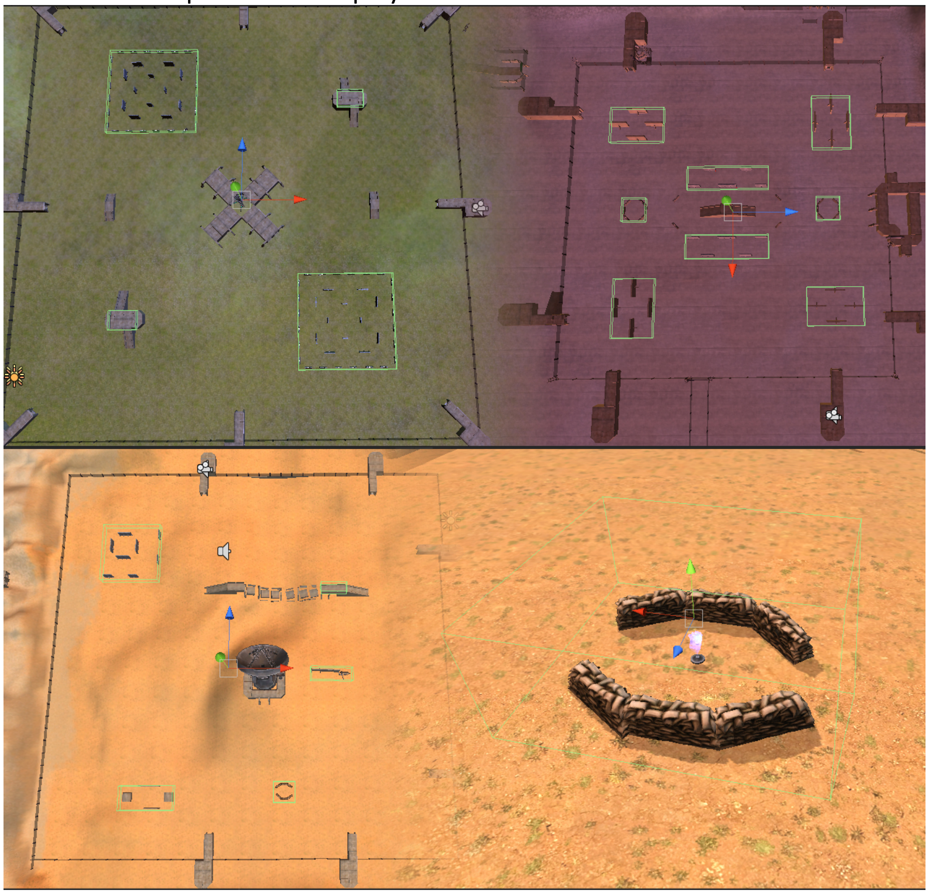


Figure 13: Shows designated "cover zones" on the three different maps, as well as a close up of a sand bag bunker and its cover zone trigger box.

5.3. IMPLEMENTATION SUMMERY

Much of the planned features of the design chapter made it in to the working prototype despite of only one man doing all of the code work. However there are a few missing features that perhaps could have ensured more usable data in the next chapter. Furthermore the "finish" of the game also suffers greatly from reduced man power. Not all animations are blended between. A menu is virtually non existent and there are unprofessional looking left overs in the graphical user interface such as character selection, that does nothing.

Despite the shortcomings there was made a playable and stable FPS that features 4 of the 5 planned methods of attack. The player can pick up bonuses and the players statistics and movement is logged for later analysis.
The adjustment to test hypothesis 1 of the LIGM is implemented, boosting the player's weapon damage and health as a function of the amount of delay he is suffering.

For a greater insight into the countless components of making a game I recommend watching the accompanying videos on the DVD.

6. Tests

In this chapter it will be discussed how two tests were conducted. Analysis of data and a critical look at the first test led to a secondary test which ultimately highlight other issues.

6.1. TEST 1 - PILOT TEST

Much like described in the design section, this test session was organized as a tournament. 4 players in the age 15-18 was invited under the pretense of helping alpha test a new game and hunt bugs. All four players was present at the same time during a LAN marathon so that there still could be a competitive feeling to it even though they were not directly playing against each other.

While this test was not specifically meant to look for evidence of hypothesis 2 and 3, it was still organized into three stages of test:

- "Training" with no delay enabled to get a base for how each player behaves under normal circumstances
- "Competition 1" where the delay was enabled and continued to increase for each life the player had. This was to see if there is a difference in behavior and performance as the delay increased.
- "Competition 2" where the game was still delayed, but the LIGM was also enabled.

A conceptual representation of the game logic can be seen on Figure 15. However for this first test the selection of delay, no delay and LIGM is not random. It was chosen by the user.

Each stage of the test the player played each of the 3 maps for a number of lives. The number varied during the session and the reasoning will be discussed in a following section.

