Biocybernetic Closed-loop System to Improve Engagement in Video Games using Electroencephalography

by

Stefan Klaassen

A thesis submitted in partial requirements for the Masters degree in

Computational Science

The Office of Graduate Studies

Laurentian University

Sudbury, Ontario, Canada

Abstract

The purpose of this paper was to determine the level of engagement with a specific stimuli while playing video games. The modern video game industry has a large and wide audience and is therefore becoming more popular and accessible to the public. The interactions and rewards offered in video games are a key to keep player engagement high. Understanding the player's brain and how it reacts to different type of stimuli would help to continue improving games and advance the industry into a new era. Although studying human engagement had started many years ago, the application of measuring it in video game players has only been applied more recently and is still an evolving field of research. This thesis will be taking an objective approach by measuring engagement through electroencephalogram (EEG) readings and seeing if it will help improve current dynamic difficulty adjustment (DDA) systems for video games leading to more engaging and entertaining games. Although statistically significant findings were not found in this experiment, the technique for future experiments were laid out in the form of classifiers comparison and program layouts.

Keywords

EEG, DDA, Video Games, Engagement, Flow, BCI, HCI, behavioral sciences, Unity, Python, C#

Acknowledgments

Dr. Grewal, Dr. Dotta, Parents, CHIL, David Vallieres, Stephane Horne

THESIS DEFENCE COMMITTEE/COMITÉ DE SOUTENANCE DE THÈSE
Laurentian Université/Université Laurentienne
Office of Graduate Studies/Bureau des études supérieures

Title of Thesis
Titre de la thèse: Biocybernetic Closed-loop System to Improve Engagement in Video Games using Electroencephalography

Name of Candidate
Nom du candidat: Klaassen, Stefan

Degree
Diplôme: Master of Science

Department/Program
Département/Programme: Computational Sciences

Date of Defence
Date de la soutenance: January 06, 2022

APPROVED/APPROUVÉ

Thesis Examiners/Examinateurs de thèse:

Dr. Ratvinder Grewal
(Supervisor/Directeur(trice) de thèse)

Dr. Blake Dotta
(Committee member/Membre du comité)

Dr. Kalpdrum Passi
(Committee member/Membre du comité)

Dr. Aniket Mahanti
(External Examiner/Examinateur externe)

Approved for the Office of Graduate Studies
Approuvé pour le Bureau des études supérieures
Tammy Eger, PhD
Vice-President Research (Office of Graduate Studies)
Vice-rectrice à la recherche (Bureau des études supérieures)
Laurentian University / Université Laurentienne

ACCESSIBILITY CLAUSE AND PERMISSION TO USE

I, **Stefan Klaassen**, hereby grant to Laurentian University and/or its agents the non-exclusive license to archive and make accessible my thesis, dissertation, or project report in whole or in part in all forms of media, now or for the duration of my copyright ownership. I retain all other ownership rights to the copyright of the thesis, dissertation or project report. I also reserve the right to use in future works (such as articles or books) all or part of this thesis, dissertation, or project report. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that this copy is being made available in this form by the authority of the copyright owner solely for the purpose of private study and research and may not be copied or reproduced except as permitted by the copyright laws without written authority from the copyright owner.

Table of Contents

Table of Contents

List of Figures

List of Equations

List of Tables

1 Introduction

This thesis will look at testing the usage of video games, electroencephalography, and classification algorithms as a biocybernetic system all in conjunction to form a more positive and engaging experience while playing a video game. The introduction will look at each of these areas and see how they could be merged to create such a result. It will also include the author's motivation to create such a system, which includes two main objectives: finding a classification algorithm that was able to work in real time on brain data, and testing to see if the game would be more enjoyable with the help of the classification algorithm. Additionally, it will cite some similar research experiments that helped the creation of this experiment.

1.1 Video Games

Video games have been around since 1958 when the first video game was created called Tennis for Two [30]. Computers at this point were very large and expensive. The performance available back then was also very limited. The computer used to make and play the first game was analogue instead of the well known digital systems that are common today. Nowadays we have devices that are much more powerful and relatively inexpensive [38]. Because of this, video games have become more accessible to the public. Resultantly the videogame industry has been growing more and more every year. The growth can be found in not only the revenue of the video game industry [31] but also in its player base as more people are playing video games. Games are popular not only because of their accessibility, but also because of the interaction with the player. The interaction is a cognitive effort and results in an intrinsic feeling of reward

from putting such mental effort into the games' stimuli. Because of this positive reward feeling, some people enjoy spending their time playing games. A problem that has been published in numerous papers and is the basis of research for many video game companies is how to increase such intrinsic reward that is felt while playing such a game. Mihály Csíkszentmihályi has developed a model that has been used extensively in the game design industry called "the Flow model." [42] What this model tries to describe is how a player is most engaged with a game when they are in a Flow state. In the Flow model, as the author describes, the player is in the "sweet spot" between the difficulty of the game and the player's skill level. If a game is too hard and the player skill is too low then the player's will get frustrated and will be taken out of their Flow state. Similarly, on the other side of the spectrum, if the game is too easy compared to the player's skill, then the player will get bored. A bored player is more likely to then stop trying [10]. More details pertaining to Flow can be found in section 2.5.1 .

1.2 Biofeedback

Biofeedback is a biocybernetic technique where the use of indirect biosignatures is fed back into a computer. Some examples of biofeedback measuring devices are electroencephalography, electrocardiography, electromyography, magnetoencephalography, electrodermal activity, functional magnetic resonance imaging, electrocorticography, eye tracking, implicit-association testing, and behavior tracking. When trying to pick what kind of tool that should be used for an experiment, there are a few questions that must be asked, such as: what is the cost to undertake an experiment? What areas of the brain are needed to be measured? How fast do certain measurements need to be? How much computing power is needed or have available? How invasive does the experiment need to be? How will the data be used? The reason

for this is because each biofeedback device has its pros and cons. Biofeedback devices give the computer the ability to detect what state a user is in while they are interacting with something. What makes biofeedback devices such a powerful support for videogames is that they are able to make a feedback loop so the game can make modifications based on the user's reaction to the game.

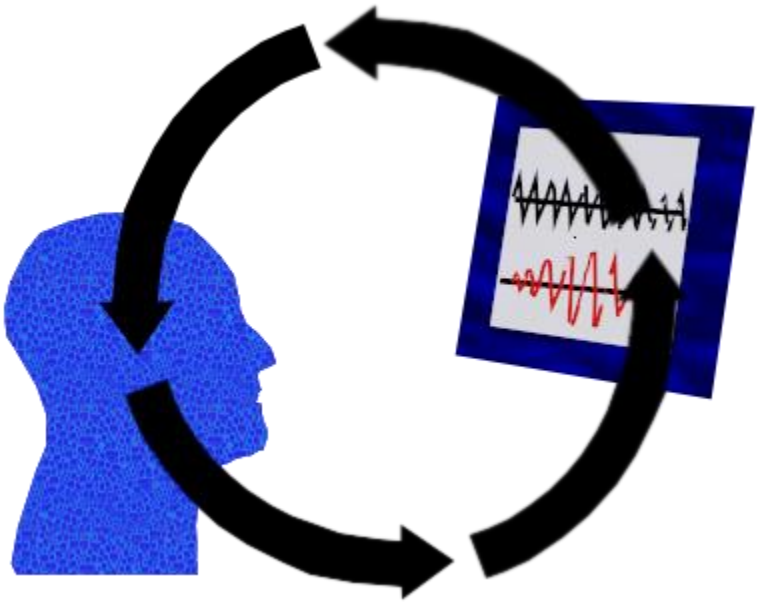

Figure 1: Example of biofeedback

1.3 Dunjions

The video game used for this experiment is called Dunjions. The first version (V1) was made by David Vallieres [49], a Laurentian alumni. A second version (V2) was developed by Stephane Horne a couple years later. The core game comes from another game called Dojies which was a game that won first place at the Sudbury Game Design Challenge in 2016, now called Northern Game Design Challenge. The game was made using the personal version of Unity 3D game engine. The game has two windows displayed. The first window is for the researcher and second is for the player. The reason for there being two windows is so that

information can be monitored while the game is being played. In V2 of the game, the researcher screen was used to conduct an experiment where the researcher manually changed certain elements of the game while it was being played. The game is primarily played using an Xbox 360 or Xbox one controller but does require the use of a mouse to start. An Xbox controller was used because research shows that players' preference in input device is measurable through EEG signals [3]. By choosing a device that is widely used on both console and PC devices, there would be a higher chance that the participants would be used to playing games with this input device.

The game is primarily programmed in C#. Because of this, the game can do many complicated things without extensive programming. Some examples of things C# has allowed to easily do is file I/O, networking, multi-thread processing, efficient memory usage, and quick compiling.

Dunjions is a top-down hack and slash dungeon crawler. Top-down is a term that means that the camera is looking straight down at the player. Hack and slash is a genre of games that the primary mode of interaction is through a sword. Dungeon crawler is another video game genre that means that there is dungeon-like environment. The main character of the game is a knight. The knight has two stats that are shown to the player. These two stats are health and stamina. The health starts at 100 and once it reaches 0, the knight dies. Once the knight dies, there is a respawn timer of five seconds that starts and once it reaches 0 the wave restarts from the beginning and the knight starts with full health again at the center of the room. The stamina stat determines how much damage the player can do and is used up by doing actions such as attacking and dashing. Attacking is done by pressing the right trigger. Unless the player is controlling the direction the knight is facing using the right analogue stick, the attack will be in the direction they are moving

in. The knight has two ways of moving about the environment. The first is by walking. The character can walk in any direction on the 2D axis. This movement is achieved by using the left analogue stick and pointing in the direction of travel. Another ability that the knight has is another movement ability called the dash. The knight is able to get a boost in speed in the direction of the movement analogue stick by pressing the right bumper.

2 Literature Review

Given the complexity of the thesis project, many different areas of study were taken into account. The disciplines that were reviewed range from computer science to neuroscience to behavioral sciences. The topics that will be covered in this section will include game engines, video game difficulty, the brain, engagement, biofeedback in video games, EEG patterns signal processing and signal classification. All these topics will be reviewed to current research that is conducted in their respective fields.

2.1 Game Engines

Programs designed to help programmers make other programs have been increasingly used by game developers. This is a method that has led to higher-level languages. One such program that is very popular is the use of an operating system. Such a system allows programmers to think more abstractly about what they are making and allows them to make more complicated programs with less time or funding. Game engines are another example of such a program. Game engines vary in complexity but their designers try to solve a common problem: how do we make the process of creating a video game easier?

The Unity 3D engine was used for this study. The following will explain how it has given programmers the ability to make a game more feasible for such a study. Unity was first released in 2005. Back then the Unity team was imagining an easy-to-use system with professional tools. Since its official release, the team has been releasing updates constantly to improve its engine. Nowadays, it’s a multiplatform program that has been used for a multitude of games with over a million users around the world. The program allows users to design their game using popular

languages such as javascript and C#. Other tools that can be found in the engine are animation systems, 3D rendering, VR/AR support, efficient physics/collision detection, rapid compiling and real-time script state debugging, 3D sound, and others [43]. Another feature of game engines and Unity in particular is the user community behind it. Whether someone is a beginner and needs lessons, something in their code is not working properly, or there is a tool missing, there is someone else on the Unity forums that can help. A feature that the Unity team added to enable easier asset distribution is their own asset store where someone can either share for free or sell their work to other developers. Therefore, whether one is a programmer, behavioral scientist or even a mathematician, help with Unity is given to those that need assistance.

2.2 Video Game Difficulty

Video games tend to differ from each other with respect to their complexity and structure. Given the wide variety of video games, it can be difficult to assess what makes games easy or hard to play. Another added complication is that the term difficulty varies from player to player. For this reason, it is important to try to break down how to properly define what difficult means before we can measure it. A recent paper by M. V. Aponte et al [7] investigated such a problem and proposed a model to describe difficulty in videogames. Their definition consists of three categories of difficulty, namely:

1) Sensitive
2) Logical
3) Motor

Sensitive difficulty according to the research team is where location-based complications occur. Such a difficulty, as an example, would arise when trying to find something on a map. Logical

difficulty is said to arise when complex inferences are required by the player. This is something that is especially applicable to puzzle games. The last, motor, is a difficulty that arises from the interaction between time and space. An example that most games have of this type of difficulty is the controller with which they are playing the game, where they need to perform a specific combination of buttons or tweaks on an analogue stick while only having a short period of time to perform it. The three difficulty categories are then applied to what is called a core challenge in which the event is undertaken in as small a portion in time as possible. Examples from their experiment of core challenges, using a first person shooter, was to take cover from enemy fire and another being to aim and shoot back. These two types of challenges would be grouped together into a composite challenge. It is important to note that when you are trying to measure a game player's skills and abilities, they should be rated on the core challenges. Their ability to complete composite challenges can be extrapolated from their abilities to complete core challenges.

2.2.1 Difficulty Adjustment

There have been many different techniques used to try to deal with the game difficulty vs player skill problem. The simplest way would be to set game parameters to fit the average players abilities. This would be done through play testing until the desired progression across all play testers is as even as possible or suits the majority of skill levels. Another technique used is to have multiple settings of difficulties. An example of this would be to have an easy, medium and hard mode for either a part of or the whole game. This way if a player has low skills they can pick the easy setting before starting the game and not feel frustrated. A more skilled player would get bored trying to play on the difficulty level that the average player base should have.

This paper uses another approach called Dynamic Difficulty Adjustment (DDA) system. What this system tries to do is to measure how well a player is doing and to modify the difficulty level so as to keep it balanced with the player's skill level. A DDA system can make any number of changes depending on the game, such as the intelligence/parameters of enemies or the complexity of puzzles. An example of intelligence of enemies would be its path finding algorithm or dodging abilities. An example of the complexity of puzzles that could be changed by a DDA system is in chess. A bot player could make the depth of their choice or make mistakes every now and then to make the player feel like they are doing better or worse. Since there are many games with many different genres, there are many aspects that can be used by DDA systems to change a game's difficulty. A video game that used a DDA in the form of a "director" is the Left 4 Dead series. [55] The director would monitor a player's accuracy, movement, health, and other criteria to adjust several aspects of the game including zombie spawning locations, ammo, and health packs.

2.3 Brain Physiology

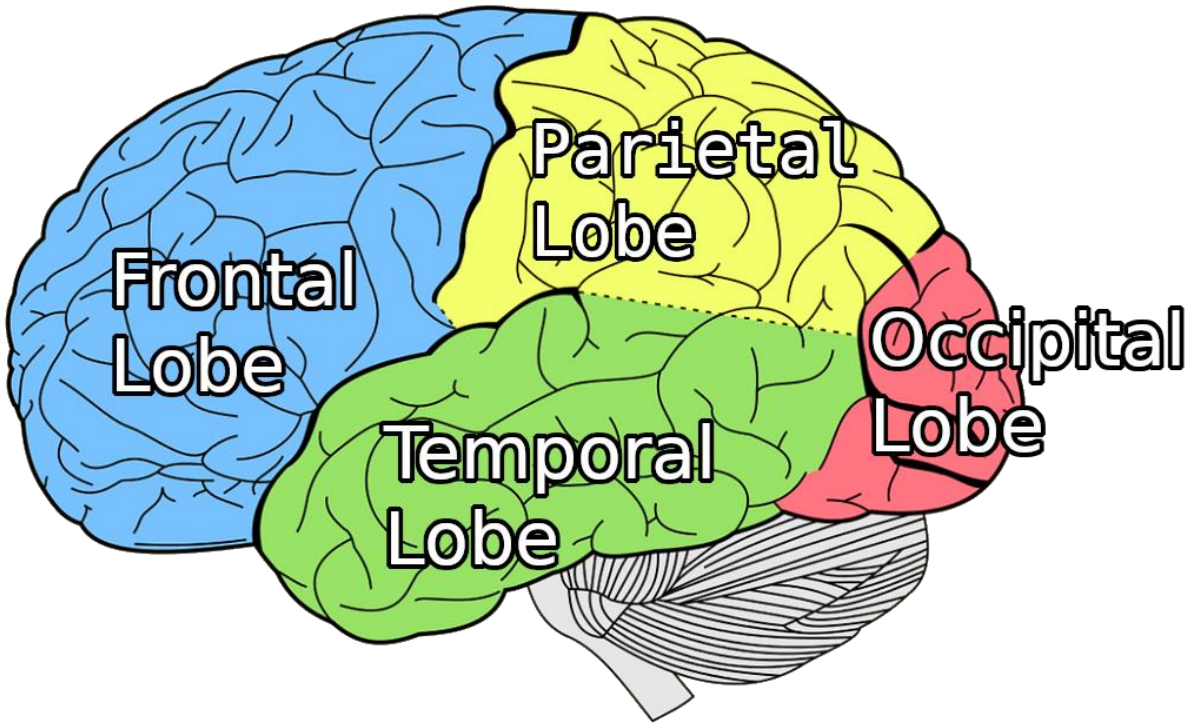

Figure 2: A representation of the brain with different lobes labeled

The human brain, like many other organs, is very complicated. One reason for its complexity stems from its job of processing and controlling most systems in the body. As part of the central nervous system which contains 10 ^12 neurons [28], it is woven in intricate patterns. The brain is a single organ but different parts of the brain typically handle different functions of the body. For this reason it is important to know the different components to know which one is important to the topics in this thesis. The cerebrum, or the part of the brain that is associated with higher-level thoughts, is split into four main lobes. These lobes are the frontal lobe, temporal lobe, parietal lobe, and the occipital lobe. According to P. Michael Conn in Neuroscience in medicine [44], The frontal lobe has been found to be related to problem solving, emotions,

speech and intelligence. The temporal lobe has been measured to be related to hearing, high level visual decoding and memory. The parietal lobe has been found to be related to senses and their interpretations, language functions, and visual/ spatial perception. Lastly, the occipital lobe has been shown to correlate with visual perception. These are the most relevant of the many important functions each of these aspects of the brain is equipped to handle.

The cerebrum can be further broken down into the left and right hemispheres. There are many functions that can be handled by either side of the brain. Although they might share the same name each side handles things slightly differently. two such function tested in the R.E. Wheeler study [22] are hand preference in writing and emotions. The study tested the effect of positive and negative experiences from a stimulation and the way that it activates the frontal cortex on either side differently. This is also known as asymmetrical activation.

