

THE UNIVERSITY OF MANCHESTER

---

**CogniDriver**

---

*Author:*  
Maria - Daniela Florescu

*Supervisor:*  
Toby L. J. Howard

2015

## **Abstract**

### **CogniDriver**

**Author: Maria - Daniela Florescu**

The aim of the project is to provide an overview of the development of CogniDriver, a 3D mind-controlled car driving game which uses the Emotiv EPOC interface to control the car movement. The headset also allows the activation of various game effects such as raining when the frustration levels are too high or changing the camera view when the player does a left wink.

The Unity game engine has been used while developing the game in order to gain access to an advanced physics engine and to a multi-platform deployment tool.

The results of user testing on 13 participants have shown that although it is more difficult to control the game while in Cognitiv mode, the game appears more challenging and more interactive than a normal Keyboard mode play.

In the conclusions a short overview of the current deficiencies are presented as well as potential for further development.

**Supervisor: Toby L. J. Howard**

### **Acknowledgements**

I would firstly like to thank my supervisor, Toby Howard, who has provided great advice and support throughout the project.

I would also like to thank my family for their continuous support during my studies.

Finally, a big thank you to everyone who has helped with the testing and whose feedback was greatly appreciated.

# Contents

<b>1</b>	<b>Introduction</b>	<b>7</b>
1.1	Applications . . . . .	7
1.2	Aims and objectives . . . . .	7
1.3	Report outline . . . . .	8
<b>2</b>	<b>Background</b>	<b>9</b>
2.1	Development tools . . . . .	9
2.1.1	Unity . . . . .	9
2.1.2	Easy Roads 3D . . . . .	10
2.1.3	Photoshop . . . . .	10
2.2	Electroencephalography (EEG) . . . . .	10
<b>3</b>	<b>Introducing the Emotiv EPOC</b>	<b>12</b>
3.1	SDK description . . . . .	12
3.1.1	Caring for the headset . . . . .	15
3.1.2	Headset issues . . . . .	15
3.1.3	EmoComposer . . . . .	16
3.1.4	EmoKey . . . . .	16
3.2	Training . . . . .	16
3.3	Current Applications . . . . .	18
<b>4</b>	<b>Design</b>	<b>19</b>
4.1	Overview . . . . .	19
4.1.1	Project workflow . . . . .	19
4.1.2	Requirements . . . . .	20
4.1.3	Separation of concerns . . . . .	21
4.2	Design Patterns . . . . .	21
<b>5</b>	<b>Implementation</b>	<b>23</b>
5.1	Interfacing with Emotiv . . . . .	23
5.1.1	EdkDll . . . . .	23
5.1.2	EmoAffectiv . . . . .	23
5.1.3	EmoCognitiv . . . . .	24
5.1.4	EmoEngine . . . . .	24
5.1.5	EmoEngineInst . . . . .	24
5.1.6	EmoExpressiv . . . . .	24

5.1.7	EmoGyroData . . . . .	25
5.1.8	EmoProfileManagement . . . . .	25
5.1.9	EmoState . . . . .	26
5.1.10	EmoUserManagement . . . . .	26
5.1.11	LabelDraw . . . . .	26
5.1.12	Gyro mode . . . . .	26
5.2	Scene description . . . . .	26
5.2.1	CarChoiceScene . . . . .	27
5.2.2	HelpScene . . . . .	27
5.2.3	MainMenu . . . . .	27
5.2.4	MainScene . . . . .	28
5.2.5	SplashScreen . . . . .	28
5.2.6	TrainingScene . . . . .	28
5.3	Interesting aspects of development . . . . .	28
<b>6</b>	<b>Results and analysis</b>	<b>29</b>
<b>7</b>	<b>Testing and evaluation</b>	<b>30</b>
7.1	User testing . . . . .	30
<b>8</b>	<b>Conclusions</b>	<b>31</b>
8.1	Achievements . . . . .	31
8.2	Personal development . . . . .	31
8.3	Further work . . . . .	31
8.4	Summary . . . . .	31
	<b>Bibliography</b>	<b>32</b>
<b>A</b>	<b>Domain class diagram</b>	<b>34</b>