The players were asked to select the different settings as I instructed before playing a new map. Knowing nothing about game development they

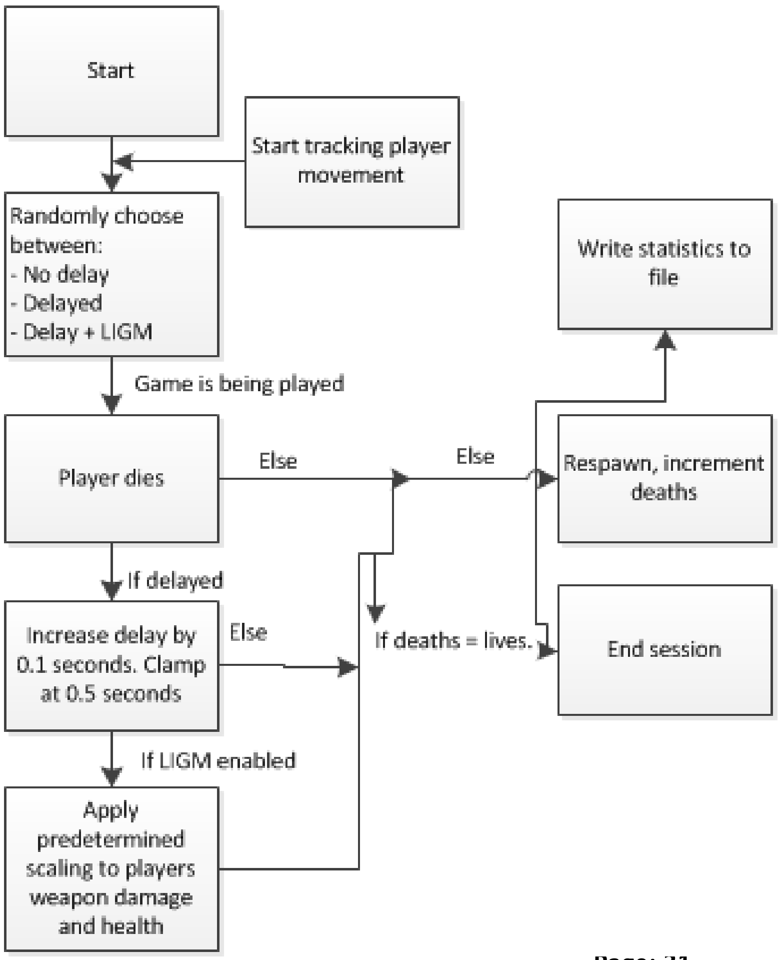


Figure 15: Overview of game session logic.

were told that the 3 settings were 3 different ways the game used the graphics card in their computer, and that I wanted to see if there was a difference in between them.

6.1.1. DATA ANALYSIS AND SOURCES OF ERROR

Even with a combined 5.4 hours of recorded play between the four test subjects, very little can be concluded from the test due to a number of factors. Before the analysis goes any further it is evaluated that this test should be viewed as a pilot test.

The first issue that was encountered was that players figured out a way to keep themselves alive for a very long time and thus each game took so long that the players got bored. The problem being a flaw in the time it took for a pick up to respawn and the way that they were placed. This enabled the players to continuously circle the level and pick up the health boosts.
Figure 16 shows two different player on two different maps carrying out this circling and thus achieving both long life and boredom. The image was made using the player tracker discussed in the implementation chapter.

As a result of this the respawn time of the pick ups was changed from 5 seconds to 30 seconds. And lowering the number of lives from 10 to 5.

