

**Dissertation**

**Evaluating the effectiveness of using Bio-feedback to affect Gameplay**

**Michael Little**

**Word count Without References / Appendix: 9320**

**Table of Contents:**

1. **Abstract**
2. **Introduction**
3. **Project Aim**
4. **Literature Review**

**4.1 History of EEG Devices and BCI’s**

**4.2 Emotiv EPOC**

**4.3 SFML**

**4.4 Using Games to trigger Emotions**

**4.5 Using EEG devices in games**

**4.6 Adaptive Gameplay in games**

**4.7 Enjoyment affecting a person’s sense of time**

1. **Methodology**

**5.1 Development**

**5.2 Implementation**

**5.2.1 Emotiv**

**5.2.2 Screen Manager**

**5.2.3 Input Manager**

**5.2.4 Shooter Screen**

**5.2.4.1 Player**

**5.2.4.2 Enemy**

**5.2.4.3 Boss**

**5.2.4.4 Bullet/Enemy Fire**

**5.2.5 Emotiv Shooter Screen**

**5.2.4.1 Emotiv Update Thread**

**5.2.4.2 Global Variables Class**

**5.2.4.3 Get Emotiv Method**

**5.2.6 Integration of data**

**4.3 Gameplay**

**4.4 Testing**

1. **Results**

**6.1 Data Obtained from Testing**

1. **Discussion**
2. **Conclusion and Future Development**

**8.1 Conclusion**

**8.2 Future Development**

1. **References**
2. **Appendices**

**9.1 Appendix 1: Code Snippets**

**9.2 Appendix 2: Questionnaires for testing**

**Figures:**

**Figure 1: Space Shooter Screenshot**

**Figure 2: Emotiv Control Panel – No Connectivity**

**Figure 3: Emotiv Control Panel – Perfect Connectivity**

**Figure 4: User Experience Chart**

**Figure 5: Preferred Game Chart**

**Figure 6: Game Order Chart**

**Figure 7: Which Ended First Chart**

**Figure 8: Average Times**

**Figure 9: Higher Score Ratio Chart**

**Figure 10: Average Emotional Rating Chart**

**Figure 11: Average Emotional Score Chart**

1. **Abstract**

The purpose of this project was to evaluate the effect of using a user’s own biofeedback to affect the gameplay of a specific game, and to see whether or not this would have a positive effect on the users experience as a whole.

The project featured the use of an Electroencephalogram (EEG) device, which allows us to gain high resolution measurements of a human’s emotions, thoughts and feelings.

The projects development consisted of two parts, the first is creating an application that was able to take in readings from an EEG device, in this case the Emotiv EPOC headset, and translate it in real time into a format that could be used to interact with the code of a video game. Secondly is to create a simple yet entertaining game that would make use of the user’s feedback.

This application was then tested on several different candidates, they would play both the Emotiv and control versions of the application, while wearing the headset and after each game they filled out a questionnaire about said application. The data taken from these sessions was used to form the following hypothesis.

After all the data was processed and analysed it was discovered that the majority of the players preferred the Emotiv version and were able to perform better and achieve a higher score. The emotional data gathered from the headset and questionnaires also suggests that the users on average reacted better to the Emotiv version on an emotional level and seemed to achieve a better sense of flow due to a difference in their sense of time passing.

Keywords: Electroencephalography (EEG), Flow

1. **Introduction**

Videogames have been constantly evolving since their conception, from gameplay to graphics videogames are always trying to one up themselves. From Spacewar! (Steve Russel. 1961) to Super Mario Bros. (Nintendo. 1985) games have constantly improved and re-interoperated what it means to be a game and continue to do so today. Within the past decade games have started to tell compelling narratives, and try to affect the player emotionally, games like Gone Home (The Fullbright Company, 2013) and Dear Esther (The Chinese room, 2012) aren’t played to get a high score, they are played to have the user immerse themselves in a world that explores themes that aren’t usually explored in games.

Another aspect of technology that has also been developing is the field of Electroencephalography (EEG), modern variants of these devices have been in use since the mid 70’s and were used to read the electric signals in patient’s brains. This helped medical science achieve a much greater understanding of how the human brain works, and what signals are sent when the users are confronted with certain stimuli. This technology has been perfected to the point that it is now available for manufacture and general purchase, the Emotive EPOC headset is one such example of this.

Intentionally or not Games do trigger an emotional response from us, so would using these to affect the way we play our games be a good thing? Previous examples of exploring the way we play games have been met with praise and have helped us come up with new and innovative ways to play (e.g. Nintendo Wii’s motion controls) so it’s entirely possible that using the players emotional state could be the next big thing to change the industry.

The ability to read the users emotional state and use it in our games design has almost limitless possibilities: from our heart rate, horror games could craft an even more horrifying experience. From our anger, Hack ‘n’ Slash games could have our attacks decimate our enemies. And from our excitement, games could craft us a tailored experience that only gets better the more we play.

The aim of this project it to find out whether or not these theories had any truth to them, by creating an application that would use data gathered by the Emotive EPOC headset and change its gameplay based on the users emotional state. This application will then be tested on a group of candidates and a conclusion will be crafted using the gathered results.

1. **Project Aim**

**“Does using Bio-feedback to affect gameplay, have a positive effect on the player’s experience?”**

EEG devices are no longer just for medical use, they provide us with a better understanding of how our brain reacts to certain stimuli and this data is useful if hundreds of different fields, from marketing to music, we can now see how people react on a neurological level, and now thanks to the Emotiv EPOC this technology has now come to videogames.

Objectives that can be derived from this question are:

* Design an application that can retrieve the emotional data at real time and output the data.
* Design a game that is simple and fun to play.
* Combine the previous steps into an application that can modify itself in real time to a user’s data.
* Design a questionnaire to help evaluate the user’s experience.
* Evaluate feedback and draw conclusion

1. **Literature Review**

Development of this project was built upon the techniques taught in the Computer Games Application Development course at Abertay University, and personal research done for the module throughout the year.

The Application was built in Microsoft Visual Studio and uses the Emotiv EPOC Software Development Kit (SDK) and SFML (Simple and Fast Multimedia Library ver.2.2). These choices allowed the application to easily run on different machines, as well as allow for simple and easy implementation of the Emotiv headset and its various features.

* 1. **– History of EEG Devices and BCI’s**

Brain-Computer Interfaces (BCI’s) and ElectroEncephaloGram (or EEG) are based upon Hans Bereger’s research into the human brain (David Millett, 2001). Hans discovered that the brain gave off electrical signals, he then used this to record human brain activity through an EEG, and thanks to this research he was able to help science gain a better understanding of the human brain as well as how brain diseases can affect our brainwaves.

The first BCI were fairly primitive by today’s standards, they consisted of silver wires being inserted under the patient’s scalp, but thanks to advances in technology BCI’s no longer have to be so invasive and are now much more readily available and just need to be in contact with the scalp to get clear readings.

Modern EEG device has been in medical use since the mid 70’s, they are used to gather electrical readings from the users brain by placing electrodes coated in electricity conducting gel on the users scalp. EEG’s are a popular non-invasive in the medical community due to their accurate readings, ease of use and low set up costs. The downside to this method comes from their susceptibility to noise which can give mixed readings (Harold Shipton, 1975).

EEG devices have been used in various situations ranging from:

* Clinical: to help diagnose patients suffering from seizures, comas, etc.
* Research: to help better understand the human psychology, neuroscience, etc.
* Everyday use: videogames and market research.

(David Millet. 2001) A look into Hans Berger’s life and his research into EEG and how he developed it over 40 years of research and the events that put him on the path of research.

(Harold Shipton. 1975) A historical look at the development of the EEG from Hans Berger’s original discovery to modern (at the time) clinical EEG devices.

* 1. **– Emotiv EPOC**

The Emotiv EPOC headset is an ElectroEncephaloGram device that’s available for general purchase, the headset has been gaining popularity in the technological and medical circles as a accurate piece of hardware that is cost effective, readily available and can provide a valid alternative to the current medical standard (Badock et al, 2013). The headset was developed by Australian electronics company, Emotiv Systems, the headset was specifically designed to be used as a peripheral for gaming.

(Badock et al, 2013) A look into using the Emotiv EPOC to record auditory Event Related Potentials (ERPs) as opposed to standard laboratory EEG equipment. The research shows that the EPOC headsets provide a valid alternative to the current standard.