# List of Figures

2.1	A composition of alpha, theta and delta brainwaves. Credit: [Gall, 1992] .	10
3.1	Sensor positioning for the Emotiv EPOC. Credit: Emotiv . . . . .	13
3.2	AF3 and AF4 channel response to eye blink. Credit: [Adelson, 2011] . .	14
3.3	From left to right: clean inside of golden plate; clean outside of golden plate; oxidised inside of golden plate; oxidised outside of golden plate. . .	16
3.4	Cognitiv suite training profile . . . . .	17
4.1	Domain class diagram for CogniDriver. The instance variables have been omitted for simplification. . . . .	22
5.1	Scene interaction diagram from the moment the application starts loading.	27

# List of Tables

2.1	Information about brainwaves types [Ning-Han Liu and Hsu, 2013]. Image Credit: Hugo Gamboa . . . . .	11
3.1	Left: Abbreviations for sensor positioning; Right: Sensor contact quality and colour codes . . . . .	12

# Chapter 1

## Introduction

### 1.1 Applications

Using BCI (Brain Computer Interfaces) to control an application is of great use to persons with motor disabilities whose use of keyboard or mouse devices might not be convenient. In addition, BCIs provide a great way to exercising brain functions and improving concentration.

### 1.2 Aims and objectives

The aim of this project was to learn about the ... and falls of BCIs, how these interact with the computer and reach a conclusion about how they could be used in the future. Because the Emotiv EPOC Control Panel allows only up to 4 actions to be recognised at any one time, the simplest choice of a game was that of a car driving one since you can also observe the car moving in the dictated direction which can ease the activation of an action through the visual feedback.

The key objectives of the project were to:

- Allow the car to move in each one of the four directions: forward, back, left, right through keyboard use;
- Reach access to the interpreted data of the developers' SDK;
- Allow players to train new profiles inside the game;
- Provide an array of cars and colours that the user can select from.



## **1.3 Report outline**

The report begins with describing the used tools and provides some information about the way the brain works in Chapter 2. Next, Chapter 3 describes how the Emotiv EPOC works and how it is organised. In this chapter, the report also contains a quick look over some other applications developed with this technology.

Chapter 4 focuses on the description of the design of this project, while Chapter 5 oversees the implementation details. Chapter 6 reviews the obtained results, while Chapter 7 described the performed testing. Appendix 1 shows the user questionnaire used during the user trials. Finally, Chapter 8 contains the conclusions of this project.

# Chapter 2

## Background

This chapter aims to provide a general overview of the tools used while building CogniDriver. It also describes some basic concepts about how the brain works which would be useful to know before diving into the description of the headset.

### 2.1 Development tools

#### 2.1.1 Unity

Unity is a game engine which allows:

- Game deployment on a variety of platforms;
- Easy object importing;
- Object manipulation;
- Scene design;
- Creation of primitive shapes (sphere, cube, etc.);
- Collision detection;
- Automatic object updates;
- Game physics.

I should also highlight some of the reasons why Unity was chosen in favour of other game engines such as Unreal. Unity allows file import by a simple drag-and-drop and it also auto updates the objects if you change the source files using another program (e.g. Paint, Photoshop). The advantages of Unreal over Unity would be better graphics (mainly lighting) and the fact that it can fracture meshes. In terms of scripting, Unity allows development in JavaScript, C# and Boo, while UDK only allows UnrealScript. For CogniDriver, C# has been the scripting language choice, because it allows Object Oriented Programming (OOP) and it is scalable for larger projects.

Unity has been used to create games such as Assassin's Creed Identity, Need for Speed World and Battlestar Galactica Online.