2.3.1 Brain Arousal

Generally, the more a person thinks, the more their neurons are active. The important distinction here is that the firing of neurons is not only defined in intensity but also in intensity in certain frequency [28]. People will have a different amount of arousal levels even on the same stimuli. Some of the more common reasons for the variance are:

- Some people put in more cognitive effort than others
- Some people are predisposed to a disorder such as epilepsy [28]
- Other research has shown that lacking in arousal or reduced arousal can be linked to defects such as mental illnesses [27] and depression [22].

2.4 EEG

Figure 3: An EEG cap representation

Electroencephalography is a biofeedback technique that has been around since 1875. It is a technique that measures the movement of electricity through the brain by measuring voltage potential differences, and is generally associated with measuring the electrical potential fluctuations of neurons firing in the brain. A simple rendition of the workings of an EEG goes like this [39]: A metallic conductor(electrode) touching the skin of one's head detects a very faint voltage change based on the electrical energy that is moving through the neurons through a process called volume conduction. This small voltage change is then sent through an amplifier and measured by a voltmeter. Using this information, one can tell how active a specific area of

the brain is working. With the use of multiple electrodes across the scalp and the knowledge that different areas of the brain are responsible for different aspects of human bodily functions, one can now piece together an indistinct picture of what the brain is thinking. Blurry being an important aspect here, as a single electrode will be picking up the electrical activity of millions, if not more, neurons firing. The aspects of an EEG that make it easy to work with is that it is typically non-invasive and has a high temporal resolution. In the past, a substance needed to be applied to make a proper connection between the scalp and the electrode. This substance is generally a conductive gel. Nowadays, with the invention of accurate dry electrodes, EEGs can be as easy to use as putting on glasses or a hat with only a slight hit towards accuracy.

Figure 4: Muse EEG [53]

EEGs over the years have become less expensive, on average, and have become more viable in the consumer market. According to recent papers, because of this shift, more and more brain computer interface studies are being conducted with EEGs, with 72.5% used in 2014 [11]. Another reason expressed by Gianluca Di Flumeri et al. was that EEGs have also become more portable [40]. With increased computational power in smaller chips due to smaller transistors and more efficient batteries, alongside better wireless technology, portability has been a huge improvement over older EEGs.

2.5 Engagement

An important topic relevant to this thesis is the concept of keeping someone's attention on a stimulus also understood as engagement. The definition of engagement is therefore important. Engagement as it relates to this thesis is “emotional involvement or commitment” [48]. As it relates to video game interactions there is no formal definition of engagement, only theories discussed by psychologists. One such definition was developed for the FUGA project in 2006 [2]. The researchers approached it by determining the important features of video games’ subjective experiences. Their model was built on three research papers that looked at subjective videogame experience. These three research papers are Game Flow [36], Fundamental Components of the Gameplay Experience [35], and the Motivational Pull of Video Games [34]. The final output of this research and experimentation was that engagement is made of 10 different components. These components are competence, flow, suspense, enjoyment, sensory immersion, imaginative immersion, control, negative effect, social connectedness, and social negative experience. Through extensive testing, researchers were able to break down the measurement of engagement through four different questionnaires, one of which is used for this study. These tests are: The Game Experience Questionnaire, In-game Experience Questionnaire, Post-game Experience Questionnaire, and Social Presence Gaming Questionnaire. The questionnaires are a series of questions where the player rates their experience on a scale of 1 to 5 on a series of questions reflective of how it represents their subjective feeling. A version of the In-game Experience Questionnaire can be found in appendix 8.3.

2.5.1 Flow

An important part of engagement and video game design over the years is Flow Theory first introduced by Mihály Csíkszentmihályi in 1975. The theory was conceived in an attempt to explain why some people become so focused on a specific type of stimulus. He was specifically interested in artists and athletes; however the theory is applicable to almost any activity, including playing video games and education.

The theory states that a person is experiencing flow if all six of its requirements are met. These requirements are:

1) A clear set of goals
2) Feedback relating to progress
3) Feelings of control
4) Loss of self consciousness
5) Distortion of time
6) Balance between challenge and skill

Even though there are six individual points, they do intertwine in their functions.

To attain flow, it is imperative that the goals are laid out in such a way that it is clear to a player how to reach the end of the activity. A player may get lost or sidetracked if what they are supposed to be doing is unclear. For this reason, it is important that the person trying to achieve flow is given proper feedback towards reaching the goal as well. Clear goals and good feedback on progress are controllable in the design of video games. Feeling in control of an activity may create a feeling of loss of self consciousness. The reason for this is that if someone feels like they are in control, their attention is on how to complete the activity instead of reflecting on their own abilities. Distortion of time is a byproduct of flow that many people experience while in the flow

zone. It also has its own saying “time flies when you are having fun.” Lastly is the balance of skill and challenge. The reason this part of flow is important for video games is because it is possible for game designers to make games easier and harder, allowing them to adjust the balance of skill and challenge to an engaging level. The below figure shows an example of how one would move through these dimensions and a designer would try to keep the player in the flow channel.

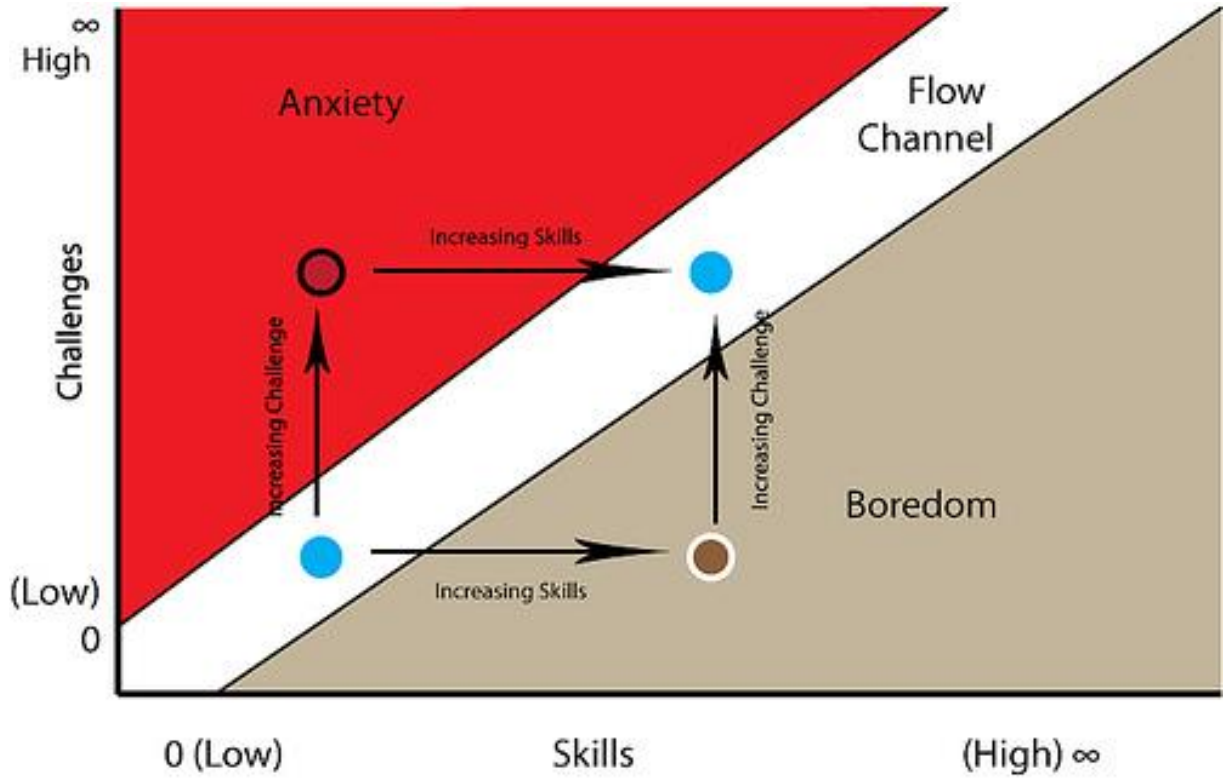

Figure 5: Graph of Flow [42]

There are two things that game designers can directly control when it comes to Flow. The first is the goals of the game and the associated feedback in trying to complete such goals. The other is the balance between challenge and skill. The first is relatively easy in the game design world. Assuming the game is not too complicated, the game designer needs to ensure that the player is aware of what they need to do and to acknowledge that they are doing something

productive toward the end goal. A tactic that is generally implemented to make this easier is the use of a tutorial. This ensures that the player knows what they need to do and how to do it. Secondly, the challenge and skill aspect of flow is more challenging because not all players have the same abilities and therefore there is no single difficulty level that will be suitable for all players. This thesis is specifically focusing on the latter concept.

2.5.2 Effort and Demand

Flow can be used as a definition of a simple point in time but there is another factor that has been tested by K. C. Ewing et al. [10]. The effort vs demand connection is a task load test where a person can keep up with the demand of a game up to a certain point. If the demand gets too large then there is an overload and the user will stop putting in the effort required in order to continue to play the game. Another research paper by G. Chanel at al. [18] looked at testing classification of game difficulty using biosensor data. They concluded that they were getting a large number of their participants classified as easy/ bored even when it was in the hardest mode because they were simply giving up on the game. This experiment seems to support the previous research paper by K. C. Ewing. The approach G. Chanel and his team took to overcome effort and demand overload was to start over with the easiest difficulty when the game got too hard. This way the participants knew that it would be a manageable difficulty and they would start putting effort into the game again.

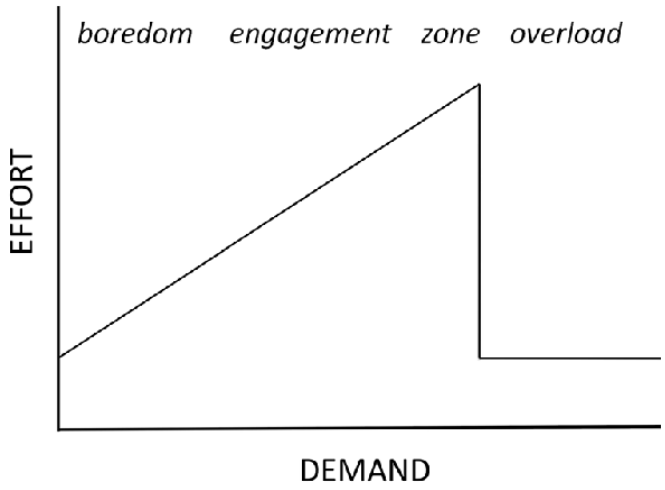

Figure 6: Graph of effort vs demand [1]

2.6 EEG Biofeedback in Video Games

EEG biofeedback devices have been used on consumer video games over at least a decade [37]. Other common biofeedback devices are heart rate monitors and electrodermal response sensors.

Many researchers doing biofeedback studies use more advanced feedback hardware and techniques. Some of the more common devices include EEG, electrocardiograph (ECG), and Galvanic skin response (GSR) [11]. A few more obscure devices that have been used are thermometers, electrocorticography (ECoG), Magnetoencephalography (MEG), functional magnetic resonance imaging (fMRI), accelerometers, and gyroscopes. Other techniques that are not devices that have been used in biofeedback research are eye tracking and gesture recognition

There are two forms of biofeedback systems: active and passive. Active biofeedback is a way of controlling the game with intention. An example of an active biofeedback device would be to control a character's movement with the player's mind using an EEG. Passive biofeedback is a system where a player's biosignatures are measured and fed into a program that modifies

itself without the player necessarily knowing that the game is changing. An example of a passive system would include a program that measures heart rate and then attempts to intimidate the players when they seem to be relaxed [45]. This technique would ensure that the player is constantly engaged with the horror game or help them manage their anxiety.

Another aspect of video games using biofeedback is its ability to change people's behaviour and motivation in a lab type setting. This could lead to higher recruitment numbers for brain-computer interface (BCI) experiments [11] and give researchers a greater ability to test specific elements of human behaviour.

2.7 EEG Patterns

EEGs are a way of looking at the effects of a person's brain activity but cannot tell you exactly what the person is thinking. Researchers first need to convert the voltage potentials measured into data that is readable and understandable. From this need, there have been techniques proposed by researchers to make sense of EEG data for measuring flow in video game players. Two of the most popular techniques are power analysis and phase analysis [11]. Another method called hybrid interfaces is starting to be used in more recent research.

2.7.1 Power Analysis

Power analysis, or event-related potentials, is the measurement of the amplitude of the voltage measured by EEGs in the time domain. This is generally seen as a more simple method but does have many applications. One reason this method is used is because it is much easier for the person being measured to control the amplitude from their own brain waves. With only limited coaching a participant can be trained to lower their amplitude. This form of training has

been used in some research funded by the military to reduce stress in their soldiers [1]. Although there are many ways of using power analysis, some of the more popular methods in video games include P300 and steady-state visually evoked potential (SSVEP) [11].

2.7.2 Phase Analysis

Phase analysis is another approach to interpret what the brain is doing. It does this by breaking down the signal into the frequencies that are measured by the EEG. One method to utilize phase analysis is a technique called Fourier Transformation. This technique will be expanded upon in the section 2.8.1. Once a signal has been broken down into its component frequency, it is averaged into bands. The bands more commonly found in research papers are the Alpha, Beta, Delta, Theta and Gamma bands [11].

Table 1: Frequency Bands of brain

Name of Band	Band Range (Hz) [32]
Theta	4 - 7
Delta	0.5 - 3
Alpha	8 - 13
Beta	14 - 30
Gamma	31 - 50

The activity of each band and where in the brain they were measured from has been found to have correlations to different emotional and physical processes. Some early work using this technique was on sleeping patients where the researchers were able to tell what state of sleep the subjects were in [46]. In more recent studies, especially ones relating to the topic of this

thesis, researchers looked at the application of certain bands at specific locations and how it applies to engagement, stress and emotions [1,3,5,6,8,10,14].

Equation 1: Equations used to calculate engagement

$\frac{Beta}{Alpha + Theta}$	$\frac{Theta}{Alpha}$
Equation 1	Equation 2

Once a signal is converted into bands, some papers [6,8,26] looked at forming equations to try to describe the extent of a players engagement. One such equation used in various research papers is Equation 1. Researchers would average out the oscillatory ranges of Beta, Alpha and Theta across all EEG electrodes and then input them into Equation 1. Researchers also used the same EEG output and input them into Equation 2 but the results showed a higher degree of noise and lower accuracy towards predicting engagement.

Another algorithm that has been mentioned and tested in a few studies [22,23,5,14] is the asymmetric activation of alpha band frequency of the frontal cortex. Through testing, it was found that one alpha band frequency was more active than the other related to a positive or negative experience. More specifically, if the right frontal lobe of the brain is less active than the left then it relates to a negative perception of a stimulus. Other asymmetrical activation theories have been proposed such as measuring the parieto-temporal regions of the brain [23] but none have been tested as much as the frontal asymmetrical activation and attained as statistically significant results.

2.7.3 Hybrid interfaces

A method that has been examined by various researchers is to mix two or more systems together, making a hybrid system [11, 47]. Hybrid systems have been shown to have some benefits over single systems. Some of these benefits include improved accuracy and additional control signals.

2.8 Signal Cleaning

When analysing and measuring real world data, it is often contaminated with noise and interference. Signal processing and cleaning must often be performed on data before it can be properly analyzed. In this section, we will look at two techniques used to clean and to prepare data that have been utilized in many EEG experiments [5,6,10,14,19, 32]. These two techniques are Fourier Transformation and Independent Component Analysis [17,21].

2.8.1 Fourier Transformation

An algorithm that is used quite often in signal processing is the Fourier Transformation. This algorithm allows users to break down a modulating signal into its component frequencies. More specifically it breaks it down into the sinusoidal frequencies. Fourier Transformation functions by mapping a signal over time into its frequency representation. The equation for the finite discrete Fourier Transformation is:

Equation 2: Fast Fourier Complexity

$$X_k = \sum_{n=0}^{N-1} x_n * e^{\frac{-i2\pi kn}{N}}$$

X_k =The frequency domain
x_n =The signal

The problem with the above noted Fourier Transform algorithm is that it is slow to compute with a complexity of $O(n^2)$. When significant amounts of real time signal processing need to be done, computational speeds must keep up with the incoming data. The Fast Fourier Transform (FFT) is an improvement over the traditional Fourier Transform by having a complexity of $O(nlogn)$. The drawback of the FFT algorithm is that the signal needs to be processed with an array size containing a number of elements relating to the powers of 2. The equation for the FFT is the same as the regular Fourier Transformation but it overlaps the calculations to reduce redundancy.

2.8.2 Independent Component Analysis

A noise detection and extraction algorithm is the independent component analysis algorithm (ICA). It is a method of doing blind source separation and has been used to find noise patterns in data. One such pattern that has been found to work well with ICA is noise picked up by an EEG when the eye muscles blink [21]. Such patterns can compromise data and what ICA allows researchers to do is to isolate and then remove the noise while still keeping the underlying and useful data generated by the brain. The algorithm maximises the distance between the

potential components based on how many dimensions/sources there are. In the case of EEGs, the input data would be each electrode around the scalp. The output would be components that create the largest difference across each signal. An example of its application can be explained by means of the cocktail party scenario. The starting point of this scenario is having multiple people speaking and microphones in a room at the same time. The algorithm is then able to isolate the voices of specific people based on their unique voice frequencies and intensities received by the microphones. The ICA algorithm is able to distinguish most of the voices of each of the people who were talking, assuming there were as many people talking as there were microphones.

A paper by T. Jung et al. [21] notes that the ICA algorithm should only be used if there is a limited amount of data. If there is a surplus of data that can be used in the EEG research, one should discard the distorted data.

2.9 Signal Classification

The output coming from the processed EEG data can sometimes be obvious enough to be read with human eyes. If games are using EEG data, the games must be able to classify the brain patterns independent of a human observer. Classification in this case means that the game needs to be able to use the data that is derived from the EEG and make the necessary changes in a timely manner without interrupting gameplay. Another term that is used to describe classifiers is machine learning. Machine learning means that the algorithm finds patterns automatically and without a programmer describing the rules. One requirement of machine learning algorithms is that the computer requires data to decern the patterns through a process called training.

2.9.1 Linear Classifiers

Linear Classifiers function by interpolating the data in all dimensions by a linear function and classifying it accordingly. Two forms of linear functions are Linear Discriminant Analysis (LDA) and Support Vector Machines (SVM). A technique that augments linear classifiers is called kernels.