* 1. **– SFML**

SFML (Simple and Fast Multimedia Library) (Laurent Gomila, 2007) is an open-source cross-platform software development library that is designed to provide a simple interface to various components in computers, to provide ease in developing games and other multimedia applications. The library is written in C++ but has versions for several other coding languages including Java and Python, for this project, the latest stable build at the time of this project was used (Ver 2.2).

In this project SFML is used to render windows, 2D graphics, draw text, handle collision detection, create an asynchronous thread to improve performance and mutex’s to help stop the data becoming corrupted, as well as many other things that helped bring this project to fruition. These Libraries were very easy to integrate and also had a strong online database and community to assist in any development problems that cropped up along the way.

(Laurent Gomila, 2014) A website dedicated to SFML, contains links to download and a library of resources explaining its various features.

* 1. **– Using Games to Trigger Emotions**

Ever since their inception Videogames have always triggered emotions; from the joy of beating a boss, to the frustration of dying from a boss for the fifth time. But as technology advanced and new options presented themselves, developers are now able to start giving a greater focus to story, visuals and audio, games have begun exploring a much wider range of themes. These days there are several games that have begun standing out from the crowd by not giving us a power fantasy or having us rescue a princess, but from having us explore concepts of love and loss (E.G. Gone Home, To the Moon, Dear Esther) this shows us that the medium is evolving and taking full advantage of its ability to invoke emotions, maybe now we should look into how these emotions can be used to benefit the way we play our games?

There are several ways games, like other mediums before it, connect with us on an emotional level. Similar to literature early games that were limited on graphics and audio due to the hardware constraints would try to evoke emotional response through writing, these include Text based games, some of which had no graphics or sound so they had to rely on writing to guide the player. As technology advanced we could begin to start adding higher quality audio and graphics, this helped lead to games that would try to emulate movies. These were called FMV (Full Motion Video) games and they were very popular during the early 90’s. Cut to today when the games industry has reached the point where we can have published writers, high definition graphics and crystal clear audio, game production has reached the point where we can have amazing fully rendered cut scenes created in the games engine as well as beautiful fully orchestrated scores with comparable quality to modern cinema.

As a medium games have always had one thing that was uniquely theirs, and this is something that’s been around from the beginning, Interactivity. The ability to give the user a choice and have their actions play out is something that has helped define games as a medium, and continues to be present in our games today. From simple things like controlling when to jump in a platformer, too much grander things like deciding the fate of a characters life or the future of an entire civilisation, interactivity in games can provide users with an experience that no other medium can replicate. And thanks to production quality only getting better, games could one day overtake all other mediums and become the penultimate form of art and entertainment.

(Harold Goldberg, 2012) An article on the 25 Greatest Breakthroughs throughout the history of Videogames, goes into good detail about the early years of videogames.

(Frome, 2007) A detailed look into the various ways we emotionally interact with our games. May help development of the application by helping predict how the users may react to certain features of the application.

* 1. **– Using EEG devices in games**

There have been a few examples of EEG’s being used with games, these range from using the players emotional states as a form of second controller to play the game (Nacke et al., 2011), or from using it to measure how a player responds to certain situations and to measure a players sense of flow (Cowley et al. 2008).

For this project we will be using the Emotiv EPOC to do a combination of the both, so the application will use the players emotional data to affect the gameplay like a second controller but in a way that tries to ensure the player has a good sense of flow throughout the entire game, and thanks to the Emotiv EPOC being specifically designed to be used as a peripheral in games, the ability to have our games be effected by our emotions has never been easier.

(Nacke et al., 2011) A look into using user Biofeedback to have Direct and Indirect control over a game and whether or not it has a positive effect on the user’s experience. The paper details various ways they had the user be able to affect the game directly/indirectly utilising an EEG headset and a Bio-harness. The results showed that the features the users could directly control were positively received; the indirect ones weren’t as well received. Could be due to poor implementation as several of these features either didn’t directly affect the gameplay (e.g. background graphics) or were just a hassle to the user, as some users ended up trying to trick the system by purposefully speeding up their heartrates, this warrants investigation into better ways the Emotional data can be better integrated.

(Cowley et al. 2008) A detailed look into flow; what it is, how it happens and how different users experience it in different ways.

* 1. **– Adaptive Gameplay in Games**

As the games market continues to grow bigger and more diverse developers have had to implement new and unique features to help their games stand out from the competition, one example of this is the inclusion of adaptive gameplay that changes the game based on the players playstyle/skill so that each play through is different. There are a few examples of this in modern games these games feature adaptive difficulty that change based on the player’s strength or weakness in order to help create a unique and tailored experience.

As games can come in many different forms the ability to adapt the gameplay to suit the players can sometimes be easy or difficult to accomplish. An easy example of this can be found in the MMORPG (Massively Multiplayer Online Role Playing Game) RIFT (Trion Worlds, 2011) players take the role of a character and continue to level them up and gain better equipment as such all players are essentially governed by numbers. In the game there are events know as Rifts (hence the name) these events will randomly spawn around the world or can be summoned by a player. These Rifts create an encounter for the player(s) to fight, they spawn monsters and create objectives based on the player’s strength, but since these events take place in a game world that features hundreds of other players the designers had to include some conditions so that these events can be enjoyed by everyone. The first step was to make sure that every area of the world is divided into a level range (e.g. level 10 – 12) then they implemented a feature to change the players strength so that their scaled to the difficulty of the encounter. These feature help the game be constantly accessible to low level players while allowing veteran players have the ability to take part and have an element of challenge.

A much more famous and complex example of Adaptive gameplay would be the AI Director found in the game Left 4 Dead (Valve, 2008), this game is a co-operative First Person shooter where a team of up to 4 players have to make it from point A to B while surviving waves of zombies. Unlike RIFT this game features no levelling system or stronger equipment, instead the players are given a pre-set character and have to use skill and teamwork to complete their objective (Booth, 2007). To allow the game to be appealing to new and skill players alike the designers created a feature known as the AI Director, this is essentially an artificial intelligence that strives to provide each game with a unique and tailored experience. The Director does this by spawning enemies and items based on how well the players are doing, for example if a group of players are doing well the director will begin spawning more and more enemies for the players to fight and making helpful items less available, and vice versa for players not doing as well. The director is also has the ability to spawn enemies specifically designed to disrupt the players plans and encourage them to try and get to the objective as soon as possible. One example of this would be if the players are choosing to hold up in a single location rather that continually moving around, the director could choose to spawn and enemy known as the Smoker, who is able to pull players towards them and disable them, this encourages the other players to leave their safe area and rescue their fellow teammates.

A recent example of adaptive gameplay in games can be found in the Rocksmith (Ubisoft. 2012) games. Rocksmith is a series of music rhythm games where the player has to play a real guitar to play the game. It features several licenced tracks that the player can play on demand, once selected it begins playing the song and a track with notes begins heading towards the player, the player then has to play the specific note on the guitar to the beat of the song. The game features an adaptive difficulty that finds how skilled the player is and gradually starts throwing more notes/techniques into the song as the player’s skill improves (Ubisoft. 2011). This system while not perfect actually helps less skilled players slowly but steadily learn how to play the guitar for real, while not throwing anything that the player cant handle at them.

(Booth, 2007) A detailed look into the various systems in place in Left 4 Dead, including the logic behind the AI Director.

(Ubisoft. 2011) A promotional video detailing Rocksmith’s adaptive difficulty system, and how it adapts based on the players level of skill.

* 1. **- Enjoyment affecting a person’s sense of time**

The saying “Doesn’t time fly when you’re enjoying yourself?” comes from the fact that human beings tend to perceive time differently depending on whether or not were enjoying ourselves. This is why a 6 hour shift at work can seem to take much longer than playing your favourite game for 12 hours, this is something that games, especially long ones, bank on to keep the user’s playing them.

A term for this phenomenon that is thrown around in psychology circles is Flow or the zone, this is a mental state where a person is fully immersed in a task and has a feeling of energised focus and an enjoyment for the activity (Mihaly Csikszentmihalyi. 2014). This term was originally named by [Mihály Csíkszentmihályi](http://en.wikipedia.org/wiki/Mihaly_Csikszentmihalyi) the former head of the department of psychology at the University of Chicago, he referred to flow as focused motivation and it may represent the ultimate experience in harnessing emotions in the service of performing and learning.