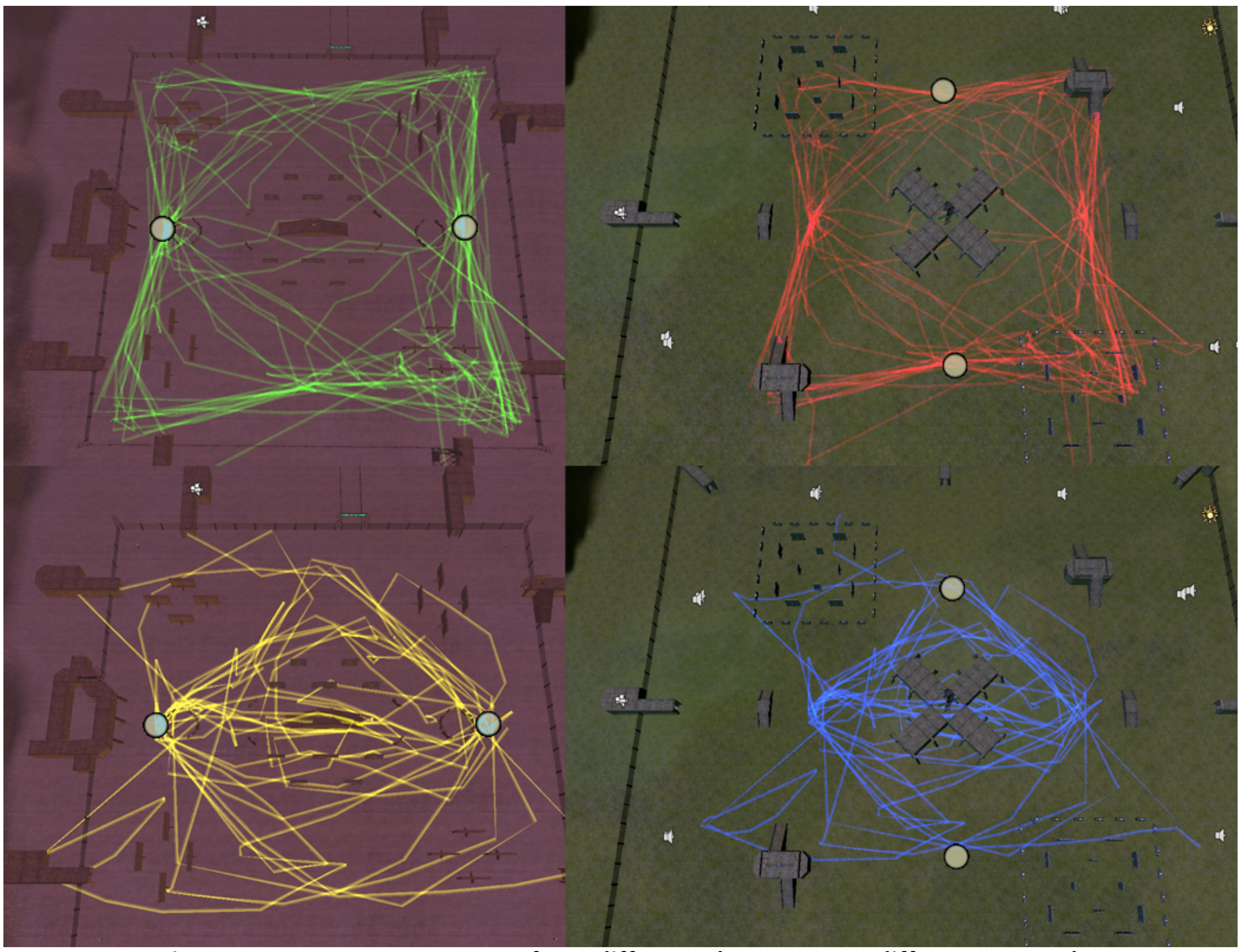


Figure 16: Movement patterns of two different players on two different maps. The movement patterns show a systematic approach to surviving using the health and ammunition pick ups.

However this did not entirely eliminate this exploitation. This change was made from test 1 “Training” to “Competition 1” and thus it is hard to compare the average “time alive” and “number of pick ups” gathered from the control to the delayed versions.

However some pieces of data could still be used such as accuracy and kills/time.
It should be noted that when analyzing the data, only entries made with the players’ most used weapon was used.
Some of the data deemed usable from the entirety of the test session can be seen on Figure 14.

Player /Attribute	Alexander	Simon	Mads	Nick
Kills/minute - no delay	6.84	9.08	6.73	5.14
Kills/minute - delayed	7.61	9.86	8.79	3.89
Kills/minute - LIGM	10.85	9.04	11.66	4.79
Favorite weapon / percentage used	Machinegun / 94%	Sniper / 91%	Shotgun / 73%	Sniper / 47%
Accuracy - no delay	52.7%	53.6%	19.9%	38.5%
Accuracy - delayed	41.6%	51%	8.9%	51.6%

Figure 14: Table of performance statistics and player choice across the three different settings; “no delay”, “delayed input” and “delayed input + LIGM”.

One interesting observation was that players tended to use only one weapon throughout all of the versions. This merits the fact that players take on a role as postulated in the description of hypothesis 2. Furthermore it appears that the weapons are indeed balanced based on the damage modeling done in the design phases as each weapon is used heavily.

The test suffers from a number of other major sources of error. Lack of data is a primary source of error. Even with 5.4 hours of active play time.
This is partly due to making sessions only last 5 lives, but the alternative was players getting even more tired.
Another major source of error is that the players did not choose the correct settings they were asked to when playing, which makes the distribution of data scatter instead of having equal amounts of each of the settings.
To add to this, one of the test subjects accidentally deleted his game folder containing 30 lives of data.

In fact there is a long list of potential sources of error such as bugs where the bots are able to shoot through cover or if a player dies while being damage boosted by a pick up results in the player dealing 0 damage his next life.

Other sources of error are there from stupidity such as displaying the players health value and displaying the damage done on an enemy as floating numbers. The players quickly noticed this when they were using the LIGM version, noticing that they had more health and were dealing floating point damage values.

Further tweaks to the difficulty of the bots should be made, as one player discovered that he could effectively use his melee attack as a primary weapon and not shoot at all. Relying on hit and run tactics and collecting health pick ups afterwards.