Linear discriminant analysis

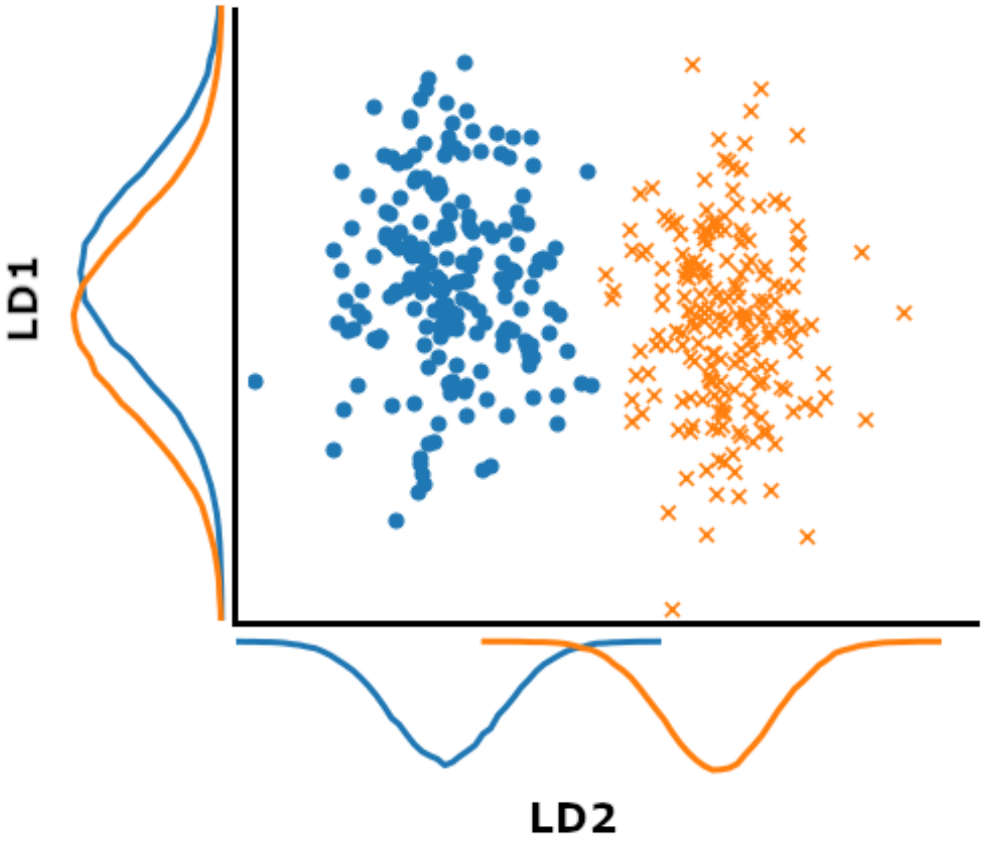

Figure 7: Depiction of the use of an LDA

LDA functions by doing data reduction which gets a projection of the data and then separates it with a hyperplane. Figure 7 shows an example where data reduction was made by a LDA algorithm and can create a line to separate the data on the x axis. It functions by

dimensional reduction by only using the important dimensions that will help classify data, and separating data accordingly. This technique is used for several reasons; It is simple, if the data you are trying to separate is obvious in its distributions it gives an easy way to classify it, and it requires relatively low amounts of computational power. For the sake of this study, it was not a viable option since the data was not as obviously clustered.

Support-Vector Machine

Support Vector Machines (SVM) is a tool that have been used in both binary classifiers and Regression Analysis. SVMs work by creating a hyperplane between two or more classes. The hyperplane is positioned in such a way to maximize the margin between the datapoints. An example of this is if you were trying to classify two different types of fruit based on their weight and diameter. This would be considered 2-dimensional data. The SVM would create a line separating the groups of fruits. Depending on what side of the line(hyperplane), we can classify it as one of the types of fruits. This can be used for as many dimensions as needed. The SVM also has some helpful techniques that make it more powerful such as kernel's which are discussed in the next section.

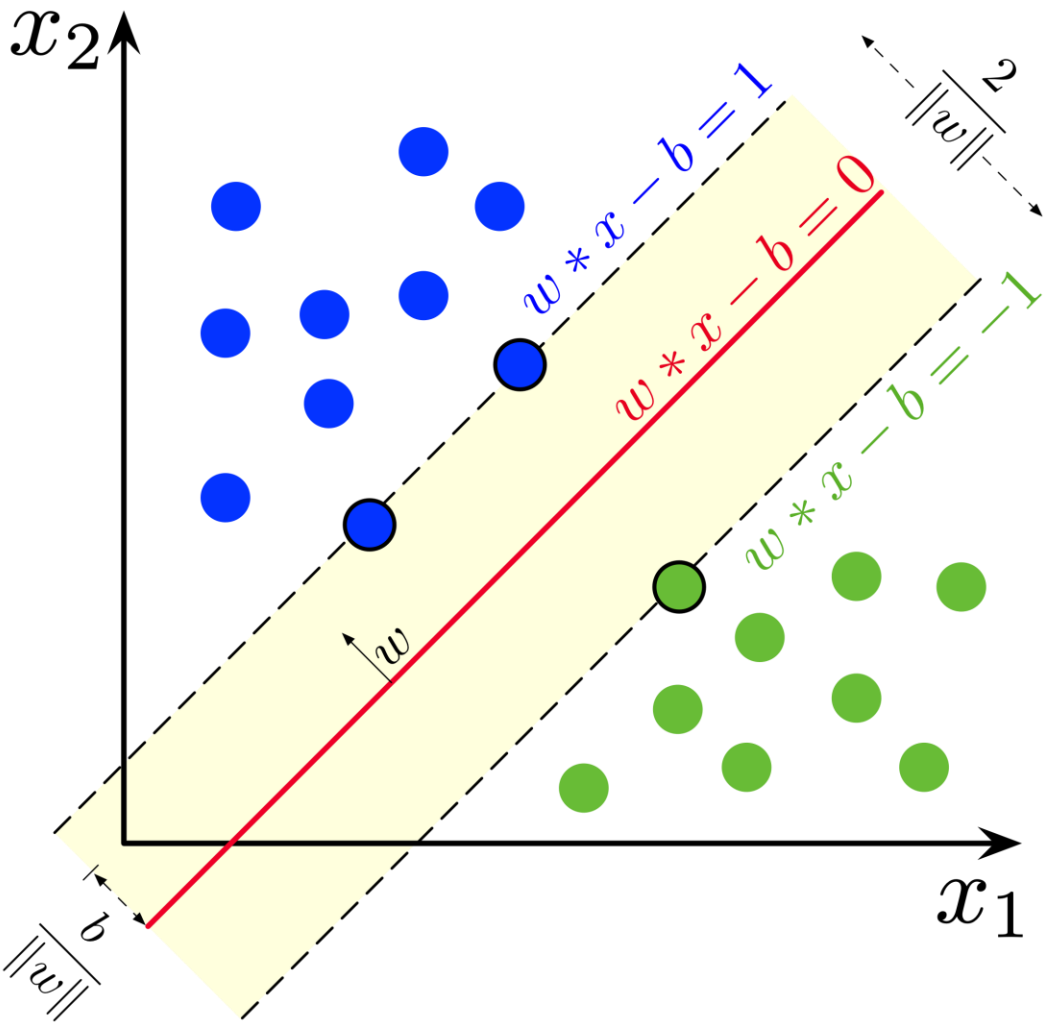

Figure 8: Depiction of an SVM [54]

SVMs are a good option for BCI classification because they are good for generalization [15]. Generalization means that the algorithm does not suffer as much from over training and is therefore quite stable. As more data is added, the output of a training process should not change too much unless the data contains drastic outliers.

In a study that reviewed different classification algorithms [15], kernelized SVM had some of the highest accuracy for synchronous experiments. Some speculation as to why this is when compared to more complicated algorithms is that it is less sensitive to outliers, its

robustness to curse of dimensionality, and its ability to regularize make it less affected by the often-noisy BCI data.

Kernels

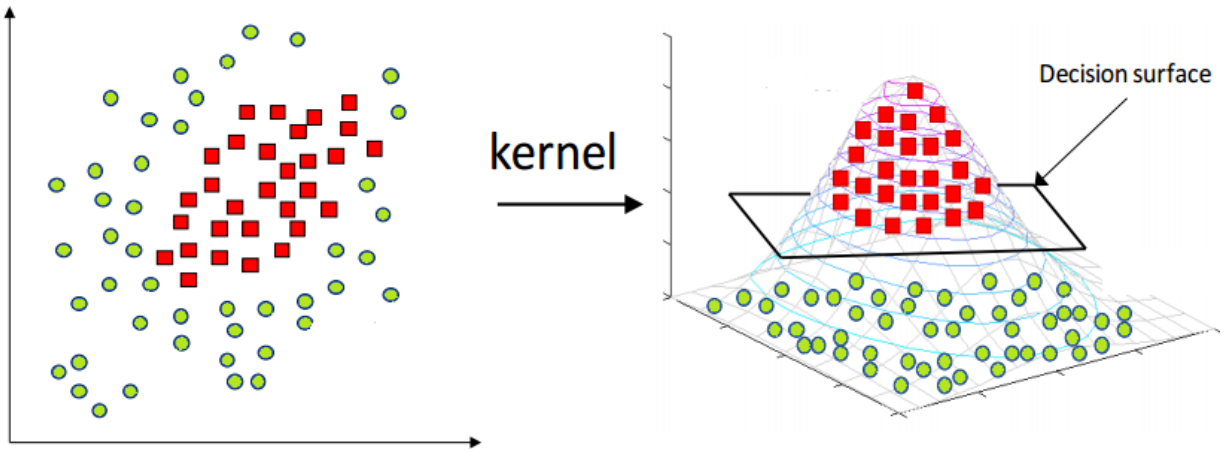

Figure 9: Depiction of a Kernel [51]

Kernels map data from their original form to another form. There are many kernels used to transform data but they all work on the same principle by having a known conversion equation. Conversion equations should be reversible. One way this can be attained is by adding another dimension and making the value of the new dimension a combination of its other dimensions. This would be interpreted as a linear kernel. As shown in Figure 10 another kernel that is popular for SVMS is the radial basis function (RBF) also known as a Gaussian Kernel. This method creates a highest point from which all other values are measured from using inverse euclidean distance. The distance can then be made non linear if needed. Therefore, the closer a point is to the highest points, the larger the calculated kernel value.

2.9.2 Neural Network Classifiers

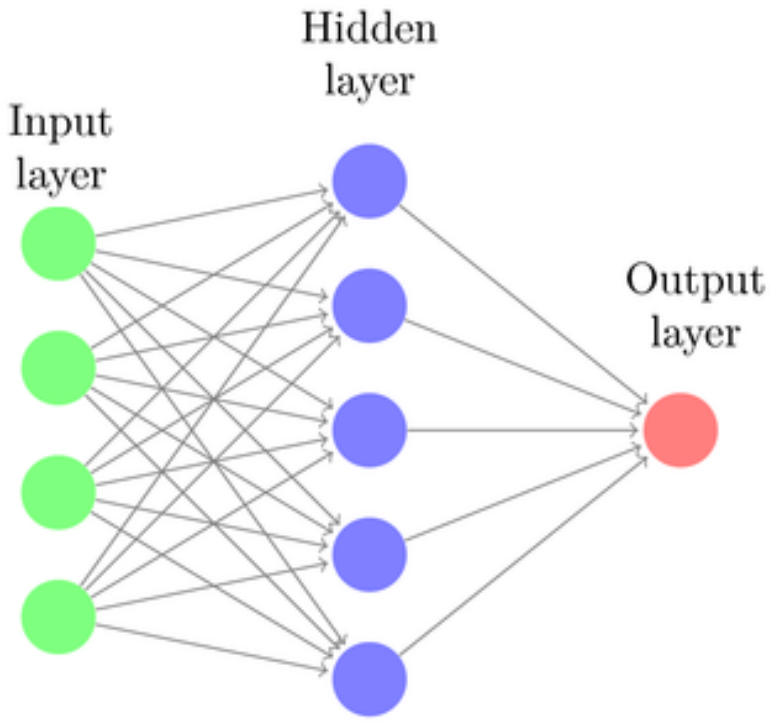

Figure 10: Depiction of an artificial neural network

Neural Networks are typically defined as the numerical representation of neurons in the brain. The "neurons" are generally placed into layers where all of the neurons from one layer have a connection to the next layer. This is shown in Figure 11. Each calculation is normally done layer by layer. Each neuron represents a value that is first multiplied by a weight and then added to its connecting neurons. After all the values are summed, a bias is added. The last calculation that is normally done after one layer is defined as an activation function. If there is more than one hidden layer then a neural network can be considered a deep network. If the output of a layer feeds back to itself or to a previous layer then it can be considered a recurrent Neural Network. One of many learning algorithm for supervised neural networks is called back propagation. It functions by finding the output of a network and then finding the distance also called loss value to a desired output which then provides the direction to modify the weights and

biases of each layer. The modifications of the weights and biases done by back propagation then leads to a lower loss value.

2.9.3 K-Nearest Neighbours

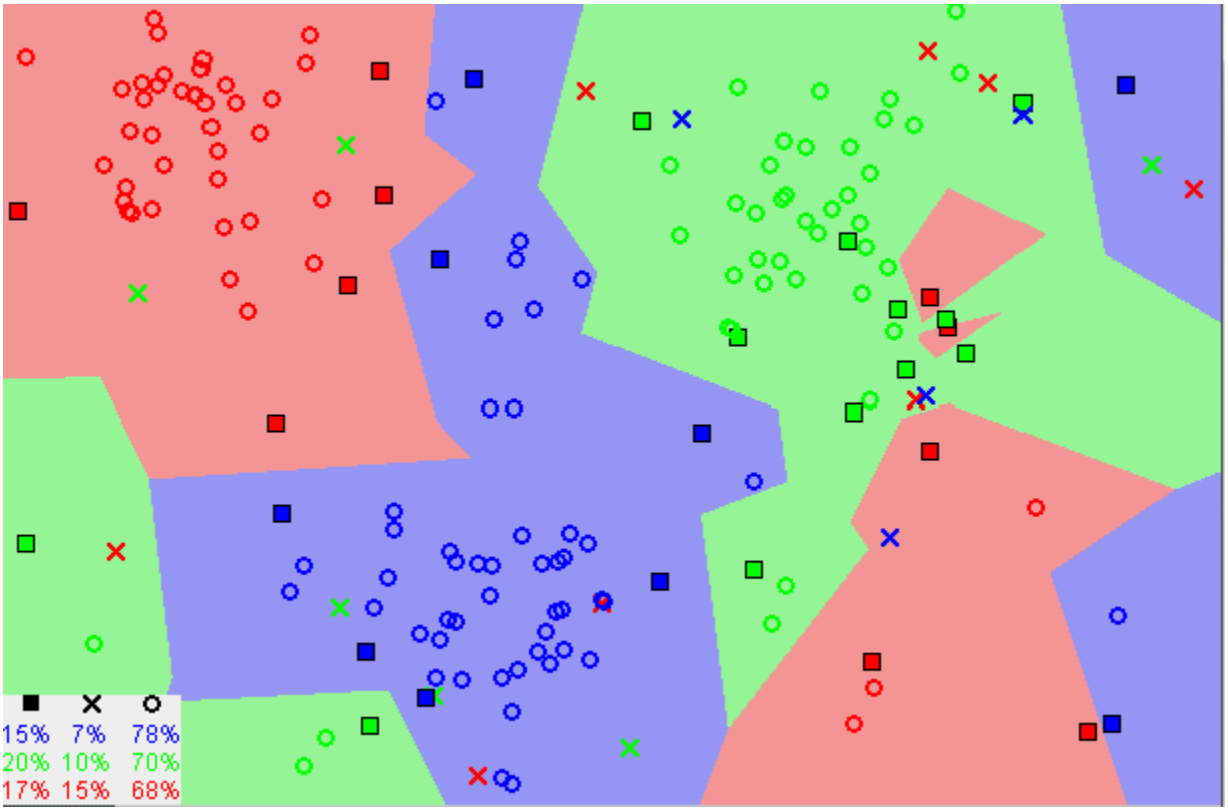

Figure 11: Depiction of the K Nearest Neighbour algorithm [15]

One of the simpler classification algorithms is the K-nearest neighbours. This algorithm works by having points in data space which describes a specific class. Then using a distance function, the algorithm can then determine which class a data point belongs to by looking at which class point it is closest to. The class allocation of the class points are determined by the nearest k points in the training phase. This is shown in Figure 12. A potential distance function would be Euclidean distance which is also known as the second norm. This algorithm is

attractive because it is simple but does have some drawbacks. One problem is that it is hindered by high dimensionality which is often accompanied by BCI data. To get a more flexible system, multiple points can be used for each class.

2.9.4 Combination Classifiers

The most popular approach in BCI research up until now has been to use a single classification algorithm. More recently there have been some strategies that have included the use of multiple classifiers to solve a single problem [15]. There are three types of combinatorial techniques that can be used: Boosting, Voting and Stacking.

- Boosting is the technique where the classifiers are in cascading order. If the previous algorithm misclassified the data, the next algorithms can possibly correct the error.
- Voting is the simpler of the combinatorial techniques. It works by each algorithm doing their own classifying and the class with the most predictions is the one that is used.
- Stacking is the application of feeding one classifier directly into the next one. It can be as wide or tall as necessary. Research has shown that it reduces variance and thus classification error.

3 Design

To properly take an objective look at measuring engagement through the use of an EEG while playing video games requires many different design choices. This section will layout the process of designing an experiment with the different components including: setup, equipment, software, and ethics. This section will also cover some complications that had been encountered in the design phase. The primary concept for the thesis experiment was to use a pre-existing video game that was modifiable and had the ability to measure engagement with an EEG. In the following section the game Dunjions will be detailed with an explanation of modifications and tools that were made specifically for the thesis experiment.

3.1 Gameplay Loop

Figure 12: In-game view of Dunjions of the participants side

The first step to design modification to a game was to understand what was currently there. This section will go into more detail what was described in the introduction.

A player's experience when playing Dunjions involved planning and hand eye coordination. The game required the following interactions:

- Movement
- Attacking
- Assessing the knights state
- Healing

The first interaction was to move around the map to find enemies and dodge their attacks. This required spatial reasoning by the player. The next interaction required the player to attack the monsters. This required that the player first recognizes the monsters and swings at the appropriate time. The player will also have to take into account their stamina as if they just swing endlessly, they will not be able to dodge properly and it will take much longer to kill the monster, making the interaction more dangerous. The last interaction required by the player is picking up health. The health pickups is in the shape of a heart and just require the players to walk over it to restore their health.