A good example of this would be RPG’s, like the game RIFT (mentioned in section 4.6), this game is designed to have the player complete hundreds of quests in the aim of getting stronger. On paper the gameplay of these games can seem very boring and repetitive, go here kill X amount of these, and then go there collect X amount of that. Yet millions of players play games like these and end up logging in days and days of gameplay, the reason for this is simple, they are designed to give just enough enjoyment to the player to have them keep them playing but not so much that they get used to it. While this isn’t the most morally agreeable method as most games that use this design end up putting the player in a repetitive loop that becomes more about waiting than actually playing a game, and even worse some games use this to entice players to spend real money to get out of this loop (Stuart Dredge. 2015). This genre of games has been given the title of Freemium, as they are usually free to play but are littered with micro transactions, despite all this the genre proven very successful in the past, and though games these days have begun to trend away from these business methods by trying to encourage players to keep playing through good gameplay etc., there are still hundreds of them available on services like iTunes (Apple inc. 2001) and Google play (Google. 2008).

(Mihaly Csikszentmihalyi. 2014) The second volume of collected works of Mihaly Csikszentmihalyi, including his research into Flow, attention and the other elements of Positive Psychology. The book also goes into detail about what motivates us as a species and how we have developed from a simple need to fulfil our primitive motivations.

(Stuart Dredge. 2015) An article looking into how several Freemium games have begun focusing more on monetization and less about actual gameplay, and its effect on the mobile gaming market.

1. **Methodology**
   1. **– Development**

The purpose of this section is to explain how the application was developed and why certain design choices were chosen.

As mentioned earlier the application was developed in Microsoft Visual Studio, this was a change to the proposed plan that came early in the applications development, as originally the application was supposed to be developed in Unity, a game engine designed to make development of games easier and more streamlined. This was changed as integrating the Emotiv Epoc SDK and libraries into standard Unity was proving to be too difficult and time consuming thus development was switched to Visual Studio and Integrated Development Enviroment (IDE) which is specifically designed to aid programming in several languages. This choice would allow the libraries to be integrated and handled much easier, and would also allow classes to be directly taken from the Emotiv SDK and integrated straight into the project. The next addition to development was the choice to use SFML, this was initially used to easily render windows that would hold the Emotiv EPOC data but was then expanded to handle graphics and collision on the gameplay side of the application.

Development of the Project began with the Emotiv Class, this class was based on code provided with the Emotiv SDK and would be used as a mediator between the Headset and rest of the application as it could take the raw data from the headset and convert it into a format that was easy to use. The data from this was then displayed updated in real time in a separate window, this helped show whether or not the headset data was being read and passed along correctly. After some time getting to better grips with the Emotiv software the data was being retrieved and displayed in real time, this meant that development could switch to the Game side of the application.

Before this happened it was proposed that a Game Development Engine such as Coscos() or Chili Works() could handle this side of the project, as the switch from Unity meant that this would be section would be harder to develop. Unfortunately none of the game development engines would allow for easy integration into the existing project and in some cases would need to have them be directly installed into Visual Studio. Thus the difficult choice was made and game development was to be handled using SFML, while this would cause elements such as collision detection and 2D graphics to take more time to set up, it did allow for the project to continue without dependency on any other outside sources.

Development then continued smoothly for the next couple weeks, as the various components that would be used as the building blocks for the rest of the project were implemented one after the other, these included the Screen Manager, Input Manager and File Manager class. This continued for about two week after which it was time to begin work on the Gameplay side of the project. Utilising the various components that were implemented before it, development of the class ShooterScreen managed to come together fairly quickly, couple this with SFML’s ability to easily implement the various features that made up the game. This included collision detection, 2D Graphics and user input. After 2 weeks the game was almost finished, the next step was to combine it with the Emotiv class.

The EmotivShooterScreen class was originally a copy of Shooter screen, then code that took data from the Emotiv class was implemented into it. The data taken from this was then added/subtracted from the variables that were utilised by the Player, Enemy and Boss classes, in a way that should adapt the gameplay to be more appealing if the players are bored or frustrated. Unfortunately this led to a new problem arising, retrieving the data from the headset would cause the game to suffer a horrible framerate drop that made the game unplayable. To help fix this an Asynchronous thread was created in the main class that would run in the background of the application, luckily SFML contained a feature that allowed this to be easily implemented (sf::Thread), and the Emotiv retrieval functions were moved to here. The next step was to allow the EmotivShooterScreen class to have easy access to these variables, this lead to the development of the Global Variables class.

The GlobalVariables class was used to hold and send information to the various classes of the application, this was originally for the Emotiv data but was later expanded to hold the players score, number of enemies killed, times hit and boss’s health. These variables were included with the Emotiv data and later exported to a text file through an output function located in the Emotiv thread.

After this development then switched to designing of the Test documents and putting final touches to the application. The test documents needed to be designed to get the candidates feelings about the different versions of the game and about the application as a whole, after much thought and feedback from the module supervisor it was decided to accomplish this two different questionnaires needed to be created. The first was to be filled out after each version of the game, it enquired about how the game they just played affected them emotionally and how long they felt it took to complete. The second was to be filled out after both of the previous questionnaires were complete, this one enquired about the subject’s personal details (e.g. age, gender) and which version of the application they preferred and why. These questionnaires were designed so that the subjects would be able to give their opinion on how they felt the applications affected them so that they could be directly compared to the emotional data gained from the headset as well as get their thoughts on the differences between the two versions of the game (Copy of Questionnaires located in Appendix 2). The last thing to do was implementing textures into the game, adjusting the various variables, removing the debug info from being displayed and ironing out any final bugs.

* 1. **– Implementation**
     1. **Emotiv**

The Emotiv Class was the first to be developed, based on a similar class provided in the Emotive EPOC SDK, the class is used as a mediator between the application and the Emotiv SDK. The class takes the raw data from the headset and converts it into data that can be used easily by the application.

One such example of this is the ParseAffectiv function, this method takes the raw emotional data from the headset and converts it into different easy to read variables (e.g. frustrationScore).

**An example of the code is located in in appendix 1.**

* + 1. **Screen Manager**

Due the application containing multiple screens, the Screen Manager class was implemented, this allows for easy transition, loading, unloading and updating of different screens through a simple function call.

**An example of the code is located in the in appendix 1.**

* + 1. **Input Manager**

The input Manager class was implemented to allow easy integration of keyboard controls that would be utilised throughout the application.

**An example of the code is located in the in appendix 1.**

* + 1. **Shooter Screen**

The Shooter Screen (a child of Screen manager) is where the gameplay takes place. This class was developed after all the previously mentioned classes, as it utilises both the screen manager and Input manager class to great extent.

In this class various subclasses are brought together each utilising SFML’s sf::RectangleShape class, which allows for easy creation of 2D shapes which make up the building blocks of the game.

All of the following classes also take advantage of SFML’s FloatRect function which takes an entities shape and position within the application and creates a bounding box. Thanks to this and an intersection detection function, Collision detection for the Game side of this project was made much easier.

**An example of the code is located in appendix 1.**

* + - 1. **Player**

This class is used to create a user controlled entity that’s used to play the game. The class is set up to be easily manipulated by Keyboard controls located in Shooter screen that utilises SFML’s sf::Event class. The class also features an update function that stops the player from being able to leave a specific area of the screen or move too quickly.

**An example of the code is located in in appendix 1.**

* + - 1. **Enemy**

This class is designed to be called as an array. It’s designed so that when called can spawn several entities that spawn at random locations at one side of the screens. These entities will then fly around the screen while heading towards the opposite side of the screen. There is also functions within ShooterScreen that will clone the enemies under certain conditions so that they always replenish themselves and after a while they will go from 1 enemy to approximately 30.

**An example of the code is located in appendix 1.**

* + - 1. **Boss**

This class is designed to spawn a large entity that takes up an entire side of the screen and due to conditions in ShooterScreen, only spawns after a predetermined amount of time. After spawning the entity will then begin moving towards the player, once it reaches a predetermined position it will begin firing on the player. It also features an integer called health which ends the game if it reaches 0.

**An example of the code is located in appendix 1.**

* + - 1. **Bullet and Enemy Fire**

These two classes are fairly similar as their designed to spawn entities that appear to spawn from another entities by taking in their positions. Once spawned the entities begin to fly towards a specific side of the screen.

**An example of the code is located in appendix 1.**

* + 1. **Emotiv Shooter Screen**

The EmotivShooterScreen class is essentially a clone of Shooter screen but it features some extra code that utilises the Emotiv class and its ParseAffectiv function.