6.1.2. TEST 1 EVALUATION

As stated in the start of this section, this first test should be viewed as a pilot test only. However the very points the test failed in the eyes of evaluating the hypotheses, it succeeded in highlighting a lot of problems that would have been a disaster if applied to a full scale test. It seems that it was in fact me who was fooled by my own illusion “that they were helping test the game and find bugs”, which was exactly what they did, both for the game and the test procedure.

One of the only usable pieces of data from test 1 can be seen on Figure 17, where it is possible to observe a dropping accuracy as delay increases, indicating that there is somewhat of a performance drop as the input delay is increased.

Another good indication of the test is that the game performed more stable than expected. There were no crashes or sudden drops in frame rates throughout the +5 hours of testing. This should prove that the practical work done so far is approved for further development. However many desired changes from the perspective of an artist with a technical mind set are still planned but somewhat irrelevant for the focus and scope of this project.

The player tracking feature has also turned out to be a valuable asset in gaining insight of inconsistencies in data.

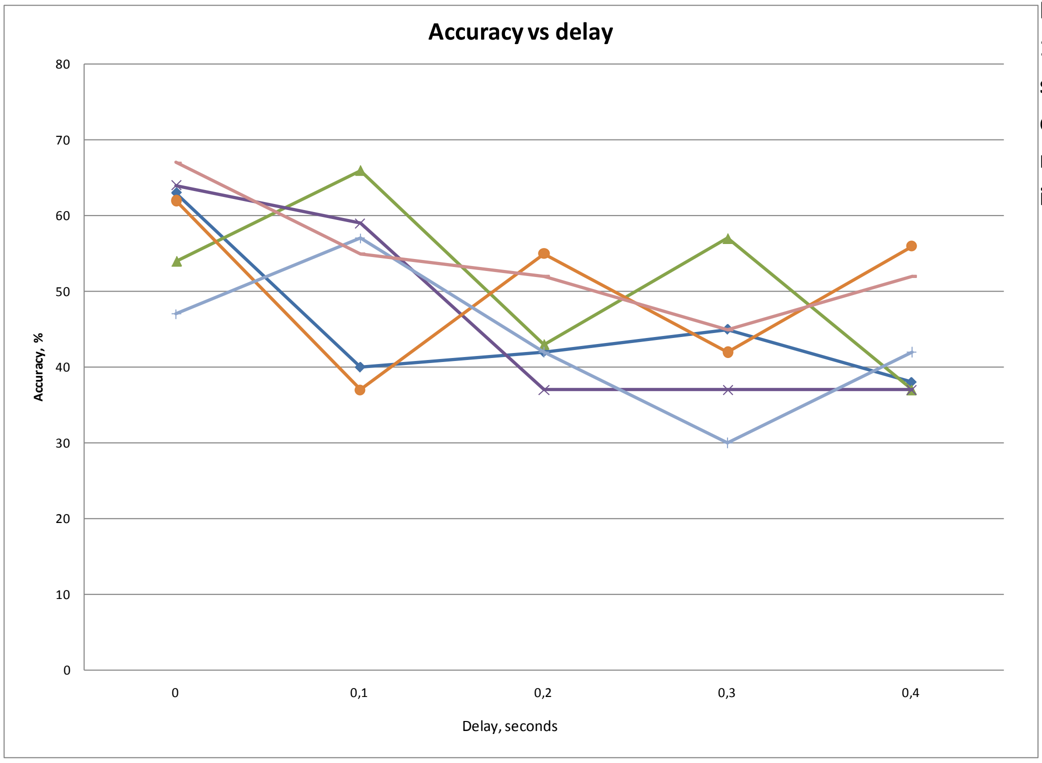


Figure 17: Test 1 data set that show a slightly decreasing accuracy as the delay increases.

6.2. TEST 2

One thing that became apparent from the pilot test was that there was too many features that clouded and reduced usable data, and not enough samples in general.

With these two things in mind a second test was done. This time carried out by myself. Now this will obviously be a source of error because I am likely to be biased, consciously or not.

However this was done in an attempt to get a clearer picture of the problem area.

To get a clear picture I gave myself some limitations:

- Only use the machine gun
- Only play on the map without health pick ups
- No using the melee attack

Furthermore a few changes were made to the game such as introducing a function that randomly picks either, no delay, delayed or delay and LIGM, at the start of a session. This way I would not know from the start which version I was playing.
Additionally the GUI element that displayed the exact health value and the damage text effects were removed to further hide the version from me.

Finally the number of lives per session was set back to 10, where in the delayed version the delay would go from 0 to 0.5 seconds in the first 5 lives and then stay there in order to get a larger sample size for the extreme condition.

6.2.1. DATA ANALYSIS AND SOURCES OF ERROR

Looking at the data of this very structured approach there are still issues, even with 120 lives spanning over 2,3 hours of active play on one map with one weapon.