As the player is playing, the game is continuously recording multiple elements of the game. Key elements are; damage taken by each different monster type, amount of enemies remaining, and how much health and stamina the player has. If the player's health reaches zero, the DDA makes the game easier and if the player eliminates a wave of enemies, the game is made more difficult. The manner by which the game is currently made harder or easier is by changing the modifiers of the enemies. The four main modifiers are output damage, attack speed, health, and movement speed. The amount by which the modifiers change when the player's

health reaches zero depends on how many times in a row it has happened. It also changes the difficulty based on the progression through the level. The progression is calculated by how many monsters are spawned compared to how many were killed. The monster that did the most damage is also modified to be easier to defeat in the event of a death. Modifications are also made after beating a wave. This type of modification uses information on how many waves were beaten since the last death and how much health the player has left.

3.1.1 Enemies

Each enemy in Dunjions had their own unique behavior. Behavior differences included movement, attack, and line of sight. This section will outline each unique behavior that the enemies possess.

The skeleton is the first monster that the players would encounter. Its behavior is that it walks around in straight lines and reflects off any walls it runs into. As soon as the player is in front of the skeleton, it starts homing on the player. If the skeleton collides with the player it takes a swing and damages the player. Once it attacks, the skeleton turns around and starts walking away. If the player attacks the skeleton from behind, it turns around and starts chasing them until they attack or the player escapes.

Figure 13: Skeleton Enemy

The spider is a non-damaging entity. Its behavior has it zooming around the map. If the player is in line of sight. It tries to run in front of the player. While it is in close proximity to the player, it drops an area of effect spider web that slows down the player. The spider can have up to two spider webs dropped at one time. The spider web disappears after 5 seconds. Only the player entity is affected by the spider web.

Figure 14: Spider Enemy

The red monster is an enemy that is mostly immobile. It has the ability to throw a single homing fireball towards the player at which point it is on a cooldown timer. If the player manages to damage the monster, the fireball cooldown is reset, and the monster can instantly attack again. This means that it can attack just as fast as the player attacks. The red monster only attacks once there is a direct line of sight to the player.

Figure 15: Red Monster

The last enemy in the game is the green monster. Like the red monster, it is mostly immobile. Its behavior is more focused on attacks. It has a large amount of health and a long cool down for its attack which is not reset when damaged. Unlike the red monster, it shoots 5 large fireballs towards the player in quick succession which can inflict a significant amount of damage.

Figure 16: Green Monster

3.1.2 Dunjions V3.0

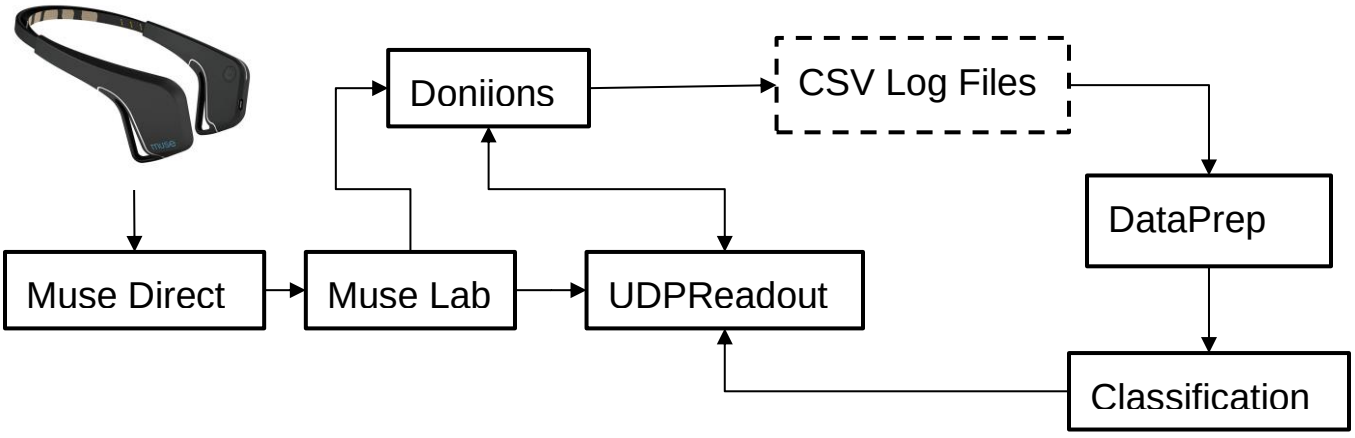

Figure 17: The flow of data through programs used in the experiment

Before this thesis experiment was created, there were two other versions. The first version looked at the effectiveness of a DDA in completion time and the second experiment

looked at comparing the effectiveness of this DDA to a human making changes to the game. The thesis experiment required the Dunjions game as originally developed by David Vallieres to be modified to add the desired feature of EEG integration and a new type of arena combat. The EEG integration required the system to have network communication capabilities.

The EEG device used for this experiment was the wireless Muse 2016 model. This EEG headband was chosen for this experiment for many reasons.

1) The first reason was that it uses dry electrodes which meant that the EEG will be much cleaner to use and less costly because conductive gel was not required.
2) The second reason for this device to be selected was because it was one of the least expensive consumer products to have multiple electrodes on the market at the start of this experiment.
3) The third reason was for its easy-to-use software to streamline the communication for PC integration. The two software used in this thesis experiment (Muse Direct, Muse Lab) will be detailed in a later section under Muse. The Muse API for Unity was designed for use with a cell phone so its built in interface was not available to use for this project.
4) Another reason that the Muse EEG was chosen was because it fit easily on the forehead like a pair of glasses. The quick fitting accompanied with the software quickly relaying proper electrode contact meant that it was easy and fast to fit onto participants.
5) Another reason that the Muse EEG was selected was that it has built in pre processed data capabilities for data that is sent through the bluetooth connection.
6) Lastly an important feature was its wireless capability which meant that it would be easy to connect to the computer without having wires to get in the way of the participants.

Modifications to the back end scripts were made to the Dunjions game to allow it to process the UDP packets sent from the Muse Lab software. The first challenge that needed to be addressed was that the data was coming in at roughly 256 packets per second for the raw EEG data. Since the game ran at 60 frames per second, there needed to be a way to read the data separately. This was solved by creating a separate thread for the EEG data that ran alongside the game. The intercepted data was then processed.

The first step in the processing pipeline was to separate the tags from the data. The tags held the information involving:

1) The user
2) Name of the data
3) Datatype / how many data points.
4) The data in binary

Table 2: Data structure of EEG packet

User/RawData, ddd ###				
Example:	User	RawData	ddd	###
Description:	User Name defined by the Muse software since multiple EEG's are able to connect to the same computer	Name of the data that is to follow. This will say if its raw voltage, battery Percent, accelerometer, ect.	Can be either I or d. I stands for integer and d for double. The amount of each of the letters tells how many digits follow	The data stored in little-endian ordering. The length is determined by the previous letters and count (it is not to be represented by the literal ascii text above)

Once the tag was separated from the data, the data was formatted from byte form to a smaller array of the desired data type. Two processes needed to be done to achieve the formatting requirements. The first process was grouping the bytes. The second process was to change the endianness of the data. The next step was to store the data to be used by the game in short term memory and also to be outputted to a file. The storage technique used in the thesis experiment

was a mixture of dictionaries and arraylists. The dictionary made it easy to separate the types of data for later reference; the arraylists were ideal for storing an unknown length of data, especially when only a small amount of the data is referenced at any one time. Since the most recent data was referenced more often the arraylist was found to be an efficient way to store and use data. Arraylist was also selected to avoid race conditions as shallow copies of the region of data could be referenced and processed by the main thread while the network thread could keep storing the data without needing a lock whenever it received the data from the network.

Another feature that was added to the game for this thesis experiment was an arena style combat area. The arena combat feature was added to version 3 of the thesis experiment for two reasons. The most important reason was that it was much easier to isolate the type of difficulty the player was subjected to. There are three types of difficulties found in video games [7] (Sensitive, Logical, Motor) as discussed in section 2.2. By isolating a specific difficulty, it was determined that participants would be subjected to similar game experiences and have to overcome the same difficulties. In the case of this thesis experiment, the main difficulty that was chosen was motor. Sensitive and logical difficulties were still present but not as dominant as in versions 1 and 2 of Dunjions.

Each wave was made to be progressively harder with different enemy variations. The composition for each wave is shown in Table 3.

Table 3: Wave Composition in Dunjions v3.0

Wave	Enemies	Wave	Enemies
1	Skeleton: 5	7	Skeleton: 15 Spider: 2 Red monster: 3
2	Skeleton: 7 Spider:1	8	Spider: 2 Green monster: 1
3	Spider:1 Red monster:2	9	Skeleton: 5 Spider: 1

			Green monster: 1
4	Skeleton: 10 Spider:1 Red monster:1	10	Skeleton: 10 Spider: 1 Red Monster: 2 Green monster: 1
5	Skeleton: 15 Spider: 2 Red monster: 2	11	Skeleton: 15 Spider: 2 Red Monster: 2 Green monster: 2
6	Spider: 2 Red monster: 4		

3.2 Motivation

It has been a goal for many different areas of behaviour related sciences to understand the brain and how it reacts to certain types of stimuli. Typically, researchers approach this problem by measuring behaviour through the use of questionnaires or the effects of a person's reactions such as body language. Questionnaires and visual observations are and have been invaluable methods that researchers have used to understand a persons behaviour. More recently, with more powerful hardware becoming less expensive and new data processing techniques becoming available, a wider range of researchers are now able to easily test and analyze how the brain reacts to outside stimuli.

3.3 Similar research

The starting point for this experiment was research derived from David Vallieres experiment [49] testing the use of a DDA algorithm in a video game. The revised experiment focused on a different area of research using EEGs to predict engagement levels to elevate the

abilities of the DDA. Three different papers were invaluable to the idea leading up to the creation of this experiment.

The first paper that influenced this experiment was an experiment undertaken by G. Chanel et al [18] where they applied an SVM classifier onto biofeedback measurements while playing a game of tetris to try to determine if the player was engaged. They tried to classify if a player was either playing in an easy, medium or hard mode and linking that to boredom, engagement or anxiety respectfully. By feeding their result into an SVM, they reached an accuracy of 53% without the use of an EEG. In a followup paper [50], the researchers used the EEG in addition to all the other biofeedback measurements and saw an increase in accuracy to 63%.

The second paper that influenced this research project was by M. Salminen et al [5] . The reason why this paper influenced this research was because they utilized oscillatory EEG signals to classify specific events in the videogame Super Monkey Ball 2. The events they were trying to classify were rated as either good or bad. For example, picking up collectables was considered good and falling off the map was bad. The method in which they classified the good and bad events was by comparing the activation of certain areas around the brain. Because of this research specific event prediction was used in the thesis experiment.

The third most important influence which is a larger set of experiments was The Fun of Gaming (FUGA) initiative [4]. This was a government backed project that had the goal to create and improve methods to measure different dimensions of video game experience. The papers that came out of this study which showed that numerous researchers were working on similar projects. The tools and techniques that were derived from this initiative gave ways to handle problems that occurred throughout the design and application of the thesis experiment.

3.4 External Components

Due to the nature of this experiment, specialized hardware and examination tools were necessary. This section will look at the components that were acquired for this study. Four main components were used for this study:

1) Computer
2) Muse EEG
3) Questionnaire
4) The Testing Lab

3.4.1 Hardware

For the thesis experiment, the following computer system components were used; a windows based desktop computer with a minimum of 8 GB of Ram with 4 physical CPU cores and Bluetooth connections with 2 monitors and 1 Xbox controller as peripherals.
The specifications for the computer and its software can be found in the Appendix 8.2. The most important aspect of the computer was that it was able to handle the computation requirements of the software and the associated algorithms.

3.4.2 Muse

Figure 18: Muse 2016

The EEG device used for this thesis experiment was the Muse 2016. This model is sold by the InteraXon company and primarily targets the meditation consumer market. The device is able to collect raw data at a rate of 256 Hz. This allows it to be used to find the frequencies generated by the neurons which are typically around 0.5Hz to 60Hz. The Muse EEG runs wirelessly using the bluetooth 4.0 hardware with a 128 bit Advanced Encryption Standard (AES) encryption with data integrity checks. The battery in the device has the ability to last up to five hours between charges. The devices have been used in numerous other studies, some which required ethics approvals and passed safety standards for consumer use by the Federal Communications Commission (FCC), CE, and TÜV SÜD. The electrode placement on the scalp

is based on the international 10-20 system and measures the F7, F8, T9, and T10 channels. Other measurements it can transmit are accelerometer data at 52Hz, battery level at 10Hz, gyroscope at 52Hz, and alpha, beta, theta, delta and gamma waves.

At the start of the thesis experiment the Muse company had two free programs for research purposes using computers. The two software offered were Muse Direct and Muse Lab. Currently, both software's are no longer supported and require special permission to use [33]. Although there is no more support for the computer software, there is ongoing support for their subscription-based application for iOS devices.

Muse Direct is a software that reads the data being sent via Bluetooth. This software has different options on what to do with the data. It gives the researcher the ability to select how the data is formatted, and what port and protocol to send it over the network. It also allows the user to save the streaming of the data to a file. The problem with data in this file is that there is a lot of information stored, much of which is not needed in this thesis experiment. Some examples of the data that is unnecessary is who the data is coming from, and the type of data on every line. This means that the files are much larger. An added advantage for some researchers is that the software allows the connection of multiple devices to be routed through this software. For the thesis experiment only one device was used, so the stability of having multiple devices was not tested.

Muse Lab is a program that allows users to visualise the data that comes in from the Muse Direct software. The program is java-based and allows the user to read the data sent from Muse Direct and specify which pieces are then sent out back to the network. Once the data is received, the software has the ability to record the information. The visualisation side of the program lets you control the speed that you see of the incoming data, the frequency of the

incoming data, and graph the data in the form of a stationary or a scrolling line. The program also allows the user to select which data goes out to the network. For the thesis experiment data was sent across two ports which allowed the subsequent programs to process the information they needed. Another option of this software that was not tested is the ability to set markers in the data for specific events. Because this requires manual use, this option was not used. A Muse Direct tool that was used for the thesis experiment was the ability to save and load all the settings applied for a specific connection including all output data. This helped ensure that all trials were as similar as possible.

3.4.3 Questionnaire

The questionnaire that was the basis for the thesis experiment was based on the "Game Experience Questionnaire" [2]. This questionnaire has 33 questions and was designed to measure seven different parts of interaction namely competence, immersion, tension, challenge, negative effect, and positive effect. A shortened version of the "Game Experience Questionnaire" that was used for the thesis experiment is called the "In-game Experience Questionnaire" and it consists of 14 questions. The response of the questions are marked on a scale between 0 and 4. For each of the seven parts of the game experience there are two questions. The final score is then the mean between the two scores. There are also extra questions in the case that the translation from its original form in Dutch is inadequate but in the thesis experiment, they were not used. The version of the questions used in this experiment can be found in section 8.3. The reason for using this questionnaire is because it has undergone reliability tests and resulted in satisfactory to high internal consistency. They found it was sensitive enough to pick up differences from gamers, game types, play characteristics, and social context of play.

A second questionnaire was also given at the end of the session to each participant to collect demographic information. This contained information about their age, sex and some information that was determined to be important for the EEG signal classification, such as which hand do they consider the dominant and if they were affected by colour blindness. Lastly, the participants were asked for information pertaining to their experience with video games, such as how often they played games in an average week and how many platforms/consoles they used when playing games. This information was found to be useful in previous game experiments. The previous experiments determined that there were differences in classification results from two main types of players: the hardcore players and the casual players.

3.4.4 The Laboratory

The experiment required the use of a Laboratory to run the tests in a controlled environment. Because of this need the CHIL lab was used as the room to run the tests in. It was used since it is a smaller white room about 3meters by 2 meters. The walls are all white except a one way window on the one side of the room. The window is there incase an experiment needed to be monitored without bothering the participant. In this thesis experiment, the participant and the experimenter were in the same room.

3.5 Software

The result of the thesis experiment was dependent on different software and programs working in conjunction. This section covers the networking protocols and the programs that were part of the thesis experiment. Software coded specifically for this thesis experiment will be explained in detail. The two main languages used were Python and C#.

3.5.1 Networking Protocols

Well designed network communication becomes vital when rapid communication is needed between programs and platforms. As this thesis experiment required numerous programs and interfaces to communicate together, internet protocols on the transport layer were needed to be taken into consideration in the design. Two main protocols that are typically used for transmission of messages are Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) [41]. Each protocol advantages and disadvantages are discussed below.

Transmission Control Protocol

TCP is able to handle safe and reliable connections across the internet. The manner in which it handles this is by having a two way connection between the two different hosts.

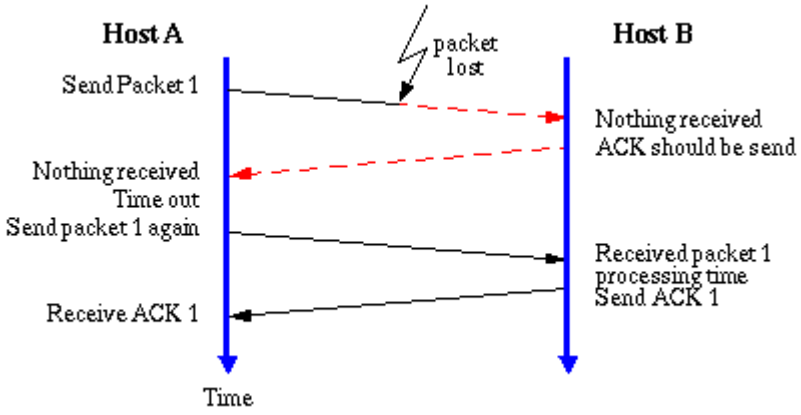

Figure 19: picture displaying two way communication of TCP [41]

The communication between the two hosts consists of a package sent to a destination host and the destination host acknowledging that they received the package. TCP also ensures that there was no corruption of the message and that if there are many messages to be sent, that they are received in the proper order. Although there are many advantages of using TCP, there are two key disadvantages, those being:

1) Large load: When there is a lot of small data being passed, the bandwidth is filled with more data since communication needs to be passed in both direction
2) Longer delays: although the delays cause by tcp are usually very small, they do add up when a lot of data is being communicated. Some systems will wait before they receive the acknowledgment that the data was received before sending more data. If there is ever packet loss, this can back up communication quite a bit.