**An example of the code is located in appendix 1.**

* + - 1. **Emotiv Update Thread**

To help gain access to the Emotiv Data without it having a negative effect on the applications performance an asynchronous thread was implemented into the application in the main class. This thread runs alongside the rest of the application and gets the data from an instance of the Emotiv class, it then passes these variables to another class called global variables. The thread also features code that allows the thread to output the data to a text file which will be used to evaluate the data and draw a conclusion. It also features a Mutex provided by SFML’s sf::Mutix class to help avoid data corruption.

**An example of the code is located in in appendix 1.**

* + - 1. **Global Variables Class**

The GlobalVariables class was created to easily pass variables between the different classes of the application. This is mainly used by the Emotiv Update thread as it uses it passes in the values gained from the headset to the class so that it can be used by EmotivShooterScreen.

**An example of the code is located in appendix 1.**

* + - 1. **Get Emotiv Method**

The GetEmotiv method is located within EmotivShooterScreen class. When called the class calls the GlobalVariables classes “Getter” functions so that they return the current emotional data taken from the headset, it then loads these into 4 local variables to be used to affect the gameplay.

**An example of the code is located in appendix 1.**

* + 1. **Integration of data**

The local variables that contain the users emotional data is then added to or subtracted from pre-existing variables that affect various entities of the game. One example of this would be the variables fireRate and frustrationScore. FireRate is used to control how often the entity bullet spawns when the player holds down the fire button, and frustrationScore is a value from 0 – 1 representing how frustrated the user is. FrustrationScore is then subtracted from fireRate, this means that whenever the player is frustrated, either from a high number of enemies on the screen or the boss not dying fast enough the player will gain a small but noticeable increase to their fire speed.

Examples like this and the others included in the final product will hopefully help deliver the player a unique and tailored experience that will be different each time they play.

**5.3 – Gameplay**

The final application was a simple side scrolling space shooter similar to games like R-type (Irem, 1987) or Xavious (Namco, 1982), these games featured simple mechanics that would not only be easy to develop but would also allow for several different possible area the users emotional data could be used effectively.

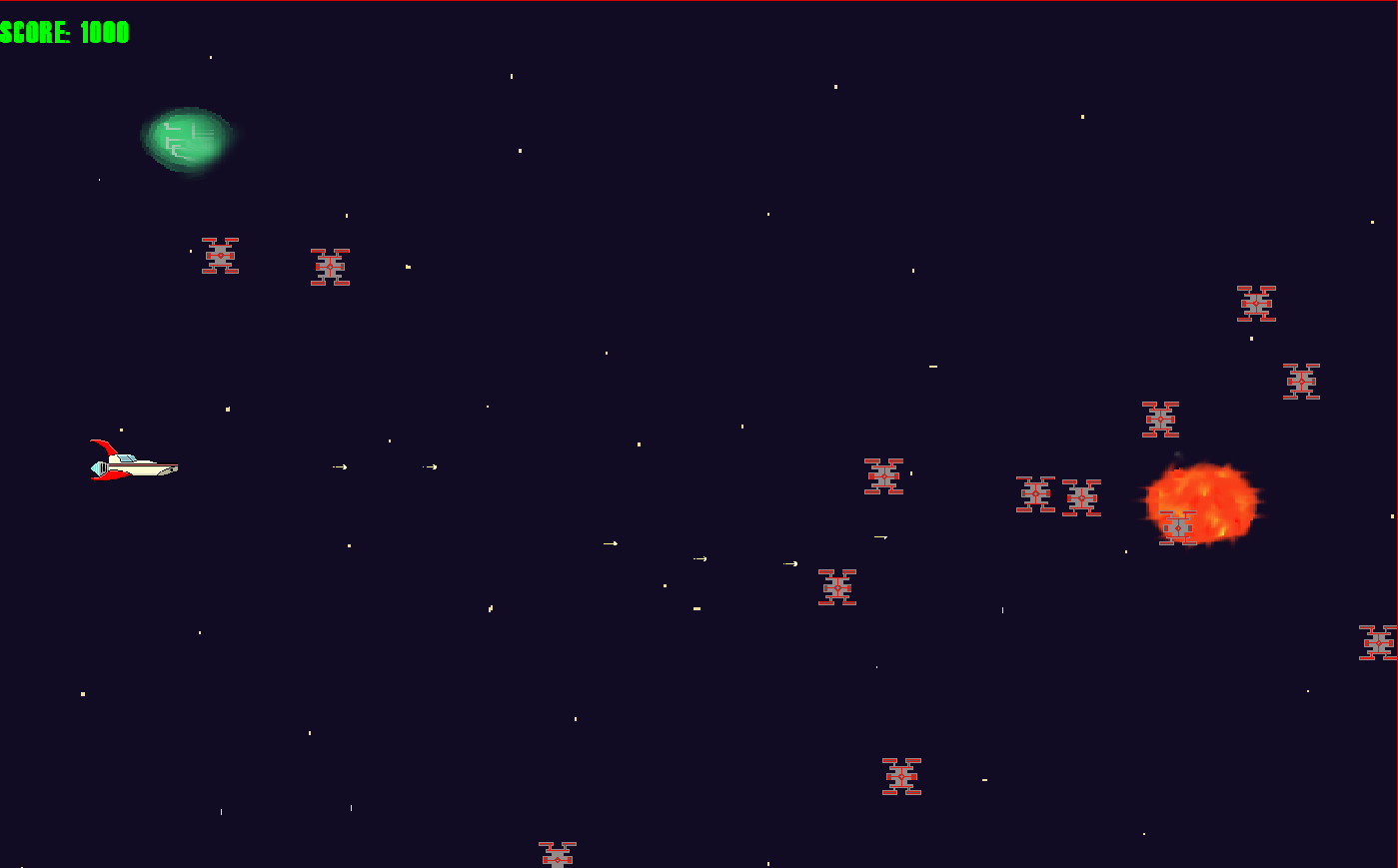
The gameplay was designed to be simple enough that almost anybody would be able to pick up and play, yet complex enough have areas where the user’s data could be used to change it. The users take control of a space ship on the left hand of the screen which can fly around within a specific pre-set barrier that the ship is unable to leave. As time passes the enemy ships will begin spawning on the right hand of the screen and begin heading towards the left hand of the screen, the player will then have to either avoid or shoot the enemy ships to stay alive. After a pre-determined time the boss will spawn and fire at the player, it will remain there until the player reduces its health to 0, thus ending the game.

The user’s feedback affects the game in the following way:

* When the player is excited boss’s attacks will spawn less.
* When frustrated the Player’s spaceship can fire faster
* And when bored the enemy ships will move faster

After looking into the different ways the emotive data could be integrated (as referenced in section 4.5) these choices were made as it was felt they would allow the game to scale in a way that would make the game easier/harder when required, without going too far and ruining the experience. After looking into games of this genre it was discovered that several of them had tense/exciting boss fights where the player would have to dodge waves and waves of enemy fire, because of this it made sense to have the users excitement affect the bosses fire rate as it would allow each user to have their own unique level of difficulty in this section that would cater towards making the fight as exciting as possible without making it impossibly hard. The next choice was having the fire rate tied directly to the users frustration it would allow frustrated players to more easily defeat the enemy ships of defeat the boss faster, and for those user that aren’t frustrated the game will last a little longer letting them enjoy the game more. And finally having the users boredom affect the enemy ship speed, this was one of the easier choices to make as having faster enemies would add to the general excitement of the game without making it too easy or difficult.

The game doesn’t feature any lives so player can’t actually die, instead the game features a high score mechanic where the player scores points from destroying enemies and avoiding colliding with anything. This way the game won’t have to be replayed each time the player dies which could prove very bad for retrieving test data in a short amount of time. Though the prospect of gaining a high score and an incentive for earning the highest score at the end of the testing period should be enough to gain an emotional commitment from the players (e.g. arcade games).



*Figure 1: Space Shooter Screenshot*

**5.4– Testing**

The Testing for the project consisted of candidates being taken to a room where they were explained what the test would involve and what they would have to do, they were told that their information will not be made available, that they would remain anonymous and they could leave at any time. If everything was satisfactory for them they signed the consent form and the test could begin.

The set up for the Emotive EPOC headset involved the candidates being shown how it worked and the proper way to wear/remove it. If further adjustment was needed the candidates permission was asked of before doing anything.