Figure 18 exemplifies one of the major issues with the single player set up in that player performance appear to be almost constant across delay values, represented with the blue line.
However, if we calculate the hypothetical amount of boost provided by the LIGM from the data points of the delayed version we get the green line. Now initially there appears to be a large spread, but for the entries at 0.1, 0.2, 0.3 and 0.4 there are only 4 sample values to average over

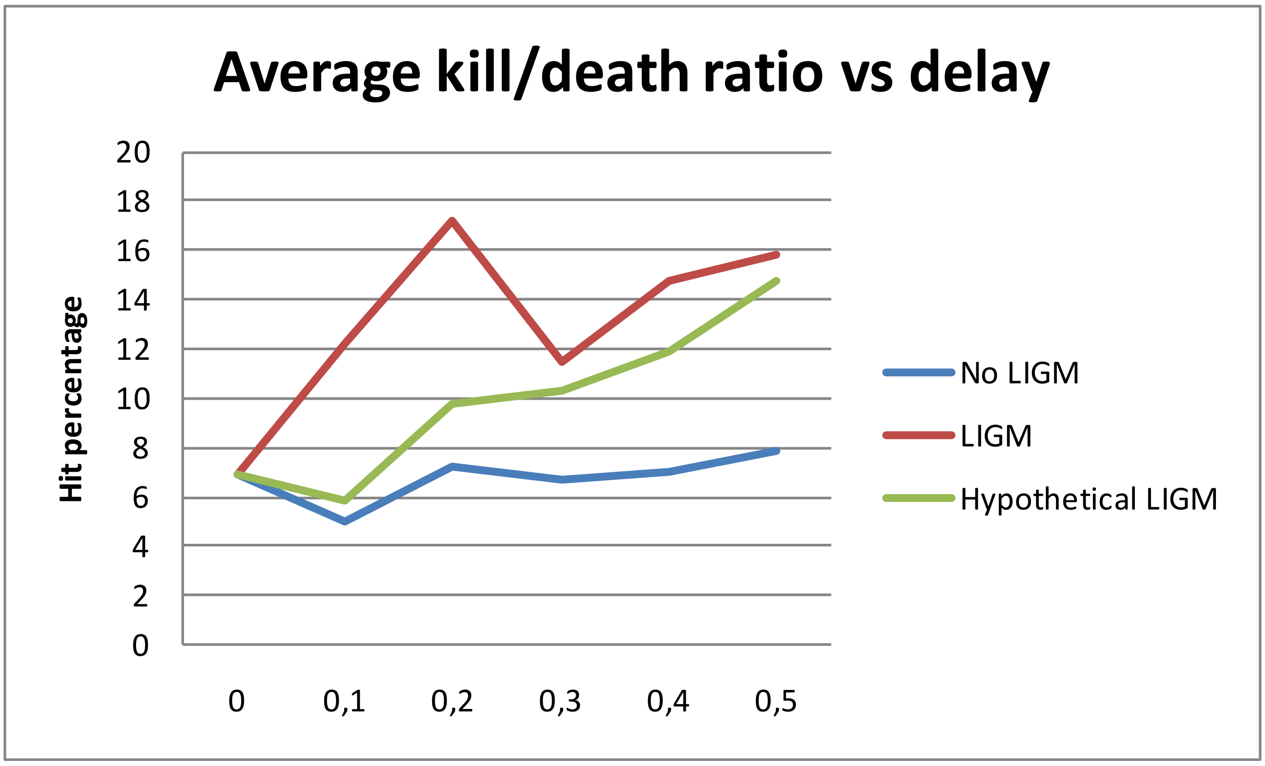


Figure 18: Shows the average kill/death ratio at each delay interval.

for each point, making these points unreliable. This is not the case for the 0.5 seconds of delay, this entry has 20 values that has been averaged for both the blue and red line. Comparing the green and red lines at 0.5 seconds of delay, they appear very similar. If this is no coincidence it goes to show that the boosting the player’s damage and health by X% does indeed produce a player’s performance output boost by X%. This may appear trivial, but it was never certain that such a great advantage in both health and damage output would not make the player even more powerful than each component alone.

In spite of this indication, it appears that the prototype was not capable of reproducing the tendencies seen in other studies. There may be several reasons for this. The most obvious villain is that the delaying of weapon fire and reload is nowhere enough to replicate the conditions on real latency.
Another large source of error apparently also lies in player’s evolving their skill.

As seen on 19, I evolve my skill and way of playing the game throughout these 120 lives, or 2,3 hours of play time. Both pictures represent a single play session of 10 lives in the same setting. The only difference is when these games were played. The green lines represent the second game I played and the red represent the 12th and final game I played.
This is clearly an influencing factor on the data.

With player familiarization such an important factor it should be considered using computer controlled players to evaluate hypothesis 1 because they can be designed not to learn anything from one life to the next. Either that or gather a huge amount of human generated data with enough play time for the player to have achieved optimal performance.