For the majority of applications TCP is adequate because the servers only have to handle a few packets per user or the delays are negligible or imperceptible.

User Datagram Protocol

UDP is known to be simple and fast. UDP packets contain a source, destination, message length, checksum and a message. When data integrity is needed UDP may not be the best protocol to use since packets are often dropped or lost due to networks being imperfect. For this thesis experiment, on the other hand, all communication was done on the same hardware so the packets went almost directly from one program to another with very little chance of packet loss or disorder. For the thesis experiment UDP was selected over TCP because of the time sensitivity of the data. By using the quickest protocol the throughput needs of the data communication was satisfied.

3.5.2 Python

Python is an interpreted programming language that is widely used throughout the scientific community. The language is interpreted at runtime. For this thesis experiment a Python program had been made to run alongside the game. The reason that python was selected was for its large amount of optimised libraries and its ease of use.

The python programs were primarily focused on the data side of the experiment. This means that it was used for:

1) Recording data
2) Preprocessing data
3) Training ML models
4) Running classification algorithms

The 1st and 4th uses were done alongside the game and the 2nd and 3rd were used seperatly.

To be able to work with the video game side, it was loaded with a rudimentary RPC (Remote Procedure Call) capabilities. The RPC side of the python program was to give the Video game the ability to trigger events to do certain things. These events included:

1) When the game started
2) When baselines were taken
3) When predictions were needed
4) When the game ended

Each event was chosen because of a specific need. The game staring was important because it told the python program to start logging the all the data coming in. The baselines were there so that the later algorithm that needed to normalise the data to each participant's brain activation levels was able to collect the clean data. The prediction call event was needed for the inference to be made on the game's behalf. The output of this call would allow the game to make modifications. The game ending was so that the python code would not continue unnecessarily.

Recording the data

The program required access to the incoming data and any other information that could be important to the classification process. For the purpose of this thesis, there were five different files that were recorded for each participant. These files included:

1) Eyeblink
2) Raw EEG
3) Notched EEG
4) Event Log file
5) Stats file

An eyeblink file, which would hold the information of when the EEG predicted when the players were blinking was recorded at a 10Hz interval using the headband's communication. The second file would hold the raw EEG data coming from all four electrodes. As this data tended to be noisy, it was later ignored for use with analytics. The next file contained the notch-filtered EEG data. This data was cleaned to remove any signals between 45 and 60Hz. The fourth file recorded was the log file. This file held information pertaining to important events that happened in the game, such as if the player was killed or reached a checkpoint. Other important events contained in the log file were; when the baselines were taken, if the player picked up health, and when there was a change in scenes. The last of the files recorded is the stats file. This file contained the changes in the states of the player throughout their gameplay. Every time it changed it would log the stats for the enemies.

Data preprocessing

The second step involved making the predictive model using SVMs. This process was done by first initializing necessary methods and variables:

1) The first variable was to be able to map ranges of frequency with their names. These were the Alpha, Beta, Delta, Theta, and Gamma.
2) Define a method to be able to open comma separated values (CSV) files and to transpose them. This would allow the use of external file storage for holding the player data.
3) Define a method to do a time to data conversion. This would allow users to convert microseconds to a specific record from the EEG data.
4) Define a method to remove large fluctuations in the data. The method would search for any data that was outside the third standard deviation of the average in either direction. Such events would happen when the player would move their hand near the EEG with or if excessive blinking were to occur.
5) Create a method that would allow a Fast Fourier Transform to be applied to large amounts of data all at once. This was done using a library called Tensorflow as its signal processing package used a method to process data packets called short-time Fourier Transform (STFT). Even though Fourier Transformations convert time series data into frequency data, Tensorflow gives the ability for frequency data to be seen across time in an efficient manner.

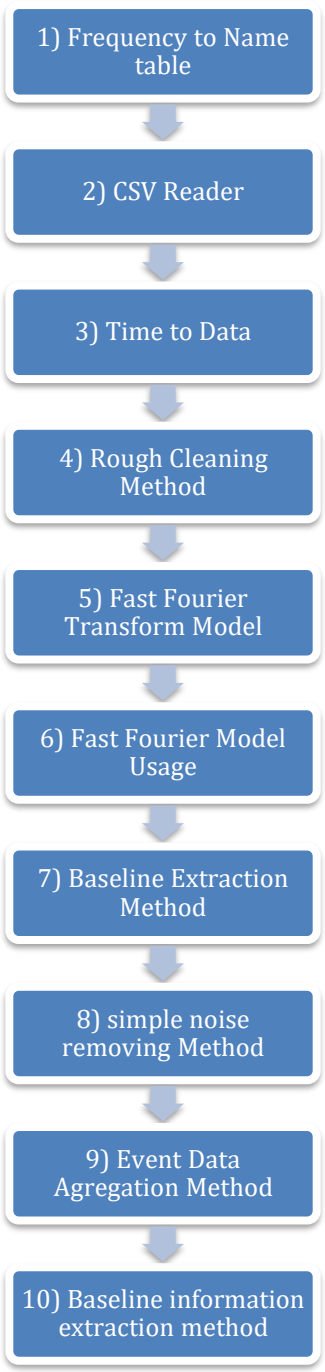

Figure 20: Data Preprocessing flowchart

6) Once the model was programmed, a method to run the model with specific data was made.
7) A method was then created for extracting the baselines from a set of data
8) A method for cleaning the data using independent component analysis was then made. This method would look specifically for markers such as eye blinks but could potentially be used to look for and clean other noise.
9) The next method that needed to be programmed was to group processed data together by events to then be fed into a classifier. The two events that were used to group the data for this thesis experiment were deaths and checkpoints. This method would take 30-second increments between the previous event and time of the current event. It would then average the frequencies in each increment into the bands of interest; in this study it was alpha, beta and theta. After that it would flatten the two-dimensional array into one dimension and store it into the output array while also retaining the information on the participant it came from and the event it was associated with.
10) The last method needed was to determine the average intensity and the standard deviation of the EEG data based on the baselines.
11) Once all these methods have been initialised, the main program is run. The main program processed the participants files by opening up their EEG data and log file. It would then loop through the log file and store the appropriate log data in separate arrays. For example baselines for eyes open, closed, and if there was a death or checkpoint that occurred. Next it would run the event extraction method and append all the different participants' data together and store it into a NumPy array ready to be used in the classification program section.

Data classification

After all the data is processed and is all in one file it is then used in the classification process. Following are some important methods used in the classification process:

1) Visualisation of class distribution. For example it is good to know how many data samples relate to deaths vs checkpoints
2) Data siplification methods. It can be easy to train any model to have a bias. For this reason a method to balance class distribution is helpful so make sure that the model does not just train to predict one class because there are more data points available. For example there are more deaths related datapoints so being able to drop some of these points so there is a 50/50 balance between chackpoints and deaths is very useful
3) Participant bias removal. When training a model, it is useful to train on different participants then you are testing on. Because of this, having a method to extract a specific person from the dataset and run a test on them seperatly is useful.
4) Next is the process of separating the data into testing and training data. This looks to see if the trained network is generalised enough to be used on new data. The ratio used for this experiment is a 70% training to 30% testing. Since the desired choice of classifier is a Support Vector Machine (SVM) the LIBSVM library is used for this next part. This library allows the use of a fine tuning argument “c value”. This allows control over how large the support vector is to be and also what sort of misclassification is allowed. By testing a handful of values, the best value can be chosen. A scoring system was made on the results to be able to automatically find the best network. The scoring system is obtained by finding the difference between the testing and the training set accuracy, then dividing it by two and raised to the power of two then subtracted to the accuracy of the

testing set. Therefore, the most general and most accurate results are preferable but still allow a little variation between the testing and training sets.

Game Processing

The last step is to have the communication open between the game and making a prediction. This is done with the third and final program. Many of the same methods from the cleaning and data preparation script are used, such as the cleaning method using the ICA algorithm, large noise data removal, baseline extraction, the tensorflow Short-time Fourier Transform (STFT) model and method, and a modified version of the event extraction. The modification comes from only having to create one data sample from the data that is read from the Muse software instead of reading it from the output of the CSV files. After all those methods are set up, it is only a matter of opening up the program to networking communication. This is first done by opening up some sockets to listen and send UDP datagrams through and initialising some variables to hold data. Some of the variables that are being initialized is an array to hold all the EEG data incoming. It is set to an initial size of 1,000,000, which gives roughly 65 minutes of data storage. This can be modified if more data is expected. Other variables include timing information for baselines and how many records are currently being stored. The last important variable is the SVM model that was determined from the previous program. Once all those variables have been initialized, an infinite loop is started and the program starts to listen. It must decode the incoming signals if: the incoming data relates to a game event, it will have the keyword game at the beginning; if not, it contains data pertaining to the EEG.

3.5.3 C#

The Dunjions game was almost exclusively written in C#. There have been 70 separate class files written for this game. The more important scripts relevant to the thesis experiment will be covered in this section. The game's programming has a hierarchical structure that allows information to be handled more through the scripts and less through the manual manipulation through the engine. The most important script is GlobalVar. There are other scripts that handle the stats, enemy logic, heads-up display (HUD) elements, network data managers, and the outputting of data to files.

Global Var

The GlobalVar script was a critical addition to the thesis experiment game as it allowed important class files that are dynamically allocated to be accessed in an efficient manner. It did this by having static references to the important variables.

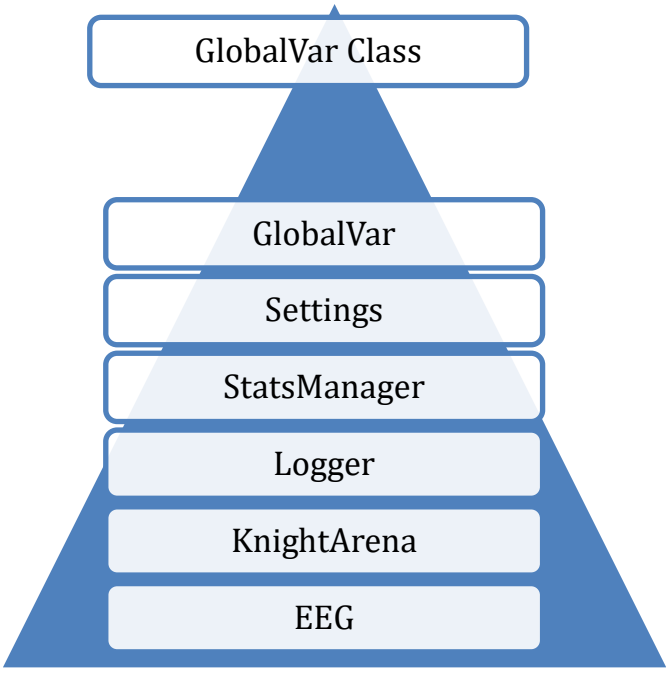

Figure 21: C# class file name scheme

These static class variables are the settings, the stats manager, the logger, the player and the networking EEG data handler.

The settings variable is used to hold three different variables: file location, volume levels, and port number. File location is used to determine where the log files and the EEG data dumps are to be stored. Volume level is used to control the intensity of the sound. The port number is used to store the port number that is used to listen for the EEG data. It is also done through the engine's Playerprefs class, which allows the settings to be stored between sessions if needed.

The stats manager is an integral class that handles storing, initializing, and modifying statistics as well as keeping track of any changes. Since all entities in the game are programmed to clone stats when they are created, the static reference in GlobalVar allows a simple way to reference the manager.

The static reference to the Logger instance is held in the GlobalVar class because there are many different other classes are programmed to output their data to one of the five output files. The Logger class contains a reference to the output files to avoid having to open and close them every time data is written. A timestamp is also placed on all the output files, which is zeroed out when the class is initialized.

The reference to a KnightArena instance is stored for easy access. KnightArena is the class that handles the controls of the playable character and handles the player stats during runtime. The reference in GlobalVar is overwritten by the KnightArena class when a new scene is loaded. This occurs because there is always a new KnightArena object every time the scene loads, and the object is reset and re-linked to GlobalVar. By having a static reference, player information, such as location, is available to all the enemies for behavior and stats use.

The last static variable that is stored in the GlobalVar class is the EEG data class. This class handles all the short-term storage, multithreading network communication, and simple processing. It is stored here for simple access for the visualization scripts.

Stats Manager

As noted above, the stats manager has numerous useful functions. It handles the storage of the enemy and player stats in two ways. The first way is to keep backwards compatibility with the older versions (v1,v2) of the game. The older versions had only a few modifiable stats that could be made. Resultantly each enemy had specific stats that were changed in a specific order when an event occurred. Each specific stat had a multiplier and a bias. In version 3 they were set manually by a serialized class called scriptableSelectiveStats to make them more modular. Once the initial stats were loaded into the stats manager the multiplier was set to 1. As the game progressed, the multiplier attribute was modified. Every time the game needed to become harder or easier, the stats instance for each enemy was modified. The stats class held information such as health, attack speed, damage, and timers. Each entity (enemy and player) cloned their default stats from the stats manager when they were created. The stats manager held a version of the current stat and the pending stat modification. The reason for holding the current and pending stats was to allow modifications to be made if and when needed without effecting the current state. In previous versions (v1 and v2) this was

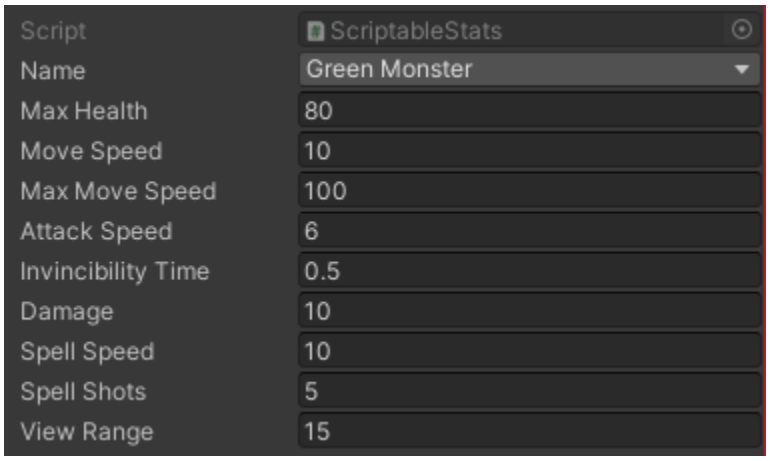

Figure 22: Example of stats of a monster

more important as enemies were spawned during gameplay and the stats were also able to be modified by another hidden player also known as a “wizard.” The stats manager will also hold other types of statistical information. Some of this information consists of; damage dealt and from which enemy, how many of each enemy is killed, how many player sword swings have been made, how much a health pickup is worth and how many health pickups have been made, which wave the player is on, what type of difficulty modification is being used and the starting time of the game.

The difficulty modification types are handled by the DDA class. In version 3 of the game there were three different difficulty modification methods. The first method which was not used was the lack of modifications on a baseline difficulty set at the start. The next method was a modified DDA system from v1 and v2. It was modified for ease of use and to be more dynamic. The last method was a combination between the previous method with the classification predictive element received from the python code. Once the prediction was received, the system modified the player’s stats for both live and for future iterations.

EEG

The EEG class was comprised of several interlocking methods and variables. The starting method was triggered as soon as it is created. It initialized an empty dictionary to store the data that came from the Muse. The dictionary mapped the data type to a linked list of type EEGData through a KeyValuePair. The EEGData class contained the time it was received, the data type, the data array associated, and a flag to see if it was modified. The flag was used to ensured that the cleaning of artifacts was not done more than once for each piece of data. The cleaning of the data was not used in the testing phase through the C# game code. All of the cleaning was subsequently done through the python code. The other stored variables in the EEG class were the

baseline values for the EEG data, the value of each channel and frequency over time after undergoing a Fourier Transformation. This was used to average out the frequency values over time for the visualization purposes on the observer screen. Lastly, variables containing the networking and multi-thread information which included the references for UDPClient, port number, and thread were part of this class.

UDP communication was the first thing that was started when the EEG class was initialized. Part of networking initialization requirements were set up on its own thread. The reason for doing so was that Unity's scripts were all run on a single thread sequentially. So as to not disturb regular game logic, networking communication was needed to be done continuously and as fast as possible to prevent information from being lost. By having the networking on its own thread, the program will lose some ability to use some internal Unity timing functions. Additionally, it will gain the ability to create an endless loop that intercepts the packets as soon as they come in. As soon as the new thread is initialized the port is reserved and the loop starts. Whenever data comes in, it is initially deciphered to detect what type of data it is. The data is deciphered by parsing through the incoming packet for a specific marker. The packet (User/RawData, ddd ###) would contain a 'd' character for double and 'i' character for integer. The quantity of the data stored in the packet were determined by counting how many of the characters were together. Once the type and quantity was determined, the program then isolated the subsequent byte data, changed the endianness, and converted the subsequent byte data from byte arrays to its generic data type. Once the conversion was complete and the processed data from the network packet was stored in an array, it was then added into the dictionary and logged to the output file.

Several helper methods and access functions were used for the EEG class. The first method was to convert a byte array to a primitive datatype. The integer conversion was done with bit shifting, index placing, and multiplication. The byte array conversion to double type used the Bitconverter library that comes with C#. Other helper methods included; getting the banded data from the raw data, logging the FFT data to check if the algorithm was implemented correctly, some cleaning algorithms from earlier in the process that mimicked some of the Python script, and a GET method for the EEG data.

DDA

A class was created to handle all the modifications for both the DDA system and to handle the predictions returned from the external Python scripts. The old DDA system from version 1 and 2 were streamlined to be simple to read and easy to modify. The streamlining was made by making each step in the DDA distinct from each other. The first step was to find the last event and then count how many times backward that event occurred consecutively. It then found the most damaging monster that was encountered since the last event. The script then determined the percentage progress that was made before the event occurred. 100% progress meant that the player had reached a checkpoint. The DDA then loops through each monster stats to modify their specific attributes according to the event type, then health remaining for a checkpoint or progress before they died. The consecutive events then added an additional modifying multiplier. Subsequently if the player died, the enemy that caused the most damage was modified again to be made easier.