After the headset was working correctly the program was run and they first game was selected and the timer was started. After they completed the game, their emotional and game data was taken from an output file and stored onto a corresponding text file for later analysis. The candidates were then handed a questionnaire to fill in asking about how the game affected them emotionally and how long they though it took to complete, the process was then repeated with the second game. After both games were completed and both questionnaires were filled in the candidates were given one last form which enquired about their details (e.g. age, gender) and which of the games they preferred.

During the tests the users emotional data, time and score was outputted to a text file for later analysis, the data would look something like this:

Current time: 2015-04-03.17:49:15

Current Score: 2000

Number of Enemies destroyed: 24

Number of times hit: 10

Boss has spawned

Boss health: 185

User Excitement: 0.2

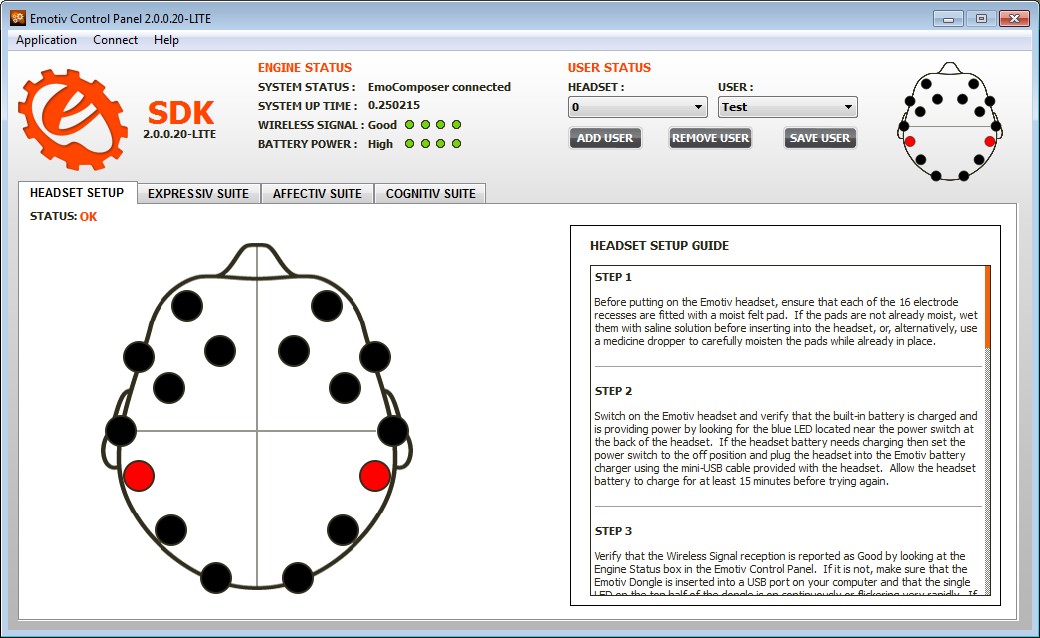
User Boredom: 0.3

User Frustration: 0.4

User Meditation: 0.5

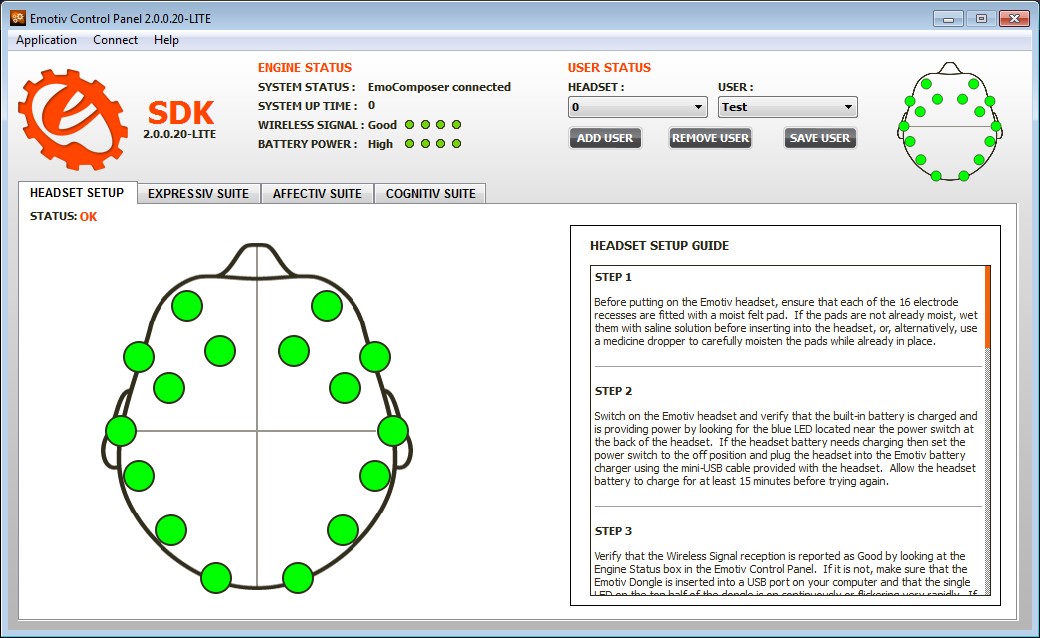
User Emotional State: Engaged

The Emotiv Control panel (as shown in Figure 1), shows all of the different nodes on the headset and how strong their signal are (e.g. Red = bad, green = good).



*Figure 2: Emotiv Control Panel – No connectivity*

If the headset was correctly adjusted the Control panel would look like Figure 2, with eh majority of the nodes being green.



*Figure 3: Emotiv Control Panel – Perfect connectivity*

Depending on their order each candidate each candidate played the two games in a different order (e.g. odd numbered candidates played the Emotiv game second). This was to rule out that the order in which these games were played would have any effect on the player’s feelings towards the different games. The candidates were also selected in a way that would keep the male/female ratio as close as possible, in case having a majority of one gender would affect the results.

As the Application uses the user’s frustration, Boredom and excitement it was decided that these should be exported to the text file to be used to get the average emotional scores for each of the applications. This way we will be able to see which of the two applications the users find more exciting, frustrating ect. This data can also be compared to rating the users can give the different applications in each of the questionnaires, to see if there’s a difference between how the users thought they felt and how they actually felt.

**5.5 – Problems and Issues**

Various problems occurred throughout the development and testing of the Application.

During the applications development the main problems that occurred were with how the application was going to be created, the first problem occurred when the Emotiv code was being integrated into Untiy and C#. After several failed attempts to integrate the Emotiv code into standard Unity it was decided that the development platform would be swapped to Microsoft Visual Studios 2010 and the coding language was switched to C++, while this had a negative effect on development time and would cause development of the game to be simpler and take longer, it did allow the Emotiv libraries to be easier integrated which did help in the long run.

The next problem to occur was when selecting a game engine to allow development of the game to be completed quicker. Several different engines were considered but in the end it proved fruitless as none of them would easily integrate into the existing application and in some cases would require the entire project to be recreated in a different format. Because of this setback it was decided that SFML would be used to develop the game, which would take longer to do, but allowed for more freedom with the development and it was already integrated into the project.

The testing sessions were impeded by several different problems, the first was that several of the nodes for the headset were damaged so they refused to stay on unless held in by the subjects head, this lead to a much longer and more frustrating set-up for each of the candidates, but despite this the headsets were adjusted until their signal was as clear as possible in each test.

Another problem with the hardware was that most of the nodes had become corroded by rust and would occasionally not give good results or not be registered by the headset, this was thankfully fixed by giving the nodes a thorough cleaning, but there was still some corruption present of several of them, so there’s no guarantee the data is a 100% accurate among all tests. To help reduce this from affecting the results all the data was closely analysed to check for any erratic changes in the individual user’s emotional scores and any junk data (e.g. -6.97301e+009) was skipped.

The last issue that would occur with the headset had problems working for most of the female test subjects, possibly due to their long hair. The headset would not work unless it was thoroughly adjusted for upwards of an hour, which proved difficult and frustrating due to the subject’s long hair or the nodes continuously falling off.

1. **Results**

The goal of this project is to determine whether or not the use of an EEG headset as a way to directly and indirectly affect gameplay will have a positive effect on the player’s experience (e.g. did they enjoy it more). There are examples of similar investigations into this subject as stated above; this project is to see if having it directly tie into the games ability to catch/hold the user’s attention will have a positive effect compared to a standard level of the game that doesn’t change. If this is implemented correctly it should provide each player with a tailored experience that’s different each time.