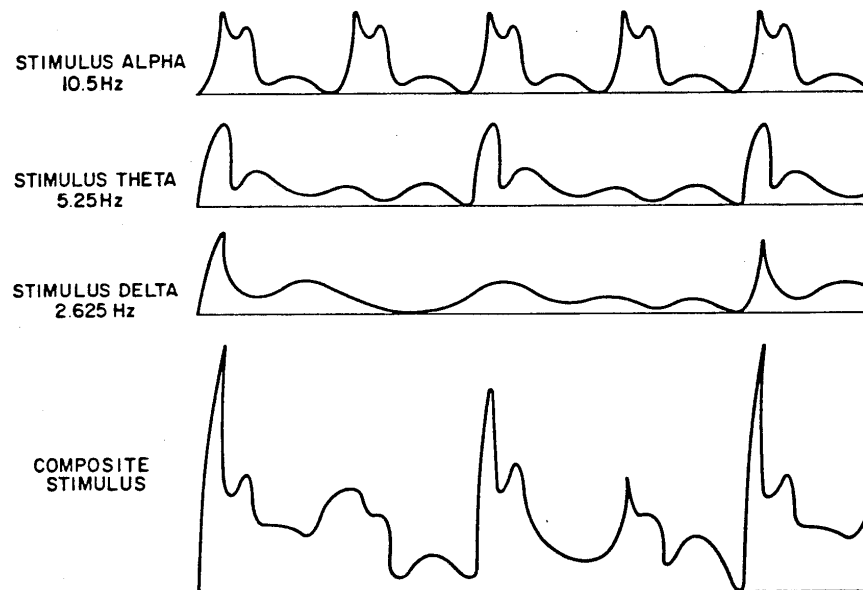


Figure 2.1: A composition of alpha, theta and delta brainwaves. Credit: [Gall, 1992]

### 2.1.2 Easy Roads 3D

This plugin is available in the Unity asset store. The free version has been used for this project. It allows placing markers through which the road should go, and given a specified width, it approximates corners and builds the rest of the road into a single layer.

### 2.1.3 Photoshop

A trial version has been used to create some materials and textures such as: skid marks, rain, smoke texture, speedometer, arrows, colours choice. None of these are difficult to create, but I like Photoshop because it is easy to use by both novice and professional users. In addition, the layers allow the user to create many compositions and modify the image at will.

## 2.2 Electroencephalography (EEG)

The neuron is the core component of the nervous system. More neurons can connect together to form synapses and when one of these cells is excited, an electrical signal is generated. The electrical signals can be amplified and the resulting waveforms, known as brainwaves, can thus be obtained. They can then be used to monitor the brain activity of a person in order to detect abnormalities. In table 2.1, a representation of each type of most common brainwave types is shown, together with the normal activities in which these occur most often.

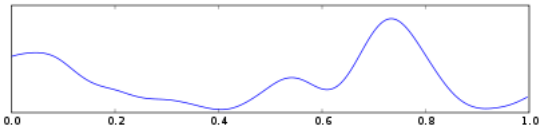
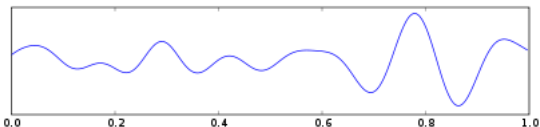
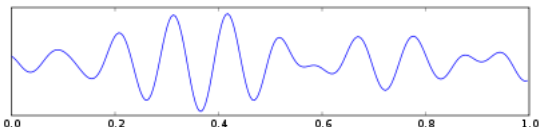
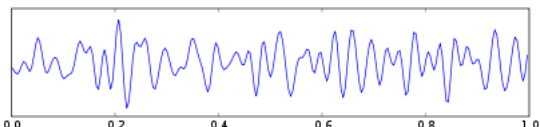
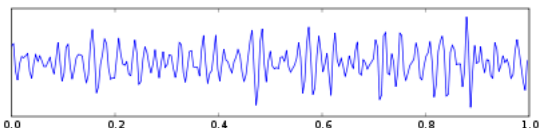
Name	Frequency (Hz)	Occurs in	Image Representation
Delta	<4	deep sleep , unconsciousness, deep anaesthesia, hypoxia	
Theta	4 - 7	stress, unconsciousness, deep relaxation	
Alpha	8 - 15	relaxation	
Beta	16 - 31	consciousness, alertness, thinking, sensory stimulation	
Gamma	32 +	selective attention, human cognition and perceptive activities	

Table 2.1: Information about brainwaves types [Ning-Han Liu and Hsu, 2013]. Image Credit: Hugo Gamboa

Please note that almost always the EEG will display a composition of a set of types at any point in time. One example can be seen in the figure 2.1. Brain Computer Interfaces (BCIs) are a channel of communication between the electroencephalograph (EEG) and the computer.

## Chapter 3

# Introducing the Emotiv EPOC

### 3.1 SDK description

The Emotiv EPOC headset is a portable EEG device which consists of 14 sensors that transmit raw EEG data (see figure 3.1 for their positioning) to the computer and 2 reference sensors (CMS - Common Mode Sense and DRL - Driven Right Leg). The CMS sensor is the point on the scalp against which everything else is measured. The DRL sensor provides a feedback signal to cancel common mode noise in the electronics [Emotiv, 2015a, BioSemi, 2015]. See table 3.1 for the meaning of the abbreviations of sensor positions. The raw data gets interpreted and the user gains access to all of the information in one of the following 4 suites: Expressiv, Affectiv, Cognitiv, Mouse Emulator.