In version 3 the algorithm added the EEG predictive modifications. The request prediction method of this class included a message sent to the same port that the Python script was listening on, telling it to make a prediction. It then started a separate thread to wait for a

response from the python code. A separate thread was used because the prediction process could take between 3 to 40 seconds in which the game would have been frozen.

3.5.4 Data Preparation

Before data was used for analysis, it was important to pre-process it because the EEG device was very sensitive and prone to noise and picking up unwanted artifacts. There was many different techniques that were used to reduce noise. The two noise sources that were removed in this thesis experiment were electrical and muscular. This section will cover how they were accounted for and the techniques used to remove them automatically.

Generally, the first noise that is to be eliminated is the background electrical energy that surrounds you when you are in a building. In Canada, the wall power runs at 60Hz, so the removal of this specific frequency is necessary. This can be done through the use of a time series to frequency domain shift. Such an example can be done with a Fourier Transformation. Once the transformation has been done, it is as simple as not looking at the 60 Hz frequency and the surrounding few frequencies.

The next artifact that had a large impact on the data was the effects of muscle movement. Three main sources of muscular noise came from blinking, jaw clenches and a hand touching the head. When someone blinks, there is a spike in the electrical activity measured. There were two techniques used in this study to reduce the effects of blinks. The first technique was to use the Independent Component Analysis. This method gives the ability to remove the majority of the blink noise, which is somewhat consistent, and keeping the underlying signal underneath. The second technique used for this study was to replace the length of the blink with a 0 Hz frequency. This means that the voltage stays the same across the event. There are two problems with using

such a technique. The first problem is that there would be a loss in brain activity measured. This loss was mitigated by averaging out the measurement across a longer period of time and therefore reducing the loss. The only resulting side effect would be the amplitude of the bands themselves would drop and the 0Hz amplitude would rise significantly. The 0Hz in this study and many others was therefore ignored. The next technique utilized was used to further reduce data loss from the previous problem. This was done by adding a larger window that the Fourier Transformation investigated. The smallest windows that should be used is the window relating to one second. This makes it easy to convert a specific data point to a frequency. For example, if the EEG is recording at 256 Hz, the window would be of size 256. This means that 1Hz would be in the 1 location in the vector and so on. By increasing the window to a value that is 2 to the power of something, the Fast Fourier Transformation can still be used and then you can multiply the frequency you want by the size of the window. For this example, the window would be of size 1,024 for an EEG that streams at 256 HZ. That means that for each frequency there are four values that can be looked at giving a higher accuracy.

3.6 Ethics

To be able to run the experiment on participants, ethics approval was required by Laurentian University. The process was done through the Laurentians University's ethics board. A few aspects of the process were interesting to note. These aspects include the hardware, the video game in question, and the data.

4 Experiment

The thesis experiment was comprised 8 steps. These steps were:

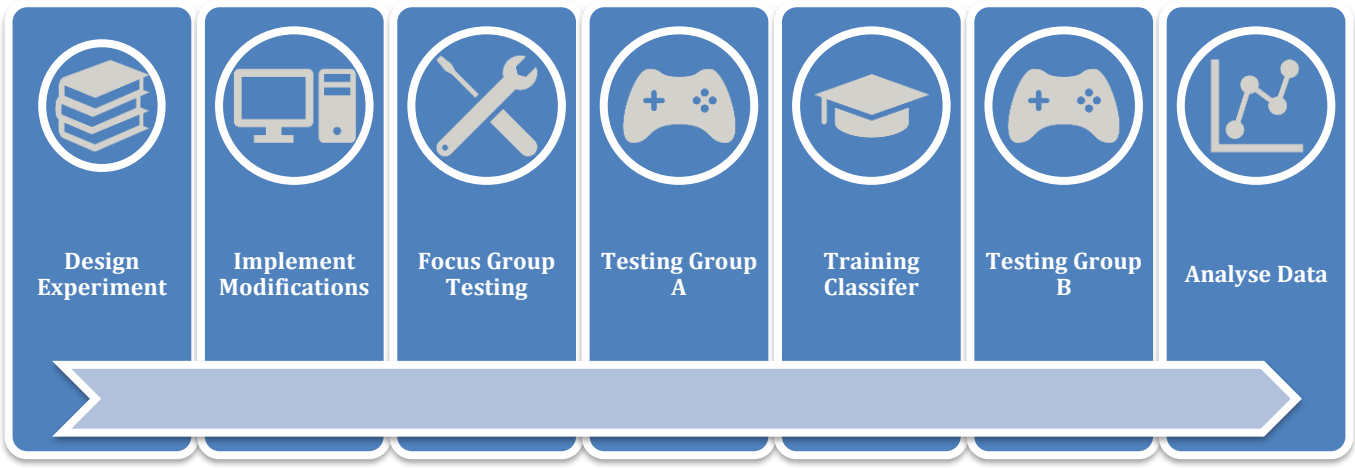

Figure 23: Experiment progression visualisation

Each step had to be done in the specific order that they were listed.

Designing the experiment was the first step. This stage involved coming up with the idea and doing literature review and seeing what kind of questions needed to be answered. This is also the part of the experiment that took the longest.

After having gone through the design phase of the experiment, next came the implementation of modifications. This stepped involved finding the appropriate modification and additions needed to be applied to Dunjions to be able to test the experiment laid out in the design step. The important modifications and additions were laid out in the design section of this thesis.

The next step involved in the experiment was the testing of a focus group. This was a necessary step in the experiment to really make sure that all the components were ready for the actual participants. This step is explained in more detail in section 4.2.1.

Once the experiment was ready for data collection, the first group was run through the game. The outline of the main experiment is explained in detail in the section 4.2. The necessity of this groups data is outlined in section 4.2.2.

Once the data was collected for the initial group, their data was now able to be fed through a classifier. Some preliminary testing on the data was done to see if the proper classification algorithm was used. Some of the results of this section are discussed in detail in the Discussion section of this thesis.

Once a classifier had been trained, it was now time for testing the second group of participants to see if the modifications based on the predictions of the classifiers helped support the hypothesis.

The last step of this thesis experiment was to run statistical analysis on the results and to further test the chosen classification algorithm to see if it improved with more data.

4.1 Objectives

This thesis experiment had two main objectives. The first objective was to find a classification algorithm that was able to work in real time on brain data. The second objective was to test see if with the help of the classification algorithm, if the game would be more enjoyable. Due to these objectives a hypothesis was outlined to be tested. The hypothesis that was set out to test is outlined below.

4.1.1 Hypothesis

H0: The enjoyment of a video game is independent of a DDA system with access to EEG data

H1: The enjoyment of a videogame improves from a DDA system with access to EEG data

The hypothesis will be tested with the use of a ANOVA analysis on the reported positive effect of the game as reported in the Game experience questionnaire between the two versions.

4.2 Main Experiment

The thesis experiment required 3 groups of individuals; focus group, group A, and group B. All three groups perceived the main experiment process the exact same way. The differences arose from either before the experiment started or after the experiment finished. Each difference will be discussed in the subsequent sections.

The room that the experiment took place in was part of the CHIL lab. The lab is in a small white room with a one way window. The reason that this room was chosen for this experiment was because it satisfied all of the requirements for this test. The requirements that were met were that the room be quiet, have minimal distractions, large enough for at least two people to sit comfortably. The setup of the experiment had a table facing the wall with two monitors on it. Only one of the monitors was facing the participant. The computer running the game stood beside the table. The participants were asked to assigned to sit on a well padded rolling seat that allowed them to slightly lean back if they wished to do so.

The following describes the process of the main experiment that was common between each group.

1. The first thing the participants did upon entering the testing room was to fill out a form stating that they agree to be tested. This form was accepted by the ethics committee of Laurentian University. The consent form had information pertaining to what they would be doing, what the EEG was and what would be done with the respective experiment data that was recorded. The participants had the added option to get a summary of the results of the study sent to them.
2. Once the consent form was filled out, the participants were fitted with the EEG and were told to relax. They were instructed to try to keep their heads still, to avoid touching the EEG, and to avoid clenching their jaws throughout the experiment. Once the EEG could be seen to have good contact to the head using the Muse Direct program the experiment continued. They were then told to try to relax and take slow deep breaths. The participant were asked to signal that they were ready or two minutes had passed.

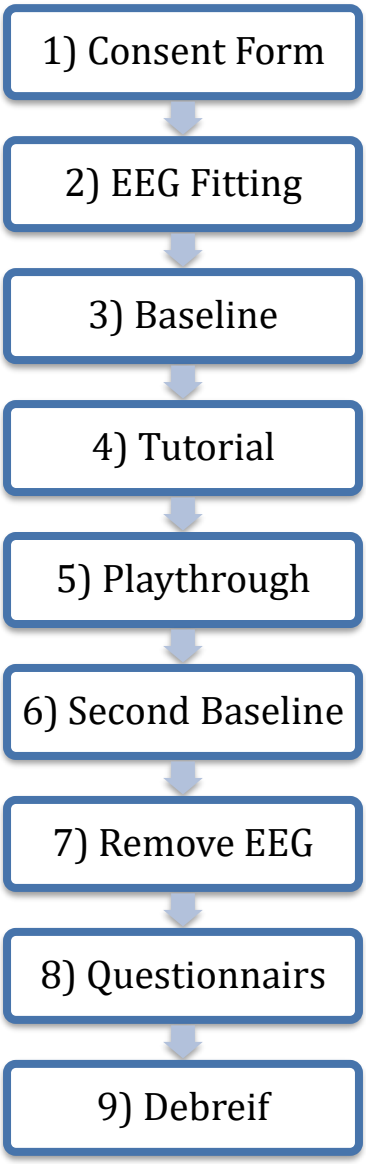

Figure 24: Experiment Procedure

3. They were then instructed to stay still as two baselines were taken. The first baseline was taken with the eyes open and the second baseline was taken with the eyes closed. Each baseline took 60 seconds.
4. After the baselines were taken, the participants played through a short tutorial instructing them on how to use the controls.

5. After the tutorial was completed, they started playing. The participants were told to play for roughly 25 minutes and reaching as far in the waves as they could. Each wave was made to be progressively harder with different enemy variations.
6. Once the time for play session ran out, the game was ended by the experimenter and another two baselines were taken.
7. Before continuing with the questionnaires, they were instructed to remove the EEG.
8. Once the EEG was removed, they were given the modified game experience questionnaire to fill out. When that was completed, they were given the demographic questionnaire to fill out.
9. Once the questionnaires were completed, it marked the end of the experiment and were given whatever time they needed to gather their personal effects and ask questions about the experiment.

4.2.1 Grouping

The Experiment was split into three groups. Although The focus group, group A, and group B went through the same procedure as mentioned above, the differences are outlined in Table 4.

Table 4: Experiment Group Differences

Group	Description
Focus Group	• Testing Experiment Viability • Bug Finding • Data used for algorithm testing. • Collecting Experiment feedback
Group A	• Control Group for Engagement score • Data collection for training AI for the new algorithm • Difficulty modification was done by only DDA based on success and death
Group B	• Difficulty modified by old the previous DDA and the EEG classification

5 Results

The analysis to prove the hypothesis was done by disproving the null hypothesis. This was done by running a one-way ANOVA analysis on the results of the Game Experience Questionnaire. After running this analysis, it showed that between the two test groups, there were no significant statistical differences and thus the null hypothesis cannot be rejected. Table 3 shows the ANOVA results for all seven elements it measures. The hypothesis proof was using the statistical results of the positive effect.

Table 5: ANOVA test Results

Element	Result (Alpha=0.05)
Positive effect	**F(1,21) = 0.60905, p = 0.44386**
Competence	F(1,21) = 0.05645, p = 0.81451
Immersion	F(1,21) = 0.39255, p = 0.53772
Tension	F(1,21) = 0.07475, p = 0.78722
Flow	F(1,21) = 1.56404, p = 0.21241
Challenge	F(1,21) = 1.27896, p = 0.27084
Negative effect	F(1,21) = 1.81655, p = 0.19209

From the above table we can see that the ANOVA test on the different aspects of the In-Game Game Experience Questionnaire wielded a non-significant result for every attribute. The most significant of the result being the negative aspect of the game meaning that the modifications the EEG was potentially making the game less enjoyable for more participants than the first group. Since the value was too far from the desired 0.05 value, this conclusion is deemed unsignificant without more testing.

6 Discussion

There were numerous results and lessons-learned from this thesis experiment. These ranged from the accuracy of classification techniques, the final results of the experiment, and the challenges that were encountered. These are further expounded below.

The experiment had finished with 23 participants (15M,8F). that was roughly 11 people in each group. All but two were right handed and so only the data for the right handed-individuals was used.

6.1 Classification Accuracy

Classification algorithms were tested on group A's data to check the accuracy and appropriateness of the processed brain signals to the relevant events. Below are the results of this testing along with comparisons to group B's data after their respective runs. The first algorithm analyzed was the K-Nearest Neighbours (KNN).

KNN was the first technique investigated as a possible classifier. The KNN class in the Scikit learn library implemented for Python has one variable that is controllable called K value. The performance of the algorithm was determined by testing multiple K values and recording the average and maximum accuracies achieved. The graphs below show that the maximum accuracy peaks to just under 53%. This accuracy was insufficient for use in classifying group B.

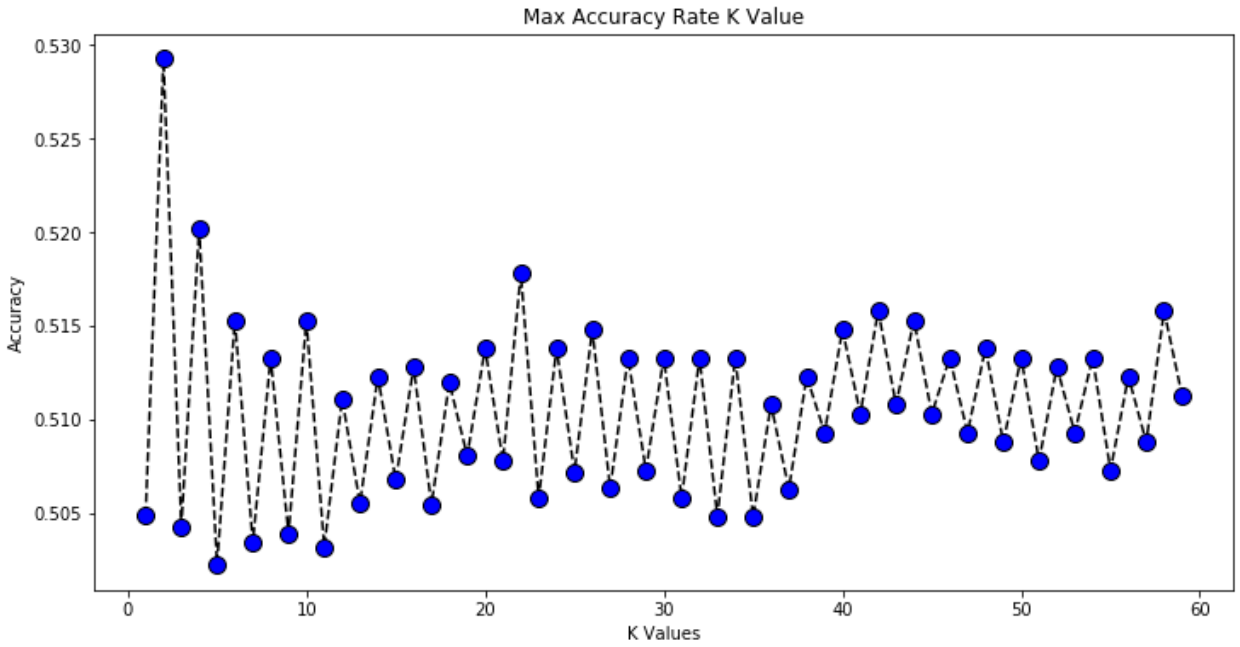

Figure 25: KNN Maximum Accuracy for KNN for Group A

As noted in Section 2.9.1, another recommended classification algorithm to use was Support Vector Machines (SVM). The code that was used for this algorithm came from the Libsvm library [29]. This library is offered for many different programming languages but the Python version was chosen. This library was chosen for its simplicity and its ability to store its structure between classification easily. Two main variables were compared in the process of selecting a SVM classifier. These two variables were the kernel type and the cost (accuracy versus computational time). The underlying library's implementation tries to find the optimal setting for most of the remaining variables automatically. Four different accuracy Tests were performed for the SVM algorithm. These tests are as follows:

1) Apply the SVM To the Group A and testing its accuracy on a single member that is seperated from the rest of the group

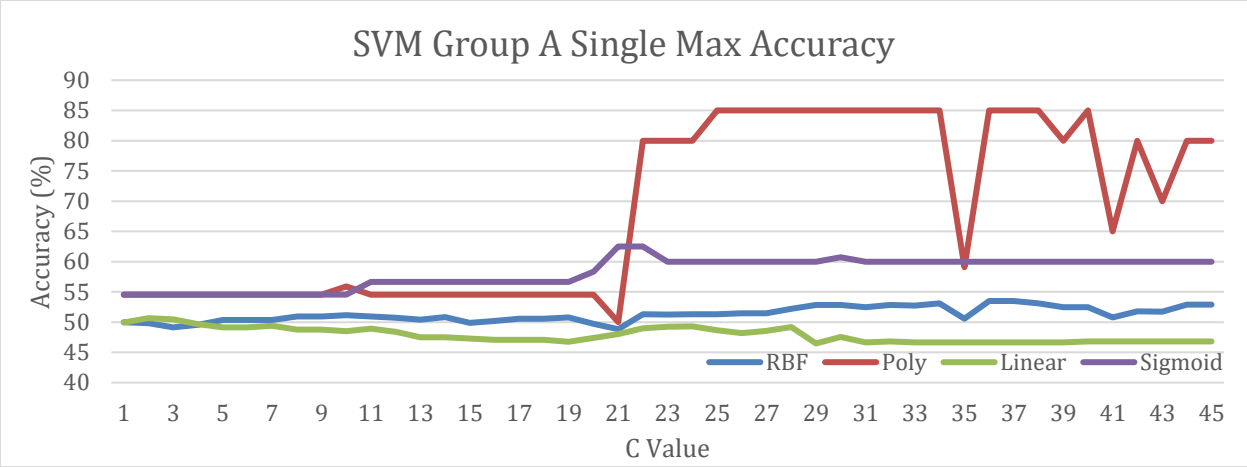

Figure 26: Single max accuracy for SVM's used on Group A

2) Train the algorithm only using Group A but separating the groups data evenly in a 70/30 ratio between training and testing