* 1. **Data Obtained from Testing**

For the testing stage, a pool of 10 candidates were chosen and a 50/50 ratio between male and female candidates was maintained, these are the results gained from those tests.

*Figure 4: User Game Experience Chart*

The candidates were asked how many videogames they had played within the last month, as you can see in Figure 4 all the candidates have played games within the last month, though the quantity of games they have played recently does vary quite a bit.

**Preferred Game: Emotiv: 7 / Non-Emotiv: 3**

*Figure 5: Preferred Game Chart*

During the Questionnaire the candidates were asked which version of the game they preferred, as you can see by Figure 5 the majority of Candidates preferred the Emotiv Version.

**Preferred game order: First game: 4 / Second Game: 6**

*Figure 6: Preferred Game Order Chart*

To rule out that the order in which the two games are played would have an effect on the user’s opinion of which game they preferred, the order in which the different games were played was switched from candidate to candidate. As we can see in the results the majority of the players preferred the second game but there were still a good portion of candidates who liked the first game.

**Game which ended first: Emotiv: 6 / Non-Emotiv: 4**

*Figure 7: Which Ended first Chart*

Due to the Users fire rate being affected by the user’s frustration level, it was possible for the user to fire faster or slower than in the Non-Emotiv version depending on the user’s emotions. As can be seen in Figure 7 the Emotiv game the Emotiv game was completed first the majority of the time, this means that the candidates were feeling frustrated the majority of the time, this could be due to the game being frustrating, or the headset being uncomfortable.

**Guessed time: Non-Emotiv 2:57 / Emotiv – 2:11**

**Actual time: Non-Emotiv 2:49 / Emotiv – 2:43**

*Figure 8: Average Times Chart*

The Candidates were asked how long they thought each game took to complete, and the actual time was noted down as well. As you can see in Figure 8, the average Non-Emotiv guessed time is pretty close to the average actual time. The average Emotiv guessed time is actually 32 seconds short of the average actual time for the Emotiv game, this may be due to the users experiencing a sense of flow, due to enjoyment, and there for the game seemed shorter than it actually was.

**Higher Score Ratio: Emotiv: 8 / Non-Emotiv: 2**

*Figure 9: Higher Score Ratio Chart*

As can be seen in Figure 9, users achieved a higher score in the Emotiv version of the Game than the Non-Emotiv version, this is most likely due to the Emotiv version of the game feeling easier to the users.

**Users average emotional rating: Non-Emotiv - 4.6 Emotiv - 3.8**

**Users average Enjoyment rating: Non-Emotiv - 4.9 Emotiv - 5.6**

**Users average Boredom rating: Non-Emotiv - 3.9 Emotiv - 4.6**

**Users average Frustrating rating: Non-Emotiv - 4.6 Emotiv - 3.8**

*Figure 10: Average Emotional Rating Chart*

During the questionnaire the Candidates were asked to give a rating from 1 – 10, how much each game affected them emotionally as a whole and for specific emotions, Figure 10 shows the average for these results, As you can see the candidates felt the Emotiv version affected them less that the non Emotiv as a whole, yet they game it a higher rating for enjoyment and boredom.

**Average Excitement: Non-Emotive - 0.43494711895382 Emotiv - 0.43739484082705**

**Average Boredom: Non-Emotiv - 0.69037146763228 Emotiv - 0.68917769485033**

**Average frustration: Non-Emotiv 0.52120251468908 Emotiv - 0.53607047390687**

*Figure 11: Average Emotional Scores Chart*

The headset associates each user’s emotion a score between 1 and 0, in Figure 11 we can see the average of these scores for each emotion for all 10 candidates. The scores are fairly similar, but there is a small difference between the two versions, the Emotive version of the game is on average slightly more exciting and less boring though it is a little more frustrating than the Non-Emotiv Version.

**Common emotions at specific times?**

1. **Discussion**

In this section we will discuss the results from section 6, and try to notice any correlations between the different results and try to form the basis of a conclusion for Section 8.

From Figure 5 we can see that the Emotiv version of the game was preferred among the candidates, this could be due to the difficulty adapting to their emotional state which provided an easier game in which the majority of the candidates were able to gain a higher score (Figure 9). Or it could be due to the fact that several of the candidates were able to finish it quicker (Figure 7) due to the change in fire-rate. Either way the Emotiv version was received better and did provide each candidate with a different experience.

We can also see from Figure 10 that on average the candidates felt that the Emotiv version affected them less emotionally as a whole, though they also said that they found it more enjoyable and boring than the Non-Emotiv version., and when you compare that to the results from Figure 11 we can see that the candidates on average actually did find the Emotiv version slightly more exciting. Though in contrast to figure 10’s results they did find the Emotiv version more frustrating, this is most likely why the majority of the Emotiv games ended quicker as the frustration score affects the player’s fire speed, though this could have been affected by the long set up times for the headset.

Due to the enemies speed, and bosses fire rate being affected by the player’s feedback, most users found the Emotiv version of the game easier (as reflected by users being able to get a higher score in this version, Figure 9), than the non-emotive version this helped to lead to a split in opinion over which version of the game the players preferred. Some of the candidates who preferred the non-Emotiv version said they found it more of a challenge, which they preferred. Some test subjects said that they didn’t actually notice a difference, and that they rated their preference based on how well they did during the game, in these cases we can assume that the user’s emotional state affected the variables in such a way that they closely resembled those of the non-Emotiv version. The subjects who preferred the emotive version said that liked that the difficulty adapted to them and as such were able to gain a higher score and defeat the boss quicker, one candidate even said that the Emotiv version helped them feel more “Involved” in the game as they were affecting the game without even realising it.

1. **Conclusion and Future Development**

In this Section we will be drawing a conclusion for the project as a whole and where this research could lead us if continued.

* 1. **Conclusion**

From the original Project aim:

**“Does using Bio-feedback to affect gameplay, have a positive effect on the player’s experience?”**

The purpose of this project was to create an application with fun yet simple gameplay and see if changing it based on the users emotional data would be a good thing. To help prove this an application that could take in the user’s emotional data from an EEG device and record it in real time. This was then added to a simple side scrolling space shooter game, where the various variables would be affected by the user’s feedback in a way that would affect the difficulty so that it would provide each player with a unique experience.

From the feedback and results gained from testing I can safely say that this experiment was a success. The Emotiv version of the game was much better received and thanks to the adapting difficulty players found it much easier and were able to have more fun (as shown in Figure 11’s average Emoitv excitement score), and perform better (as shown in Figure 9’s Higher score ratio), than in a game with set variables that didn’t change. There was a few cases where users preferred the Non-Emotiv version but I believe this could be rectified by increasing the general difficulty of the Emotiv Version of the game and having the user’s emotional data have a greater effect on the games variables.

Another interesting result gained from testing is that users on average felt that the Emotiv version was 30 seconds shorter than it actually was (Figure 8) this suggests that users may have felt a greater sense of flow when playing the Emotiv version which supports the research in section 4.7.

So in conclusion it’s felt that having a game change based on the player’s emotional data is a good thing and warrants further investigation, as it could provide almost limitless changes to the various genres of games and provide players all around the world with a unique and tailored experience specifically for them.

* 1. **Future Development**

If development of this project were to continue there would be a few areas that would warrant further investigation.

1. **Would a more difficult version of the Emotiv game that can be affected more by the user’s emotions appeal to all users?**

The candidates who didn’t like the Emotiv version complained that the game was too easy, and they preferred a challenge. So if the Emotiv version of the application was to be made more difficult (e.g. by increasing the average enemy speed and boss fire rate) and the user’s data was to have a greater effect on the gameplay would this bridge the gap and create a game that would appeal to all of the candidates?

1. **Could the other types of Emotiv data integrate well into the application?**

The Application only uses a small part of the headset’s capabilities, specifically the Affectiv’s Boredom, Excitement and Frustration scores. But the headset is also capable of reading the users facial expressions and specific thoughts (e.g. using thought to fire a special weapon), so would integrating these into the application make it a more interesting game, and open up much more possibilities for design?

1. **Would Emotiv Integration work well for other game genres?**

This project proved that Emotiv integration works for the side scrolling shooter genre of games, but there’s so many other genre’s that this could work for. There’s other classic genre’s such as platformers and action games that this could be easily integrated into, but what about the more complicated genres?

* A horror game could use the Emotiv’s ability to detect if the users feeling calm to create an even more horrifying experience.
* Fighting games could use the user’s anger or thoughts to create devastating attacks.
* Narrative/story driven games could use the user’s facial expressions to unlock special dialogue options.