Abbreviation	Meaning	Colour code	Meaning
A	Ante	Black	No signal
F	Frontal	Red	Very poor signal
C	Central	Orange	Poor signal
P	Parietal	Yellow	Fair signal
T	Temporal	Green	Good signal
O	Occipital		

Table 3.1: Left: Abbreviations for sensor positioning; Right: Sensor contact quality and colour codes

I have been using the developer's SDK, so it is worth highlighting I did not have any access to the raw EEG data.

The battery lasts about 10 to 12 hours. The charging is done via a USB cable.

According to [Adelson, 2011], the data sampling rate is 128Hz. [emo, 2014b] states that there may be up to 2 seconds delay from when the data is sent until it is received.

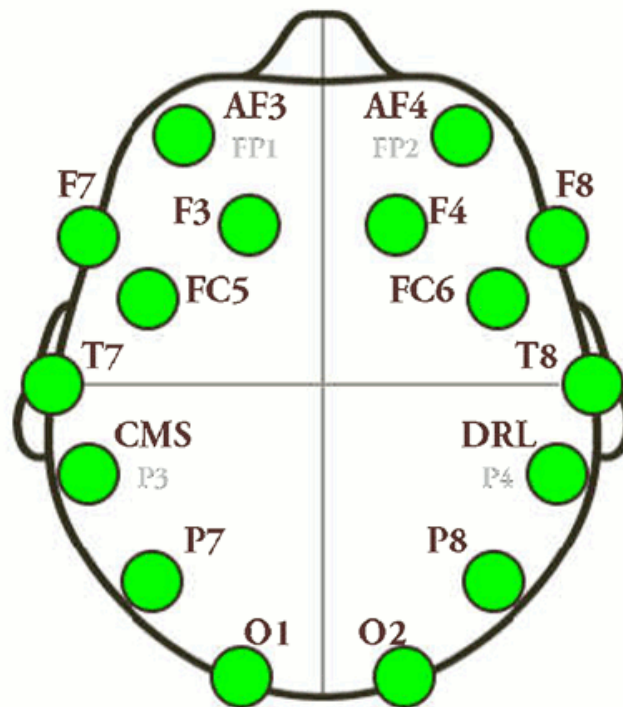


Figure 3.1: Sensor positioning for the Emotiv EPOC. Credit: Emotiv

### Expressiv<sup>TM</sup>Suite

Detects 12 facial expressions:

- |               |                  |
|---------------|------------------|
| 1. Blink      | 7. Laugh         |
| 2. Left wink  | 8. Smirk left    |
| 3. Right wink | 9. Smirk right   |
| 4. Look left  | 10. Raise brow   |
| 5. Look right | 11. Furrow brow  |
| 6. Smile      | 12. Clench teeth |

One thing to note is that if the sensor contact quality is not perfect (all green - see table 3.1), blink and winks might be mistaken; same applies to laughs and smiles. In fact, an ideal signal is obtained when all sensors display green. It is acceptable to have some displaying yellow but anything less than yellow, means data might be unreliable or unavailable.

The API usually gives back data as a `boolean`, but for smile, clench and look left/right, a `float` between 0 and 1 can also be obtained which represents the extent of that expression.

Figure 3.2 shows an image of how the EEG looks like in channels AF3 and AF4 after a blink. See figure for a screenshot of the suite.