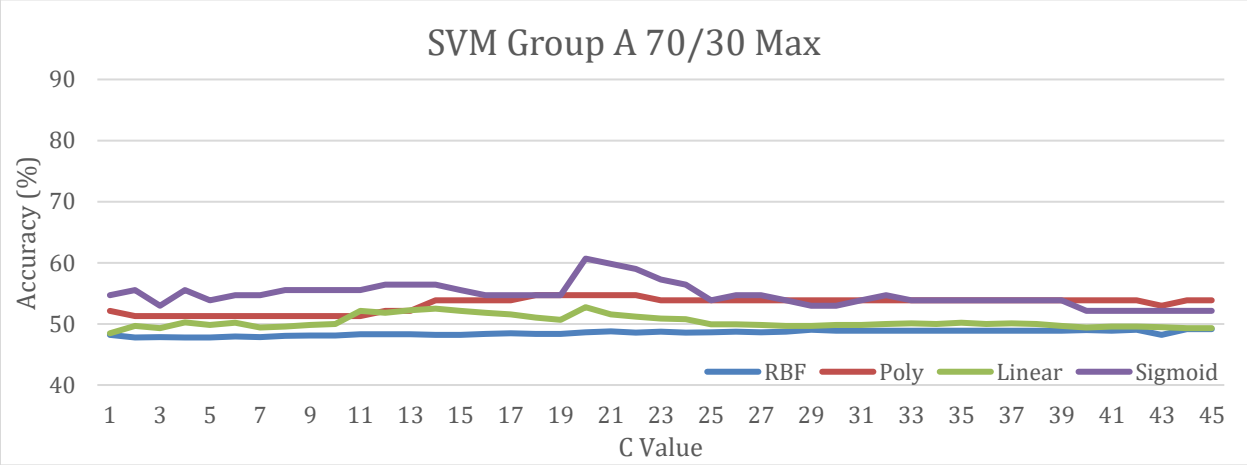

Figure 27: Accuracy of SVM using a 70/30 split on Group A data

3) Apply the SVM to both Group A and Group B and Testing how accurate it can be to predict a single persons data that is not part of the training phase

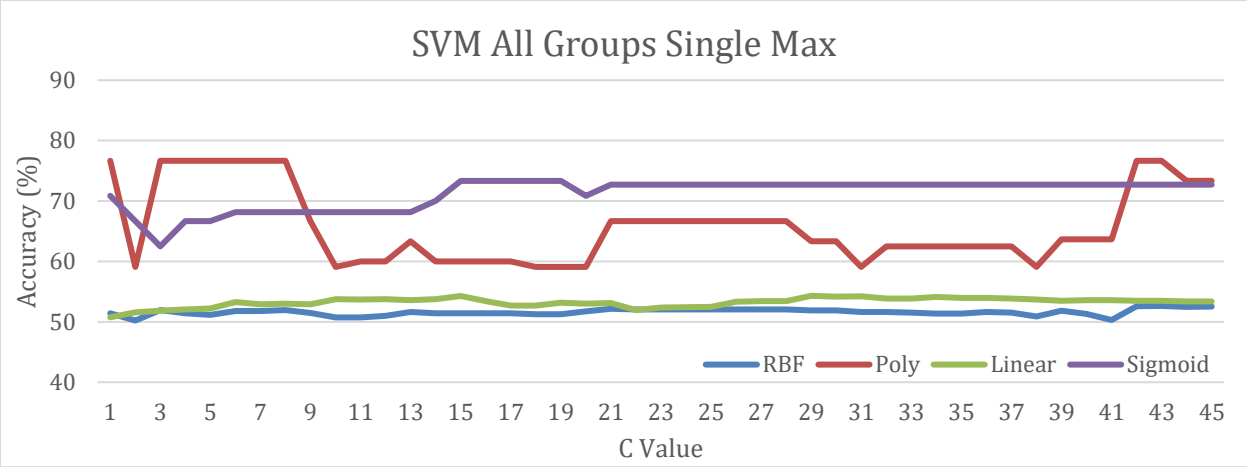

Figure 28: Single Max accuracy using SVM on all groups' data

4) Apply the SVM algorithm on both Group A and Group B data and separating it into a 70/30 ratio between testing and training

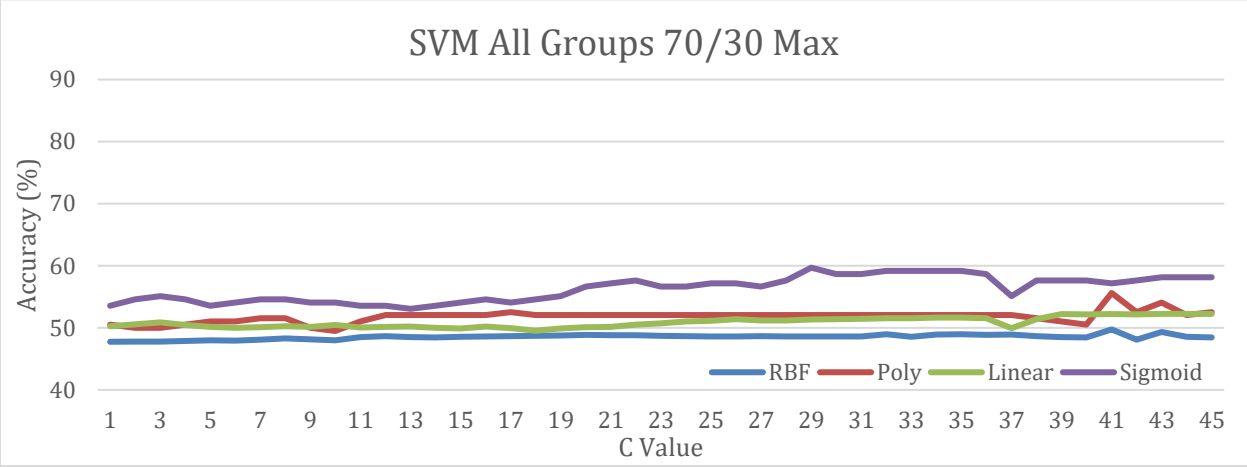

Figure 29: Accuracy of SVM using a 70/30 split on all groups' data

Each chart tested the SVM at various C values ranging from 10 to 10^6. The testing using ratio differences was applied 10 times for each c value. The data was randomised between training and testing each time. Three different kernels were tested alongside no kernel (linear). The 3 kernels compared were the Radial basis function (rbf), polynomial kernel (poly), and the sigmoid kernel. These 3 kernels were picked since they were the 3 that were built into the libsvm library.

A trend that was observed in the graphs below was that the accuracy of the SVM's regardless of the kernel type, increased as the cost value (C value) increased. Although the trend was important, peak accuracy for any single model is also important. Even though the SVM did not show a high accuracy it was still more accurate than the KNN. The first figure shows the average of 10 models for each of the kernel types with varying lengths of c values ranging from 10 to 10^6. The data points in this first test were from group A with a random subset of data put aside for testing the accuracy. The ratio of the data was set to 70 training/30 testing.. The trends confirm that as the C value increased, a higher accuracy was achieved. The results show that some kernels were performing better than other with the highest on average came from the sigmoid kernel.

Comparing the group A to the whole data set was done to see what more data would bring in when doing classification. From the results of doing such a comparison it shows that more data was not useful in getting a higher accuracy. One prediction for this is that SVM are very good at generalization and that it might of been as generalized as it needed to be without having more data. This means that for a study with not as much data, this might be a good classifier to use.

It can be concluded from the graphs above that the maximum accuracy of an SVM can vary greatly between cost values and kernel types. For this reason, it was taken into consideration when selecting the correct SVM layout.

The last classification algorithm tested was Artificial Neural Networks (ANN). The ANN library used for this research was the Tensorflow library for Python. This library was chosen for its optimisations, flexibility, and simplicity. As there are extensive options available when constructing a Neural network using this library, only certain options were tested.

The first test to run with Neural networks was to test to see what depth of the network would do while testing the breadth of the layers. Since the pool of data was quite small for using this type of neural network, it was tested using different variations of the data by retraining with a shuffled variation. The first observation to be made is that the Neural networks were prone to over training.

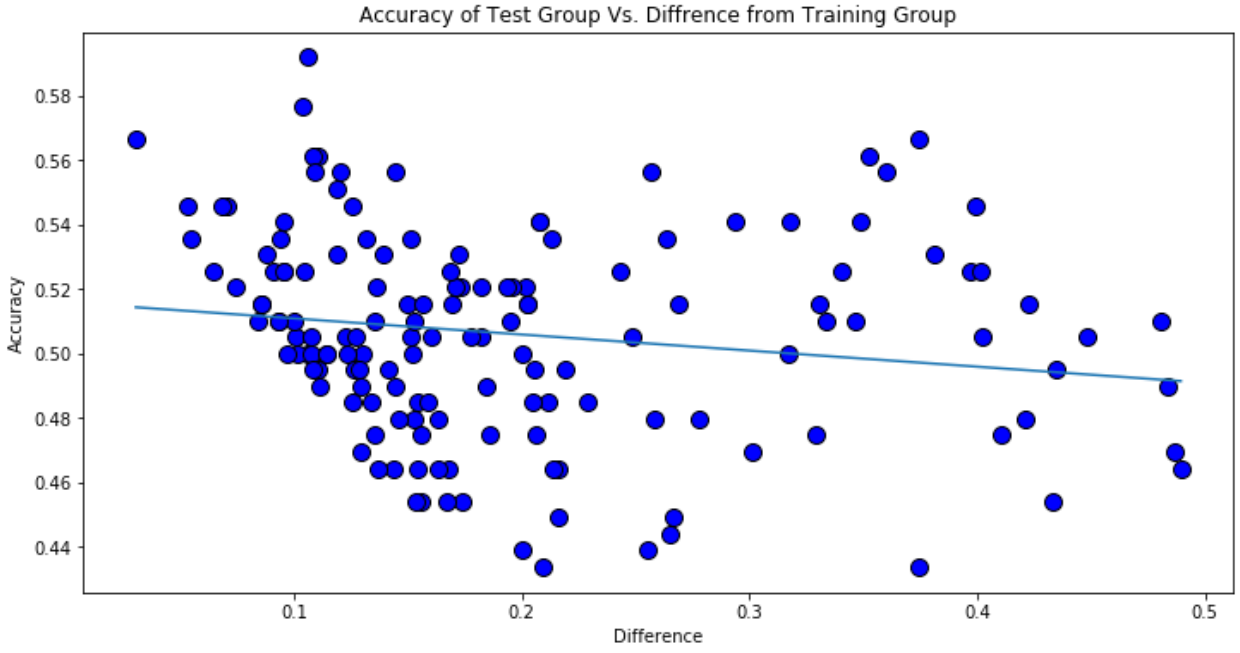

Figure 30: NN Difference in accuracy between training and testing sets

The graph above shows that the predictions from a neural network tend to have better results when the neural network doesn't over-train. This would mean that the network is more generalized. A more generalized network is more useful as it is best at predicting data that it has not seen yet.

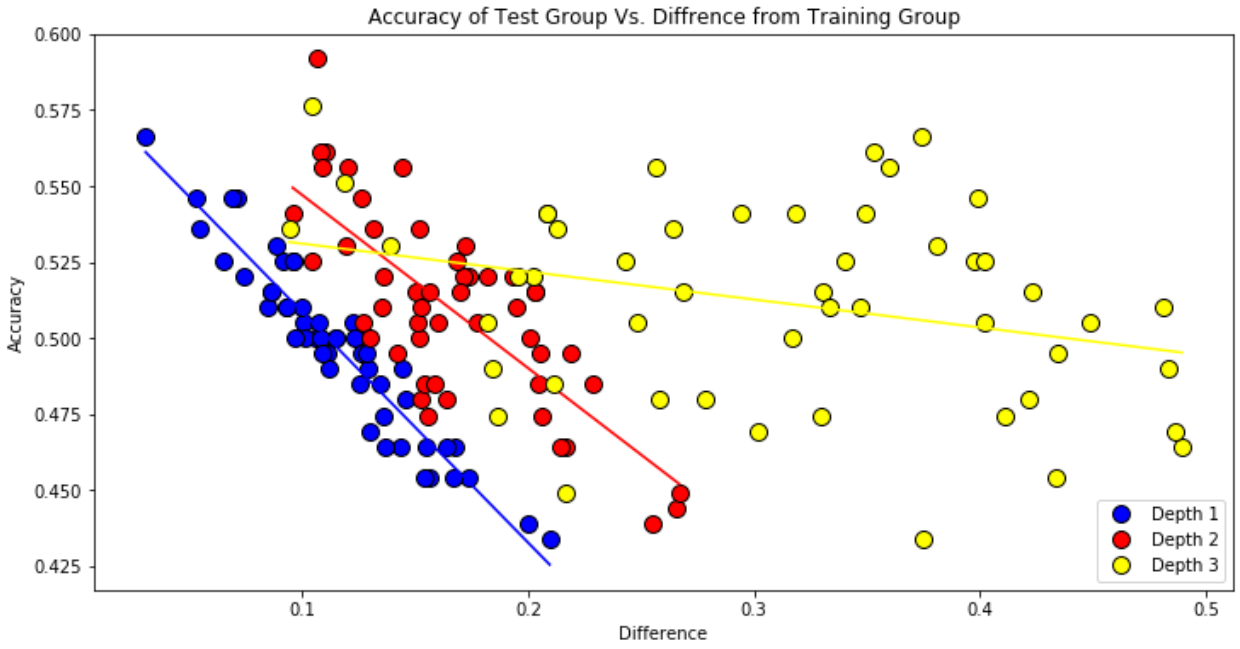

Figure 31: NN Difference in Accuracy between training and testing sets per depth

The next graph shows the same dimensions of accuracy vs difference but by looking at the use of three different hidden layers. The use of a single hidden layer seemed to show over training does affect the accuracy more. The other correlation that can be seen from this graph is that the more layers that are used, the more the difference that can be seen.

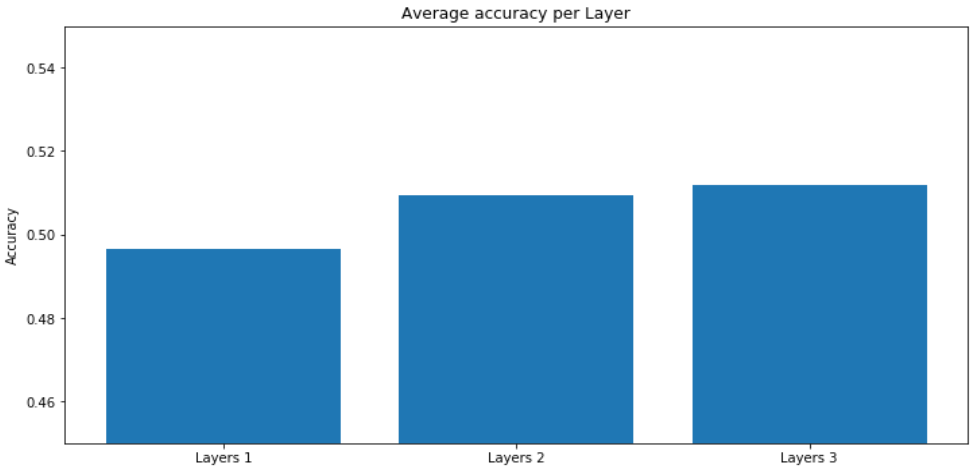

Figure 32: NN Average Accuracy per depth of layers

Even though the overtraining affects the accuracy, the overall average accuracy of the networks does increase. This means there are benefits from increasing the depth of the networks. When this is compared to the average, according to the width of the layers, the trends are not as clear.

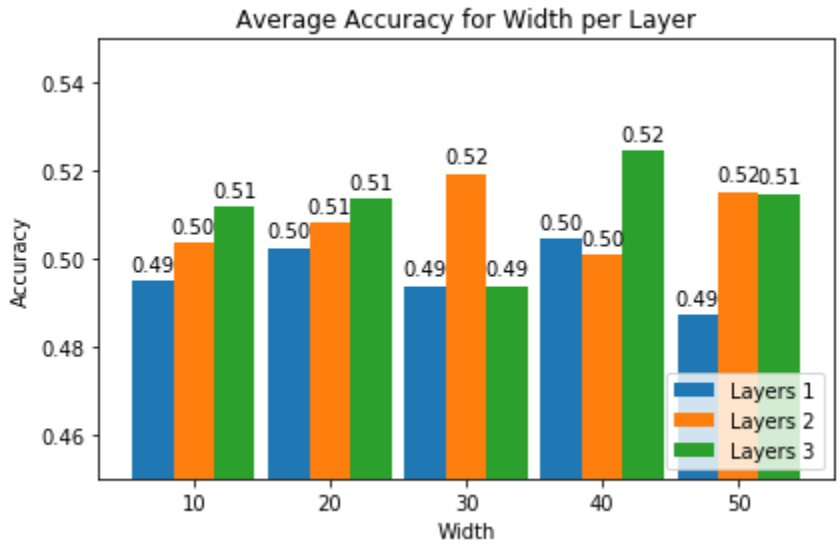

Figure 33: NN Average Accuracy for width of layer

After running multiple tests it was decided that the results were not adequate for this experiment compared to the results from the SVM.

6.2 Challenges

There were many different hurdles that needed to be overcome throughout the research, design and experimenting phase of this thesis experiment. Throughout the next few sections will discuss some of the more major hurdles encountered and any solutions that were found if there were any.

6.2.1 Muse software

One of the most significant problems encountered in this study was the fact that halfway through the experiment the Muse company ceased to support the software necessary to run the real time data forwarding that was necessary for this study. To get around this, it was necessary to download and old version manually as the company did not want to give out their software. Another problem encountered was the instability of their software. The software worked long enough for 24 participants to run through the experiment.

6.2.2 Dirty data needed cleaning

A large part of this thesis was the ability work with EEG data. This led to one of the major challenges that was faced. If you ask any neuroscientist, they will tell you that raw data is not easily used without some sort of pre processing. Because of the nature of this thesis experiment needing the data at runtime automatically, the data also needed to be cleaned automatically. For this reason, many different attempts were made to come up with a method that could account for the noise seen in the focus group testing.

6.2.3 Lack of participants

Although some papers showed a potential benefit of using videogames could increase participant interest, this was not reflected in the recruitment process of this experiment. When recruiting participants using classrooms and clubs, many individuals had shown interest. Although the initial presentation had stirred up many contacts, very few had followed through. when trying to run through some more participants, the connection software became much more unstable leading to participant data being dropped.