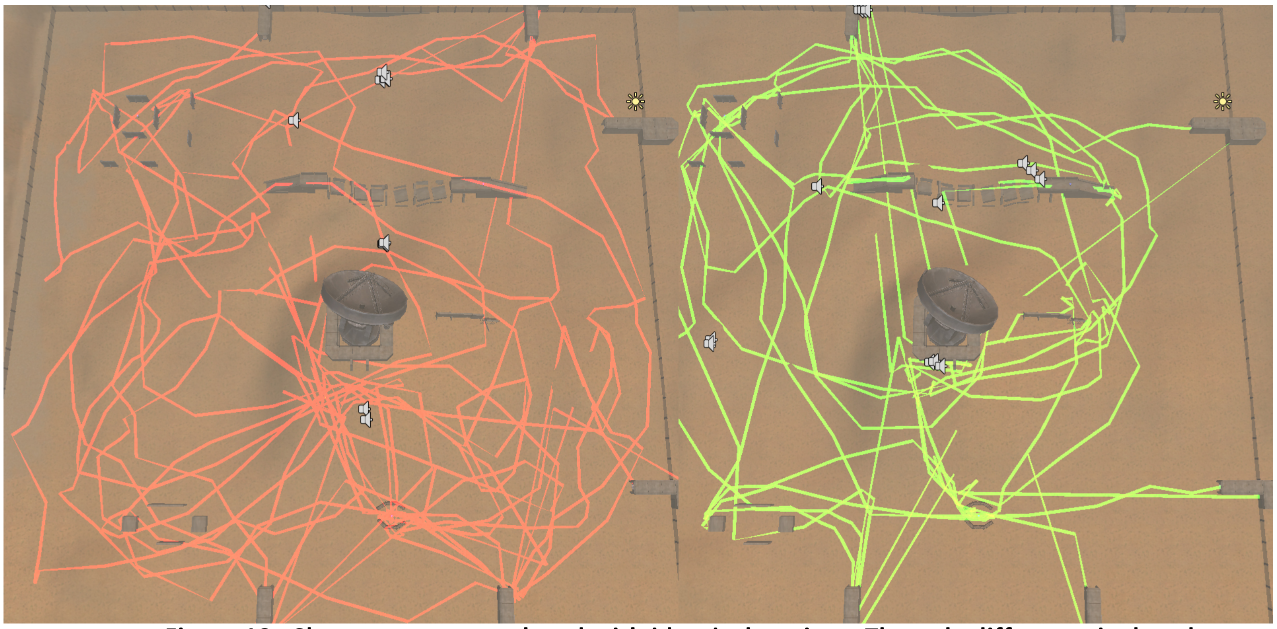


Figure 19: Shows two games played with identical settings. The only difference is that the red lines represent a game played 10 rounds later than the green. This image shows that player training is a clear source of error.

Even with familiarization an important factor, there is still a notable drop in accuracy, as the following table and Figure 20 show.

Delay	0	0.1	0.2	0.3	0.4	0.5
Average hit ratio	38.8	36.5	33.7	32.9	32.8	31.8

Even though it is hard to conclude that there is drop in performance with induced input delay, there is some evidence in the accuracy drop shown in 20. At a half second delay there is a drop of 18% accuracy from 38.8 to 31.8, which in theory implies that the player is inflicting 18% less damage.