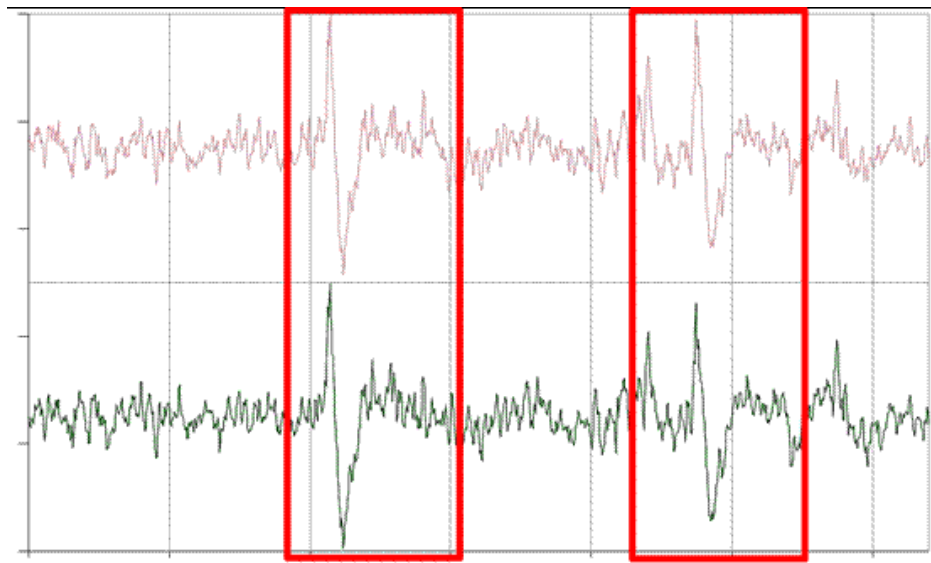


Figure 3.2: AF3 and AF4 channel response to eye blink. Credit: [Adelson, 2011]

### Affectiv<sup>TM</sup>Suite

Returns a `float` between 0 and 1 which describes the power of one of the following emotions:

- Engagement/Boredom
- Frustration
- Meditation
- Excitement short term
- Excitement long term

I have found these emotions to represent quite well my mood. One thing to note is that boredom is not the kind of boredom one feels when they have nothing to do, but it is rather the opposite of engagement, according to Emotiv. See figure for a screenshot of the suite.

The Emotiv User Manual [emo, 2014b] (page 31) provides information on the activities and brainwave types that may trigger each emotion.

### Cognitiv<sup>TM</sup>Suite

This is probably the most exciting feature of the Emotiv EPOC. Here, you can train 13 actions: push, pull, left, right, lift, drop, 6 rotations (one in each direction for the x, y and z axis) and disappear. The disappear action is more abstract because while the first 12 are real life actions, an object cannot vanish into thin air through whatever methods. The user has to train each action for a period of 8 seconds. This should be done multiple times until an as high as possible skill level is obtained. The more you get used to training, the easier it will become. I have achieved close to 100% on all 4 actions: push, pull, left, right. At

one point, I was able to achieve 100% on the push action in 5 trainings. The SDK can only detect up to 4 actions at any one time, so one has to choose wisely which actions would be best for their application. See figure 3.4 for a screenshot of the suite.

### **Mouse Emulator (Gyro)**

This data actually comes from a gyroscope fitted in the headset. The original version aims to emulate the mouse movement by changing the head position. Once your head turns to left or right and remains in that position, the centre of the circle becomes your new head position. However, I had access to the Unity 3D plugins and I was able to modify this suite so that the centre of the circle was not recalculated if the head stayed to the left or to the right because I wanted to use the gyro as yet another way to control the car movement. See figure for a screenshot of the suite.

### **3.1.1 Caring for the headset**

The 16 sensors need to be properly hydrated before starting to use the Emotiv EPOC. The hydration is done with contact lenses saline solution which has the purpose of enabling a better transmission of the brainwaves while also sanitising the sensors. First time using the headset may need 15-20 proper hydrations before good contact quality is achieved.

When the user does not expect to use the headset in the next 3-4 days, it is recommended to remove the felt parts from the sensors. Otherwise, the process of oxidation happens more quickly. See figure 3.3 for images of clean sensors and oxidised sensors. The green residue can be cleaned with a cotton swab dipped into a solution of 1/2 water 1/2 white vinegar. If cleaning the interior of the golden plates, take care not to remove the slightly white polymer paste. I have left the felt parts in the sensors when I went on holiday for about 3 weeks. When I returned, the headset was losing contact quality very quickly and after about 15 minutes, it had to be rehydrated. It looks like salt columns may also form in the felt parts. My problem though, was that the golden plates oxidised a lot and after cleaning them as described above I achieved full green sensor contact quality which is the best you can wish for. I firstly cleaned the outside of the golden plates but that did not help too much. Then, I moved to cleaning the inside as well and that did the job for me, although on the Emotiv forums this is not something they would recommend. I was amazed to see the great contact quality I could achieve and it did not have any negative effects on the sensors and they are still in perfect shape 3 months after the cleaning.

### **3.1.2 Headset issues**

One of the main problems I encountered is that the wireless connection to the computer is not reliable and it can drop out of the blue. The EPOC+ is now on the market and it seems it might come with a solution for this.