1. **References**

**Games**

Steve Russel. 1961. *Spacewar!*. [online]. PDP-1. Availiable from http://www.masswerk.at/spacewar/ [Accessed 3 December 2014]

Nintendo. 1985. *Super Mario Bros*. [Cartridge]. Famicom, Nintendo Entertainment System. Gameboy, Wii, 3DS, WiiU. Nintendo. Available from <http://www.mariowiki.com/Super_Mario_Bros>. [Accessed 3 December 2014]

The Fullbright Company. 2013. *Gone Home*. [disk, download]. PC, Mac, Linux. The Fullbright Company. Available from <http://fullbright.company/gonehome/> [Accessed 3 December 2014]

The Chinese Room. 2012. *Dear Esther*. [disk, download]. PC, MAC, Linux. The Chinese Room. Available from <http://dear-esther.com/> [Accessed 3 December 2014]

Freebird Games. 2014. *To The Moon*. [disk, download]. PC. Freebird Games. Available from <http://freebirdgames.com/to_the_moon/> [Accessed 3 December 2014]

Nintendo. 2006. *Wii*. [online]. Available from <https://www.nintendo.co.uk/Wii/Wii-94559.html> [Accessed 3 December 2014]

Trion Worlds. 2011. *RIFT.* [disk]. PC. Trion Worlds. Available from <http://www.riftgame.com/en/> [Accessed 16 April 2015]

Valve, Turtle Rock Studios. 2008. *Left 4 Dead.* [disk, download]. PC, Xbox 360, Mac. Valve. Available from <http://www.l4d.com/blog/> [Accessed 16 April 2016]

Ubisoft. 2012. *Rocksmith*. [disk/download]. PC, PS3, Xbox 360. Ubisoft. Available from: <http://rocksmith.ubi.com/rocksmith/en-GB/home/index.aspx> [Accessed 24 April 2015]

Irem. 1987. *R-Type.* [Arcade]. Arcade, PC, TurboGrafx 16, Xbox Live Arcade, Nintendo Virtual Console, Android. Available from <http://rtype.wikia.com/wiki/R-Type> [Accessed 16 April 2015]

Namco. 1982. *Xevious*. [Arcade] Arcade, NES, PC, Commodore 64, PC, Atari 7800, Nintendo 3DS. Atari. Available from <http://www.arcade-museum.com/game_detail.php?game_id=10505> [Accessed 16 April 2015]

**Papers/ Books/ Articles**

David Millett. 2001. Hans Berger: From Psychic Energy to the EEG. Available From: <http://methodsinbraincomputerinterfaces.wikispaces.com/file/view/BergerBiography.pdf> [Accessed 23 April 2015]

Harold Shipton. 1975. EEG Analysis: A History And A Prospectus. Available From: <http://www.annualreviews.org/doi/pdf/10.1146/annurev.bb.04.060175.000245> [Accessed 23 April 2015]

Badcock et al. 2013. Validation of the Emotiv EPOC EEG Gaming System for Measuring Research Quality Auditory ERPs. Available from <https://peerj.com/articles/38/> [Accessed 3 December 2014]

Harold Goldberg. 2012. The 25 Greatest Breakthroughs in Videogame History. Available From: <http://uk.ign.com/articles/2012/01/30/the-25-greatest-breakthroughs-in-video-game-history> [Accessed 23 April 2015]

Frome. 2007. Eight Ways Videogames Generate Emotion. Available from: <http://www.digra.org/wp-content/uploads/digital-library/07311.25139.pdf> [Accessed 3 December 2014]

Nacke et al. 2011. Biofeedback Game Design: sing Direct and Indirect Physiological Control to Enhance Games Interaction. Available from: <http://www.hci.usask.ca/uploads/202-Biofeedback-Game-Design_CHI2011.pdf> [Accessed 3 December 2014]

Cowley et al. 2008. Towards an Understanding of Flow in Video Games. Available From: <http://dl.acm.org/citation.cfm?id=1371223> [Accessed 3 December 2014]

Booth, 2009. The AI Systems of Left 4 Dead. Available from: <http://www.valvesoftware.com/publications/2009/ai_systems_of_l4d_mike_booth.pdf> [Accessed 27 March 2015]

Mihaly Csikszentmihalyi. 2014. Flow and the Foundations of Positive Psychology. Available From: <http://www.springer.com/gp/book/9789401790871> [Accessed 23 April 2015]

Stuart Dredge. 2015. Are freemium games focusing too much on monetisation and not enough on fun?. Available from: <http://www.theguardian.com/technology/2015/feb/17/freemium-mobile-games-monetisation-candy-crush> [Accessed 23 April 2015]

**YouTube Videos**

Ubisoft. 2011. *Rocksmith Dynamic Difficulty*. [online]. YoutTube. Available from: <https://www.youtube.com/watch?v=R0yhx5aDjws> [Accessed 24 April 2015]

**Software**

Microsoft. 2010. *Visual Studios 2010*. Available from: <https://www.visualstudio.com/> [Accessed 16 April 2015]

Laurent Gomila, 2014. *SFML Website*. Available from: <http://www.sfml-dev.org/>[Accessed 16 April 2015]

Apple Inc. 2001. *ITunes*. Available from: <https://www.apple.com/uk/itunes/> [Accessed 23 April 2015]

Google. 2008. *Google play*. Available from: <https://play.google.com/store?hl=en_GB> [Accessed 23 April 2015]

Tag Games. 2014. *Chili works*. Available from: <http://www.chilli-works.com/> [Accessed 16 April 2015]

Cocos. 2010. *Cocos2d-x.* Available from: <http://www.cocos2d-x.org/> [Accessed 16 April 2015]

1. **Appendices**
   1. **Appendix 1: Code Snippets**

Emotiv Class:

void Emotiv::parseAffectiv(EmoStateHandle eState)

{

ES\_AffectivGetEngagementBoredomModelParams(eState, &aff.rawScore, &aff.minScale, &aff.maxScale);

cout << "Engagement/Bored raw score: " << &aff.rawScore << ", min scale: " << aff.minScale << ", max scale: " << aff.maxScale;

if (aff.minScale == aff.maxScale)

{

cout << ", scaled score: undefined" << endl;

}

else

{

aff.engagementBoredomScore = aff.calculateScale(aff.rawScore, aff.minScale, aff.maxScale);

cout << ", scaled score: " << aff.engagementBoredomScore << endl;

}

ES\_AffectivGetFrustrationModelParams(eState, &aff.rawScore, &aff.minScale, &aff.maxScale);

cout << "Frustration raw score: " << &aff.rawScore << ", min Scale: " << aff.minScale << ", max scale: " << aff.maxScale;

if (aff.minScale == aff.maxScale)

{

cout << ", scaled score: undefined" << endl;

}

else

{

aff.frustrationScore = aff.calculateScale(aff.rawScore, aff.minScale, aff.maxScale);

cout << ", scaled score: " << aff.frustrationScore << endl;

}

Screen Manager:

void ScreenManager::Transition(sf::RenderWindow &Window, sf::Clock clock)

{

if(transition)

{

fade.Update(Window, clock);

if(fade.GetAlpha() <= 1.0f)

{

currentScreen->UnloadContent();

delete currentScreen;

currentScreen = newScreen;

currentScreen->LoadContent();

newScreen = NULL;

transition = false;

}

else if(fade.GetAlpha() >= 0.0f)

{

transition = false;

fade.SetValue(fade.GetActive(), false);

fade.SetActive(false);

}

}

}

Shooter Screen:

for(int b =0; b < bullet.size(); b++)

{

sf::FloatRect bulletBB = bullet[b].getGlobalBounds();

if(boss.GetActive() == false)

{

if(bulletBB.intersects(enemyBB)) // Bullet/Enemy colision

{

enemy[e].SetHit(true);

score += 100;

bullet[b] = bullet[bullet.size() - 1];

bullet.pop\_back();

}

}

if(bulletBB.intersects(bossBB)) // Bullet/Boss colision

{

boss.SetHit(true);

bullet[b] = bullet[bullet.size() - 1];

bullet.pop\_back();

}

}

Player:

void Player::Update()

{

if(GetVelocity().x > 0)// slows player down when not moving

{

ChangeVelocity(sf::Vector2f(-1.0f, 0.0f));

}

if(GetVelocity().x < 0)// slows player down when not moving

{

ChangeVelocity(sf::Vector2f(1.0f, 0.0f));