6.2.4 Algorithm complexity

When first designing the system, a fully integrated system was envisioned. But as more research had been done and realizing there would need of using systems that are already implemented, a one application system was no longer viable. For this reason, the experiment needed to be made up of multiple programs. This complexity opened the door for more issues and raised the complexity of the system dramatically. this lead to an increase in development time and delayed the testing between the two groups.

7 Conclusion

The experiment was established to prove that using an EEG and a classifier trained on events, that a person's abilities to progress could be predicted and help improve the player experience. Two methods were used to prove this hypothesis: literature review and testing. Several observations were made throughout this research:

1) Certain events can be classified with the proper techniques
2) EEG signals need to be cleaned
3) Frequency domain data is more suitable for this type of experiment
4) If a more general engagement prediction system is to be developed, then multiple games should be used in the experimentation

Given the hypothesis proposed for this experiment, the conclusion is that the results are not statistically significant given the specific implementation. The lack of significance in the results could have come from many different sources. Some key sources of variance are:

1. the classification algorithm
2. accuracy and reliability of the EEG
3. variance in the gameplay
4. amount of data collected

A possible lack of difference between group A and group B could have stemmed the fact that the DDA might have over compensated for the differences that the EEG algorithms were making. A future recommendation would be to analyze each algorithm separately to isolate the modifications done and then compare them.

The results of this research has shown that there are multiple fields of science that have to be included to be able to form a scientifically accurate result. An interesting pattern that came up in papers used for this thesis is that the subject matter was much better approached when the researches were focusing on their area of expertise. This means that a more varied team involved in such an experiment could potentially attain a much better result than any single individual or researcher can do on their own.

7.1 Future Work

One aspect that this thesis experiment can be useful is in the design of the programs and communication. To create a similar experiment, the design aspect shown below could be used as a base and built upon.

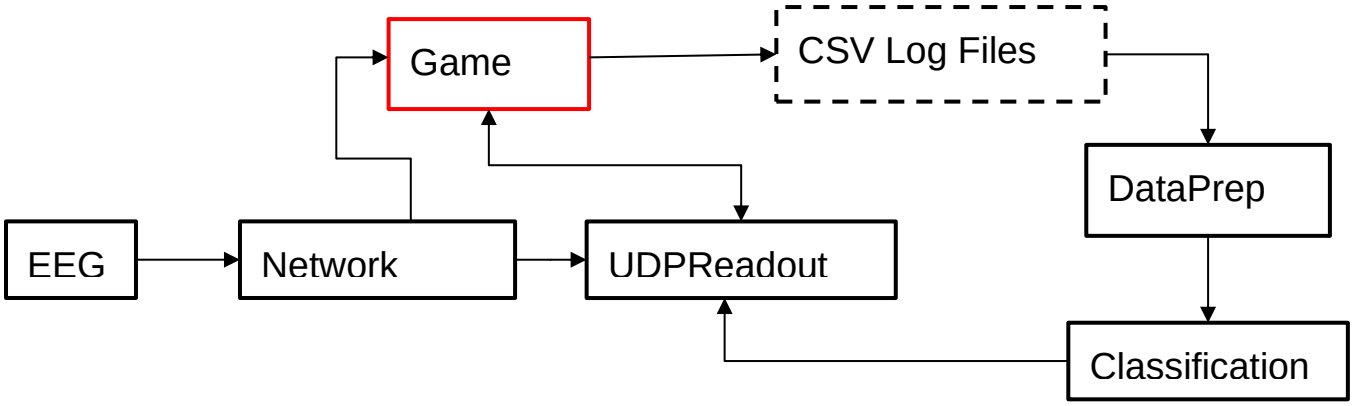

Figure 34: Experiment Blueprint

This research was used to investigate the preliminary aspects of an application using an EEG to improve player experience. Based on this experiment it was determined that improvements could be made by:

a. Using new techniques in the brain-computer interaction field being released regularly.

b. The use of more flexible machine learning algorithms.
c. The use of more data.
d. The use of regression analysis instead of the more simple classification approach

Another area of this research that could be improved in future work is by using a more accurate EEG. Although the Muse EEG headset was very convenient to use, the limited electrode placement and the dry electrodes used could link to more noise than their more accurate wet electrode counterparts. Although more electrodes can alter a players experience with the game more than a more streamlined headband, a similar device with more electrodes could prove to have data that could be more accurate. One alternative to the EEG cap that would allow the participants to focus on the game more would be through the use of a brain sensing VR headset. Since the headset is already going on one's head, the headset would be a potentially better device to collect this type of data.

Another area in which this research could be improved upon is in the choice of the game used as different people prefer different games. Another modification would be to use different difficulty types such as the one expressed in the V3 against the previous versions of Dunjions. Since this game focused more on motor difficulty in the form of combat, a future version of the game could focus more on sensitive or logical difficulty in the form of pathfinding or puzzles.

Another area that could be improved upon in such an experiment is to utilize more participants. With more complicated classification algorithms able to adjust to more intricate patterns, more data is needed. From this preliminary experiment, results showed that when trying to apply this algorithm to small changes, it can be hard to achieve a high rate of accuracy. If this

technique were to be applied to a wider range of emotions and feelings with more subjective questions to train on, it could lead to a more flexible classifier.

The last area that could be improve would be in the software applications themselves. Currently, there are many different programs working in conjunction. A more compact design would be ideal for both the consumer and academic world. In this experiment four programs were run simultaneously to allow the EEG to connect to a video game while performing classification predictions to modify the game in real time. If the simultaneous running programs could be reduced to one or possibly two, it would be much more friendly in a commercial or lab setting that might not be as computer-savvy, leading to even more research in this field (i.e. behavior sciences, video game design, etc.).

8 References

[1] M. E. Smith, A. Gevins, H. Brown, A. Karnik, and R. Du, “Monitoring Task Loading with Multivariate EEG Measures during Complex Forms of Human-Computer Interaction,” *Hum Factors*, vol. 43, no. 3, pp. 366–380, Sep. 2001.

[2] K. Poels, “STREP / NEST-PATH Deliverable D3.3: GAME EXPERIENCE QUESTIONNAIRE,” p. 47.

[3] L. E. Nacke, “Wiimote vs. controller: electroencephalographic measurement of affective gameplay interaction,” 2010, p. 159.

[4] “FUGA The fun of gaming: Measuring the human experience of media enjoyment” May 2009. Accessed: Nov. 18, 2021. [Online]. Available: http://project.hkkk.fi/fuga/.

[5] M. Salminen and N. Ravaja, “Oscillatory Brain Responses Evoked by Video Game Events: The Case of Super Monkey Ball 2,” *CyberPsychology & Behavior*, vol. 10, no. 3, pp. 330–338, Jun. 2007.

[6] T. McMahan, I. Parberry, and T. D. Parsons, “Evaluating Electroencephalography Engagement Indices During Video Game Play,” p. 5.

[7] M.-V. Aponte, G. Levieux, and S. Natkin, “Difficulty in Videogames: An Experimental Validation of a Formal Definition,” in *Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology*, New York, NY, USA, 2011, pp. 49:1–49:8.

[8] A. Rajavenkatanarayanan, A. R. Babu, K. Tsiakas, and F. Makedon, “Monitoring Task Engagement Using Facial Expressions and Body Postures,” in *Proceedings of the 3rd International Workshop on Interactive and Spatial Computing*, New York, NY, USA, 2018, pp. 103–108.

[9] Q. C. Lam, L. A. T. Nguyen, and H. K. Nguyen, “A Novel Approach for Classifying EEG Signal with Multi-Layer Neural Network,” in *Proceedings of the 2017 International Conference on Robotics and Artificial Intelligence*, New York, NY, USA, 2017, pp. 79–83.

[10] K. C. Ewing, S. H. Fairclough, and K. Gilleade, “Evaluation of an Adaptive Game that Uses EEG Measures Validated during the Design Process as Inputs to a Biocybernetic Loop,” *Front Hum Neurosci*, vol. 10, May 2016.

[11] B. Kerous, F. Skola, and F. Liarokapis, “EEG-based BCI and video games: a progress report,” *Virtual Reality*, pp. 1–17, Oct. 2017.

[12] K. Rajamani, A. Ramalingam, S. Bavisetti, and M. Abujelala, “CBREN: Computer Brain Entertainment System Using Neural Feedback Cognitive Enhancement,” in *Proceedings of the*

10th International Conference on PErvasive Technologies Related to Assistive Environments, New York, NY, USA, 2017, pp. 236–237.

[13] M. Abujelala, C. Abellanoza, A. Sharma, and F. Makedon, "Brain-EE: Brain Enjoyment Evaluation Using Commercial EEG Headband," in *Proceedings of the 9th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, New York, NY, USA, 2016, pp. 33:1–33:5.

[14] É. Labonté-LeMoyne *et al.*, "Are We in Flow Neurophysiological Correlates of Flow States in a Collaborative Game," in *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, New York, NY, USA, 2016, pp. 1980–1988.

[15] F. Lotte, M. Congedo, A. Lécuyer, F. Lamarche, and B. Arnaldi, "A review of classification algorithms for EEG-based brain–computer interfaces," *J. Neural Eng.*, vol. 4, no. 2, p. R1, 2007.

[16] J. Gruzelier, A. Inoue, R. Smart, A. Steed, and T. Steffert, "Acting performance and flow state enhanced with sensory-motor rhythm neurofeedback comparing ecologically valid immersive VR and training screen scenarios," *Neuroscience Letters*, vol. 480, no. 2, pp. 112–116, Aug. 2010.

[17] C. Lin *et al.*, "Adaptive EEG-Based Alertness Estimation System by Using ICA-Based Fuzzy Neural Networks," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 53, no. 11, pp. 2469–2476, Nov. 2006.

[18] G. Chanel, C. Rebetez, M. Bétrancourt, and T. Pun, "Boredom, Engagement and Anxiety As Indicators for Adaptation to Difficulty in Games," in *Proceedings of the 12th International Conference on Entertainment and Media in the Ubiquitous Era*, New York, NY, USA, 2008, pp. 13–17.

[19] A. Subasi and E. Erçelebi, "Classification of EEG signals using neural network and logistic regression," *Computer Methods and Programs in Biomedicine*, vol. 78, no. 2, pp. 87–99, May 2005.

[20] K. Katahira, Y. Yamazaki, C. Yamaoka, H. Ozaki, S. Nakagawa, and N. Nagata, "EEG Correlates of the Flow State: A Combination of Increased Frontal Theta and Moderate Frontocentral Alpha Rhythm in the Mental Arithmetic Task," *Front. Psychol.*, vol. 9, 2018.

[21] T.-P. Jung *et al.*, "Extended ICA Removes Artifacts from Electroencephalographic Recordings," in *Advances in Neural Information Processing Systems 10*, M. I. Jordan, M. J. Kearns, and S. A. Solla, Eds. MIT Press, 1998, pp. 894–900.

[22] R. E. Wheeler, R. J. Davidson, and A. J. Tomarken, "Frontal brain asymmetry and emotional reactivity: A biological substrate of affective style," *Psychophysiology*, vol. 30, no. 1, pp. 82–89, 1993.

[23] Louis A. Schmidt and Laurel J. Trainor, “Frontal brain electrical activity (EEG) distinguishes valence and intensity of musical emotions,” *Cognition and Emotion*, vol. 15, no. 4, pp. 487–500, Jul. 2001.

[24] G. Cheron, “How to Measure the Psychological ‘Flow’? A Neuroscience Perspective,” *Front. Psychol.*, vol. 7, 2016.

[25] M. Klasen, R. Weber, T. T. J. Kircher, K. A. Mathiak, and K. Mathiak, “Neural contributions to flow experience during video game playing,” *Soc Cogn Affect Neurosci*, vol. 7, no. 4, pp. 485–495, Apr. 2012.

[26] M. Andujar and J. E. Gilbert, “Let’s Learn!: Enhancing User’s Engagement Levels Through Passive Brain-computer Interfaces,” in *CHI ’13 Extended Abstracts on Human Factors in Computing Systems*, New York, NY, USA, 2013, pp. 703–708.

[27] D. W. Harrison, “Arousal Syndromes: First Functional Unit Revisited,” in *Brain Asymmetry and Neural Systems: Foundations in Clinical Neuroscience and Neuropsychology*, D. W. Harrison, Ed. Cham: Springer International Publishing, 2015, pp. 61–84.

[28] R. M. Beckstead, *A survey of medical neuroscience*. New York: Springer-Verlag, 1996.

[29] Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

[30] E. Tretkoff, “October 1958: Physicist Invents First Video Game,” *American Physical Society*, Oct-2008. [Online]. Available: https://www.aps.org/publications/apsnews/200810/physicshistory.cfm.

[31] NewZoo Free 2016 Global Games Market Report [electronic resource], NewZoo Games, [2016]http://resources.newzoo.com/hubfs/Reports/Newzoo_Free_2016_Global_Games_Market_Report.pdf

[32] “Recognizing the Degree of Human Attention Using EEG Signals from Mobile Sensors.” [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3812603/#.

[33] InteraXon Inc (2019). *Dear Muse Developers*. Retrieved from https://choosemuse.com/development/.

[34] R. M. Ryan, C. S. Rigby, and A. Przybylski, “The Motivational Pull of Video Games: A Self-Determination Theory Approach,” *Motiv Emot*, vol. 30, no. 4, pp. 344–360, Dec. 2006.

[35] E. Laura and M. Frans, “Fundamental Components of the Gameplay Experience: Analysing Immersion,” 2005.

[36] P. Sweetser and P. Wyeth, “GameFlow: A Model for Evaluating Player Enjoyment in Games,” *Comput. Entertain.*, vol. 3, no. 3, pp. 3–3, Jul. 2005.

[37] M. Ahn, M. Lee, J. Choi, and S. C. Jun, “A Review of Brain-Computer Interface Games and an Opinion Survey from Researchers, Developers and Users,” *Sensors (Basel)*, vol. 14, no. 8, pp. 14601–14633, Aug. 2014.

[38] “Check out how much a computer cost the year you were born.” https://www.usatoday.com/story/tech/2018/06/22/cost-of-a-computer-the-year-you-were-born/36156373/ (accessed Nov. 18, 2021).

[39] W. O. Tatum, *Handbook of EEG interpretation*. New York [N.Y.: Demos Medical Pub., 2013.

[40] G. Di Flumeri, P. Aricò, G. Borghini, N. Sciaraffa, A. Di Florio, and F. Babiloni, “The Dry Revolution: Evaluation of Three Different EEG Dry Electrode Types in Terms of Signal Spectral Features, Mental States Classification and Usability,” *Sensors*, vol. 19, no. 6, p. 1365, Mar. 2019.

[41] “The Internet Protocol Stack.” https://www.w3.org/People/Frystyk/thesis/TcpIp.html (accessed Nov. 18, 2021).

[42] M. Csikszentmihalyi, “Flow: The Psychology of Optimal Experience,” 1990.

[43] “Unity Platform Roadmap.” https://unity.com/roadmap/unity-platform (accessed Nov. 18, 2021).

[44] P. M. Conn, Ed., Neuroscience in Medicine. Totowa, NJ: Humana Press, 2003. 623-633

[45] “Keep Calm and Play On: Video Games That Track Your Heart Rate | MIT Technology Review.” https://www.technologyreview.com/2015/09/21/72421/keep-calm-and-play-on-video-games-that-track-your-heart-rate/ (accessed Nov. 18, 2021).

[46] N. Burch and H. L. Altshuler, *Behavior and Brain Electrical Activity*. Boston, MA: Springer US, 1975.

[47] H. Banville and T. H. Falk, “Recent advances and open challenges in hybrid brain-computer interfacing: a technological review of non-invasive human research,” *Brain-Computer Interfaces*, vol. 3, no. 1, pp. 9–46, Jan. 2016.

[48] “Engagement Definition & Meaning - Merriam-Webster.” https://www.merriam-webster.com/dictionary/engagement (accessed Nov. 18, 2021).

[49] D. Vallieres, “Achieving Flow in Gameplay through a Dynamic Difficulty Adjustment System,” p. 87.

[50] G. Chanel, C. Rebetez, M. Bétrancourt, and T. Pun, “Emotion Assessment From Physiological Signals for Adaptation of Game Difficulty,” *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 41, pp. 1052–1063, Dec. 2011, doi: 10.1109/TSMCA.2011.2116000.

[51] “What is the kernel trick? Why is it important? | by Grace Zhang | Medium.” https://medium.com/@zxr.nju/what-is-the-kernel-trick-why-is-it-important-98a98db0961d (accessed Nov. 18, 2021).

[52] “Brandguide.” https://brandguide.brandfolder.com/unity/downloadbrandassets (accessed Nov. 22, 2021).

[53] “Product - Google Drive.” https://drive.google.com/drive/folders/15pWkI6-CIIi-U8Y7d2yA8rkSYGMCy8Ch (accessed Nov. 22, 2021).

[54] “SVM margin” https://commons.wikimedia.org/wiki/File:SVM_margin.png (accessed Nov. 22, 2021)

[55] A. Walker-McBay, "*Mike Booth, the Architect of Left 4 Dead's AI Director, Explains Why It's So Bloody Good*," Kotaku Australia, 2018. [Online]. Available: https://www.kotaku.com.au/2018/11/mike-booth-the-architect-of-left-4-deads-ai-director-explains-why-its-so-bloody-good/. [Accessed: 08-Jun-2022].

9 Appendices

9.1 Software Versions

- Python 3.6.3
- C # .net 4.0
- Unity3D 2019.2.0
- Windows 10
- Muse Direct 0.19.1
- Muse Lab 1.6.3

9.2 Hardware

- Intel i7-3770 3.4GHz
- 8 GB RAM
- Muse 2016
- XBox 1 Controller

9.3 In-game Experience Questionnaire

Please indicate how you felt while playing the game for each of the items, on the following scale:

Not at all	Slightly	Moderately	Fairly	Extremely
0	1	2	3	4

1) I was interested in the game's story
2) I felt successful
3) I felt bored
4) I found it impressive
5) I forgot everything around me
6) I felt frustrated

7) I found it tiresome
8) I felt irritable
9) I felt skillful
10) I felt completely absorbed
11) I felt content
12) I felt challenged
13) I had to put a lot of effort into it
14) I felt good