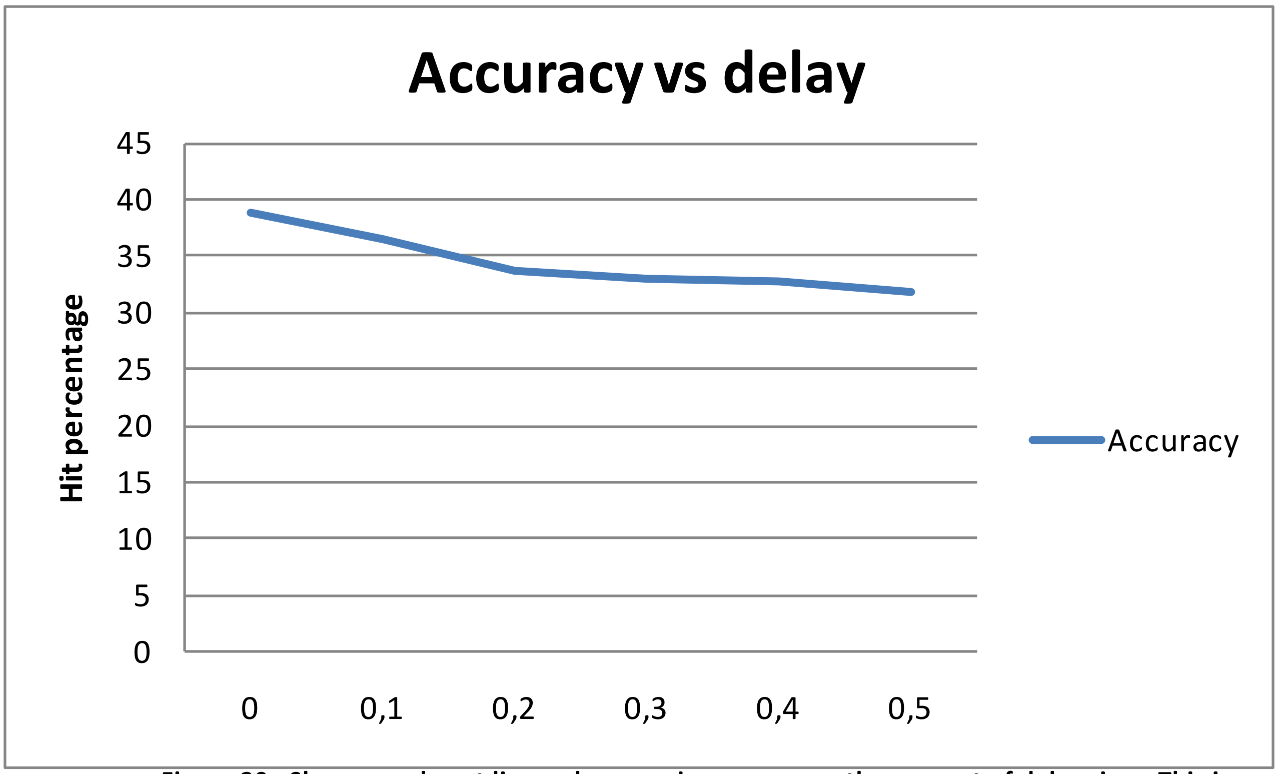


Figure 20: Shows an almost linear decrease in accuracy as the amount of delay rises. This is in spite of the heightened player performance which seems like a paradox.

6.3. SUMMERY

Two tests were carried out. The first of them was a pilot test where 4 boys helped iron out the procedure and game set up for an actual test setting. The test yielded very little useful data other than indicating that the prototype is stable and fit for further development.

A second test was done by myself in order to see if simplifying the game elements and let the game pick the test method at random would provide clearer data indications. Nearly no indications in favor of hypothesis 1 was uncovered, only a hint that the boosting of a few core parameters gave the assumed output improvement, but whether this holds true is far from certain.

7. Evaluation

The evaluation of this project will be done in 3 parts. First the project as a whole will be discussed. It will then be concluded in relation to the project goals, determining if the project has been a success or not, and to what degree.
Finally the project will be put in to context, and possibilities and priorities for future work will be discussed as well as real world application.

7.1. DISCUSSION

The first iteration should perhaps have focused more intensely on hypothesis 1. Focusing on adjusting the players parameters and see the impact of it. Not attempting to also measure player behavior change as a function of increasing latency, and whether it should be a contributing factor in the balancing adjustments.
It may have been too ambitious even if it was a two man project, and should have been left for future work once the basis was proved or disproved. However it was of personal interest and thus served as a motivating factor to drive the project forward. What can be learned from this is that it is wise to have only one focus per iteration or mile stone.

I will not use the loss of a team member as an excuse, for there are still issues with the project as it is now. The test design was too naive and it took two separate tests to acknowledge the amount of data needed to say anything conclusive.
Furthermore a series of other choices could have been made if the hypotheses were indeed the ultimate goal of this project. Choices such as using existing games and attempt to modify them to the use the LIGM scaling. Alternatively a very rough prototype could have been made using grey boxes and spheres as stand in for all the assets to test the cold hard data and nothing else, thus saving countless work hours. By doing so the project might have come further, but the passion and real focus of both of the groups original members was to make a solid network game prototype. Attempting to deny that fact will gain nothing.

Losing a team member in the last stretch of the process has taken its toll. The amount of frustration caused by the sacrifices that had to be made to the overall quality of the project and the game's code and look as well as missing features and proper time to test it has been very demotivating. The motivational issue was worst towards the end, and the written part has evidently suffered as a result.
However in the end, disadvantaged or not, hypotheses validated or not, I am personally satisfied with the outcome of this project.
During the course of this project I have derived a set of hypotheses, based on previous research, and attempted to explain the reasoning behind the design choices needed to evaluate them.
Furthermore I have achieved a working and stable FPS game platform, with all custom made assets including models, textures, animations, sounds and effects.
Additionally the game is able to output a file containing a number of statistics about a player's performance and choices during game play, as well as display his movement afterwards, which has proven to be a great tool for evaluating level design.

The prototype, while not network ready, did succeed in giving players challenging game play in form of the hastily constructed bots. Making the bots spawn all over the map meant that players could hardly find any static spot and exploit it. The bots also proved reasonable in navigating the landscape and putting up a decent fight.

One thing that is apparent after the first test sessions is that the game should be scaled back in complexity to more clearly evaluate the core of the first hypothesis, however there is a dilemma by removing content and abilities in that it detracts from the game experience, which is already limited in this very basic version.
This is assuming that human players will be used in further testing.
Given the lessons learned that players evolve and constantly improve themselves or do things differently, it should be considered using bots only for testing purposes of hypothesis 1. Still network would have to be implemented so that the bots will suffer the same conditions that humans do when playing online.