}

if(GetVelocity().y > 0)// slows player down when not moving

{

ChangeVelocity(sf::Vector2f(0.0f, -1.0f));

}

if(GetVelocity().y < 0)// slows player down when not moving

{

ChangeVelocity(sf::Vector2f(0.0f, 1.0f));

}

if(getPosition().x > 500) // player boundries

{

ChangeVelocity(sf::Vector2f(-playerSpeed - 9, 0.0f));

}

if(getPosition().x < 0)

{

ChangeVelocity(sf::Vector2f(playerSpeed + 9, 0.0f));

}

if(getPosition().y > 980)

{

ChangeVelocity(sf::Vector2f(0.0f, -playerSpeed - 9));

}

if(getPosition().y < 0)

{

ChangeVelocity(sf::Vector2f(0.0f, playerSpeed + 9));

}

}

Enemy Class:

if(enemy[e].GetPosition().x < -50)

{

enemy[e] = enemy[enemy.size() - 1];

enemy.pop\_back();

if(enemy.size() < 30)

{

enemy.push\_back(Enemy());

enemy.push\_back(Enemy());

enemy[enemy.size() - 1].SetPosition(sf::Vector2f(1100 + rand()%301, rand() % 1000));

}

else enemy.push\_back(Enemy());

enemy[enemy.size() - 1].SetPosition(sf::Vector2f(1100 + rand()%301, rand() % 1000));

}

else

{

enemy[e].ChangeVelocity(sf::Vector2f(-enemySpeed, 0.0f));

}

Boss:

void Boss::Update()

{

if(position.x < 800)

{

SetVelocity(sf::Vector2f(0.0f, 0.0f));

active = true;

}

if(active == true)

{

invincible = false;

attack = true;

}

position.x += velocity.x;

position.y += velocity.y;

setPosition(position);

}

Bullet and Enemy:

EnemyFire::EnemyFire(Boss\* bosspointer)

{

boss = bosspointer;

dimensions = sf::Vector2f(15, 15);

velocity = sf::Vector2f(-10, -15 + rand()% 25);

drawn = false;

setSize(dimensions);

setFillColor(sf::Color::Blue);

if(texture.loadFromFile("Sprites/test.png"))

{

}

setTexture(&texture);

setTextureRect(sf::IntRect(0,0, 50, 50));

}

Emotiv Update:

void EmotivUpdate()

{

GlobalVariables\* globalPointer = new GlobalVariables();

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////

// Emotiv //

emotivMutex.lock();

//cout << "emotiv thread running" << endl;

emo->checkForNewEvent();

emo->parseAffectiv(emo->getEstate());

globalPointer->SetGlobalExcitement(emo->getAffectiv().shortTermExcitementScore);

globalPointer->SetGlobalBoredom(emo->getAffectiv().engagementBoredomScore);

globalPointer->SetGlobalFrustration(emo->getAffectiv().frustrationScore);

globalPointer->SetGlobalMeditation(emo->getAffectiv().meditationScore);

globalPointer->SetDetectedEmotion(emo->getAffectiv().emotion);

//globalPointer->SetGlobalExcitement(0.5);

//globalPointer->SetGlobalBoredom(0.5);

//globalPointer->SetGlobalFrustration(0.5);

//globalPointer->SetGlobalMeditation(0.5);

//emotionDetected = emo->getAffectiv().emotion;

emotivMutex.unlock();

//cout << "current time " << globalPointer->currentDateTime() << endl;

if(globalPointer->GetGameStart() == true)

{

if(updateRate >= 20)

{

//cout << "update text" << endl;

/\* Specify file to output to \*/

outputFile.open("../Emotions.txt", ios:: in | ios::out | ios::ate);

outputFile << "Current time: " << globalPointer->currentDateTime() << endl;

outputFile << "Current Score: " << globalPointer->GetScore() << endl;

outputFile << "Number of Enemies destroyed: " << globalPointer->GetEnemiesKilled() << endl;

outputFile << "Number of times hit: " << globalPointer->GetTimesHit() << endl;

if(globalPointer->GetBossSpawned() == true)

{

outputFile << "Boss has spawned" << endl

<< "Boss health: " << globalPointer->GetBossHealth() << endl;

}

outputFile << "User Excitement: " << emo->getAffectiv().shortTermExcitementScore << endl

<< "User Boredom: " << emo->getAffectiv().engagementBoredomScore << endl

<< "User Frustration: " << emo->getAffectiv().frustrationScore << endl

<< "User Meditation: " << emo->getAffectiv().meditationScore << endl

<< "User Emotional State: " << emo->getAffectiv().emotion << endl << endl;

outputFile.close();

updateRate = 0;

}

else

updateRate += 1;

}

}

Global Variables Class

class GlobalVariables

{

public:

GlobalVariables();

~GlobalVariables();

double GetGlobalExcitement();

double GetGlobalBoredom();

double GetGlobalFrustration();

double GetGlobalMeditation();

string GetDetectedEmotion();

bool GetGameStart();

bool GetBossSpawned();

int GetEnemiesKilled();

int GetTimesHit();

int GetScore();

int GetBossHealth();

void SetGlobalExcitement(double i);

void SetGlobalBoredom(double i);

void SetGlobalFrustration(double i);

void SetGlobalMeditation(double i);

void SetDetectedEmotion(string i);

void SetGameStart(bool i);

void SetBossSpawned(bool i);

void SetEnemiesKilled(int i);

void SetTimesHit(int i);

void SetScore(int i);

void SetBossHealth(int i);

string currentDateTime();

private:

sf::Mutex globalMutex;

};

Get Emotiv Function:

void EmotivShooterScreen::GetEmotiv()

{

shooterMutex.lock();

frustrationScore = globalVariables.GetGlobalFrustration();

meditationScore = globalVariables.GetGlobalMeditation();

boredomScore = globalVariables.GetGlobalBoredom();

excitementScore = globalVariables.GetGlobalExcitement();

/\*cout << globalVariables.GetGlobalFrustration() << endl

<< globalVariables.GetGlobalExcitement() << endl

<< globalVariables.GetGlobalMeditation() << endl

<< globalVariables.GetGlobalBoredom() << endl;\*/

//cout << frustrationScore << endl

// << excitementScore << endl

// << meditationScore << endl

// << boredomScore << endl;

shooterMutex.unlock();

}

* 1. **Appendix 2: Questionnaires for Testing**

**Test documentation**

**Candidate No:**

**Quiz 1 – after each game**

**Question 1.**

Would you say the game affected you emotionally?

Not at all Definitely

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

**Question 2.**

How would you rate your enjoyment playing the game?

No fun at all Very Fun

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

**Question 3.**

How would you rate your Frustration playing the game?

None at all Very Frustrating

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

**Question 4.**

How would you rate your Boredom playing the game?

None at all Very Boring

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

**Question 5.**

How calm were you when playing the game?

Not calm at all Very Calm

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

**Question 6.**

Without looking, how much time do you think it took you to complete the game?

|  |  |
| --- | --- |
| Minutes | Seconds |

**Test Documentation**

**Candidate No:**

**Quiz 2 – after the previous quiz has been done**

**Age:**

**Gender:**

**Email (only for competition):**

**Question 1.**

If you had to, which of the two versions would prefer to play?

|  |  |
| --- | --- |
| First | Second |

**Question 2.**

How many games would you say you have played in the last month?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| None at all | A little | medium | Quite a bit | A lot |

**Question 3.**

Which of the following games do you prefer to play?

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Action | Adventure | Puzzle | Platform | Shooter | RPG |
| MMO | Racing | Sport | Party | Fighting | Casual |
| Indie | Story Driven | Arcade | Horror | Survival | Rogue-Like |
| Strategy | Comedy | Serious | Simulator | Educational | Artistic |
| Light Hearted | Hard-core | Flying | Triple A | Local Multiplayer | Other |

**Question 4.**

Did you notice a difference between the two games?

|  |  |
| --- | --- |
| Yes | No |

**Question 3.**

If you said yes to Question 1 – Please note the differences between the two games.

**Question 4.**

Would you say the difference was a good thing or a bad thing?

|  |  |
| --- | --- |
| Yes | No |

**Test Documentation**

**Candidate no:**

**Supervisor notes:**

**Date:**

**Time:**

**Time to finish + Highscore:**

**Game 1:**

**Game 2:**

**Which game was Emotiv?:**

**Game 1**

**Game 2**

**Notes:**