Figure 3.3: From left to right: clean inside of golden plate; clean outside of golden plate; oxidised inside of golden plate; oxidised outside of golden plate.

### 3.1.3 EmoComposer

EmoComposer is a tool which sends simulations of EmoEngine events to an application. This made it easy to test my game without connecting the headset. This was particularly useful in more tricky situations which would have meant spending a long time trying to achieve an action by concentration only.

### 3.1.4 EmoKey

EmoKey allows mapping of keys to all the interpretations described in this chapter. In this way, one can use the headset to send smiley faces when smiling or laughing for example. One may also use it to paint. These are two of the things that I have tried. It is supposed to be more useful for playing games using the headset. This has been tried in [Reinhold Scherer, 2012] for the *World of Warcraft* game.

## 3.2 Training

Because training is a difficult part of getting started with the headset, this section is only trying to provide some pointers as to what might work.

First of all, while training the user should try to keep as still as possible and concentrate on the action being trained. For example, if one is trying to train the push action, the thing which works most of the times is to visualise how the training cube is being pushed to the back of the virtual room.

Other people find it easy to visualise a flow of energy in the direction of the trained action.

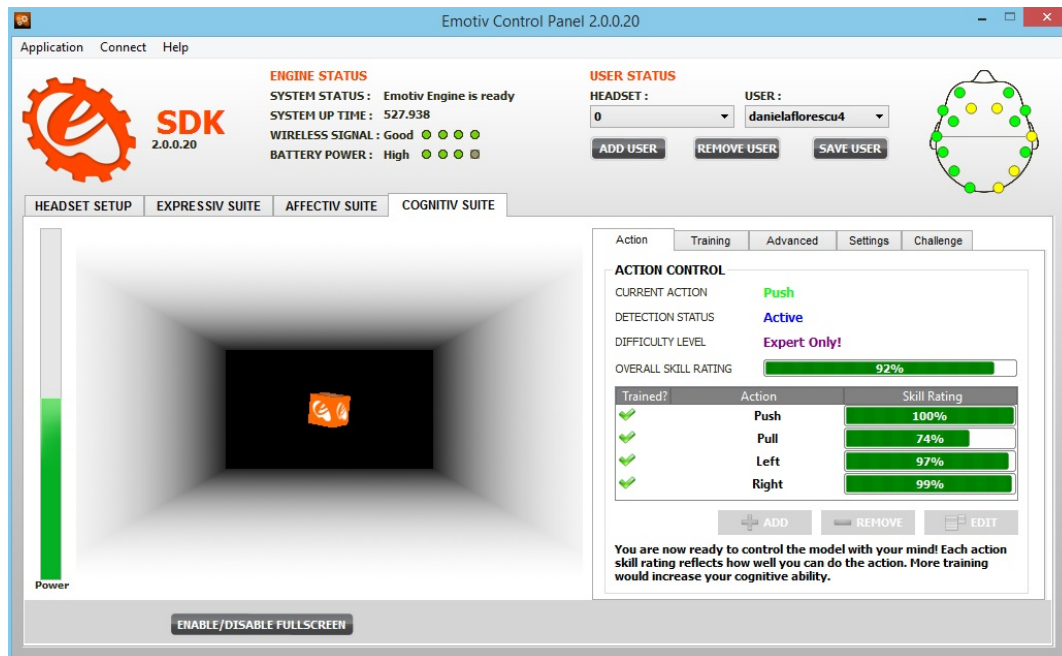


Figure 3.4: Cognitiv suite training profile

Repeating words into one's mind does not work. I have seen it working when people say words out loud but the same pace should be kept.

Doing hand gestures works sometimes as [Emotiv, 2015b] seems to suggest. I have seen one user trying to do this. It worked for a while but then it stopped. Same happened to me.

One thing I have not tried is shifting concentration on the left or right hand side of the body as it is also suggested in [Emotiv, 2015b].

Imagining colours or objects for different actions, also does not seem to work.

The problem with training is that there is no 'recipe' one can follow. Rather, the user has to try to come up with something that works for them.

With regards to the training success, in figure 3.4 you can see one of my achievements. [Emotiv, 2015b] the fastest learners seem to be young children who believe they can do anything, and very relaxed elderly. In the given example, the father of the poster, aged 82, was able to train reliably the 4 actions in less than 5 minutes. For some notes on how my user testing training worked, read chapter 7.