7.2. CONCLUSION

Concluding on the two goals set for this iteration:

- Build a platform that can facilitate testing of the hypotheses.
- Evaluate hypothesis 1.

A working and stable FPS game was made that feature most of the design requirements set forth by previous work and the design chapter. During the testing phases of the project the game proved itself fit for further development by finding few game breaking bugs. That being said, a lot of work is still left to give the game the “finish” intended for the project.

However, the game, in its current state is perhaps not capable of evaluating the hypotheses as it does not have network support, which is thought to be the reason why the test sessions were unable to replicate similar data that other studies have found.

Evaluating the first hypothesis yielded no conclusive results, only hints and subtle indications at best.

In conclusion, the project fails to meet neither of the goals entirely.

However the project is far along the road to meet the first requirement. After the loss of a group member, there was no choice other than to attempt to save the project by crudely trying to imitate network conditions and create bots and thereby attempt to meet the first goal. Looking at the situation like that, the first goal could perhaps be considered a part success for this iteration.

While both tests failed to show any conclusive drop in player performance, there was indications that a player’s accuracy was affected, and thus the damage output of the player must be lower, or at least more ammunition is wasted which with more samples might start to show.

In all cases the LIGM enabled players performed better than the two controls (delayed and not delayed input). There is some indication to the assumption that boosting core parameters of the player will yield the intended performance boost. However more data is needed to back this up.

Ultimately it was learned that to say anything conclusive a huge amount of data is required. So much so that evaluating hypothesis 1 might best be done by further developing bots and carry out the experiments on them. However for hypothesis 2 and especially 3 human interaction is needed.
In the end on thing is certain above all else, to proceed and gain any knowledge from this, a network version is needed. First step in a network version will be to confirm that it will suffer from the same issues as numerous studies show for other games.

7.3. PERSPECTIVES

It is apparent that in order to proceed on the road of investigating the hypotheses it will require more development of the prototype, making it network ready as neither of the conducted tests succeeded in replicating the performance drop seen in other studies.

When this requirement is met can the LIGM hypotheses be evaluated, starting with hypothesis 1 that in the last test gave some indication of performing as intended.

In terms of evaluating the other two hypotheses a new method of testing must be found so that the end goal is acquiring large quantities of data from a few individuals so that behavior change may be studied closely.

Assuming that one or all of the hypotheses are validated, the LIGM method can be used by game designers to improve quality of service in allowing geographically disadvantaged players to participate in a game on equal terms with everyone else.

Furthermore it might aid in assuring quality of service of indie games, by using this relatively simple technique in stead of some of the more technically demanding network compensation methods.

Finally it is naive to assume that in their current form, the hypotheses will be the ultimate balancing system for online play. The prospect of adding power to the player will no doubt attract cheating individuals and this is something to be addressed in future work.
It does however feel reasonable to assume that if the hypotheses are successful, that they might be used in some kind of balancing or score system that will help achieve fairness.

8. Appendix

8.1. REFERENCES

References used in this project is organized into appropriate secions and listed according to the Harvard referencing method. All references can be found in .pdf versions on the DVD

8.1.1. PAPERS

[1] T. Beigbeder, R. Coughlan, C. Lusher, J. Plunkett, E. Agu, M. Claypool.
The Effects of Loss and Latency on User Performance in Unreal Tournament 2003.
2004.

[2] M. Claypool.
The effect of latency on user performance in Real-Time Strategy games.
2005.

[3] P. Quax, P. Monsieurs, W. Lamotte, D. De Vleeschauwer, N. Degrande.
Objective and Subjective Evaluation of the Influence of Small Amounts of Delay and Jitter on a Recent First Person Shooter Game.
2004.

[4] T. Henderson.
Latency and User Behaviour on a Multiplayer Game Server.
2001.

[5] M. Claypool, K. Claypool.
Latency and Player Actions in Online Games.
2006.

[6] W. Palant, C. Griwodz, P. Halvorsen.
Consistency requirements in Multiplayer Online Games.
2006.

[7] G. J. Armitage.
An Experimental Estimation of Latency Sensitivity In Multiplayer Quake 3.
2003.

[8] S. Zander, I. Leeder, G. Armitage.
Achieving Fairness in Multiplayer Network Games through Automated Latency Balancing.
2005.

[9] L. Pantel, L. C. Wolf.
On the Impact of Delay on Real-Time Multiplayer Games.
2002.

[10] M. Claypool, K. Claypool.
Latency Can Kill: Precision and Deadline in Online Games.
2010.

8.1.2. WEB SITES AND RESOURCES

[11] http://www.realitymod.com/forum/f11-off-topic-discussion/94878-lag-gaming-behaviour.html

[12] http://info.iet.unipi.it/~luigi/dummynet/

[13] http://snad.ncsl.nist.gov/nistnet/