Some people might encounter difficulties in training the headset. According to [Carmen Vidaurre, 2009], between 15% and 30% of the people trying to use a BCI, will not be able to do so. This condition is called 'BCI illiteracy' and it better analysed by [Minkyu Ahn, 2013]. Their study suggests these people have high theta and low alpha waves present across different mental states such as non task related, resting before motor imagery and motor imagery. [Carmen Vidaurre, 2009] have tried to come with a solution to this by using a subject-optimised classifier.

### **3.3 Current Applications**

In this section, I am going to have a quick go-through some of the existent applications developed with the help of the Emotiv EPOC.

- NeuroPhone [Andrew T. Campbell and Raizada, 2010] is an application which uses the P300 signal and allows the user to scroll through photos of 6 contacts at a time. Once the person the user is looking for is found, the application will dial that person's number.
- [Sam Fok, 2011] have developed a system which allows the control of a hand prosthetic which opens and closes a patient's hand.
- [Francesco Carrino, 2012] shows it is difficult to successfully control applications in a highly error sensitive context, such as a wheelchair.
- Mindtunes [Smirnoff, 2013] is a project in which DJ Fresh has used multiple headsets in order to produce music from the raw EEG data.
- EmoLens [Inc., 2010] tags a user's Flickr photos with emotions and learns to show similar pictures in the future as well as allowing picture search based on the tagged emotion.

# Chapter 4

## Design

This chapter describes the design decisions taken in this project and how these work towards achieving the aims described in chapter 1.

### 4.1 Overview

As a methodology used during the project, I have followed the Agile Unified Process. This decision was taken because the process is iterative and it allows the user to be flexible if circumstances change. Agile UP also focuses on producing working code over comprehensive documentation.

For splitting the tasks, I have been using <http://trello.com>, a free project management tool which allows you to have virtual boards and cards for each task, making it a virtual Kanban system.

For producing the UML diagrams, I have used <http://www.gliffy.com/>, an online tool which allows the user to create 5 diagrams for free.

#### 4.1.1 Project workflow

The project has consisted of 4 phases:

- **Phase 1** consisted of initial project documentation, background reading and an attempt to build a 3D car model to be used during the game. The sides of the car model were realistic enough, but I could not figure out how to build the front and the back, so I have used 3 free 3D models that I have found online. See Appendix for the sources of these objects.
- During **phase 2**, I have learnt more about Unity by following the [Studio, 2014] tutorial. Following that, [Tutorials, 2014] was a good starting point to how one can create a car game in Unity. Some more advanced car physics ideas were inspired from [Monster, 2015].

- **Phase 3** has seen the start of implementing the functionality from the developer's API of the Emotiv EPOC. The most important part was getting the training working inside the game.
- In **Phase 4** the main concern was improving the user experience, getting some user testing done and allowing the player a choice of car models and colours.

## 4.1.2 Requirements

### Functional requirements

In the final game, the player should be able to:

- Create/Delete/Change a player profile;
- Train the headset for neutral, push, pull, left, right states;
- Allow the user to accept/reject/reset a training session;
- Display the updated skill level after each training session;
- Provide visual feedback as to how well the training is going through an animated object;
- Alert the user in the training scene if a training profile is not complete;
- Display the top 10 highest scores for Keyboard mode play and Cognitiv mode play;
- In the Options tab, allow the user to change sound volume, sound effects volume and whether the game will be played in fullscreen;
- Allow the user to change between Keyboard, Cognitiv and Gyro play modes;
- Provide instructions to train and play the game;
- Pause/Exit the game;
- Allow the selection of a car model and colour;
- Race yourself during the game play.

### Non-functional requirements

- The game should be portable (i.e. playable on multiple operating systems);
- Once loaded, there should be no long waiting periods of time.

### 4.1.3 Separation of concerns

The Model-view-controller architecture has been used with the purpose of separating the concerns such that the controller (the script) is manipulating the model (car or environment) which then updates the view (the user interface).

## 4.2 Design Patterns

**Abstract Factory** has been used to create cars from the abstract `Car` class. `AudiR8` and `FerrariCalifornia` are thus inheriting from `Car`. See figure 4.1 for the UML class diagram.

**Strategy** is used to decide which car control options to follow depending on the game play mode: Keyboard, Cognitive or Gyro. It is also used in the car model choice.

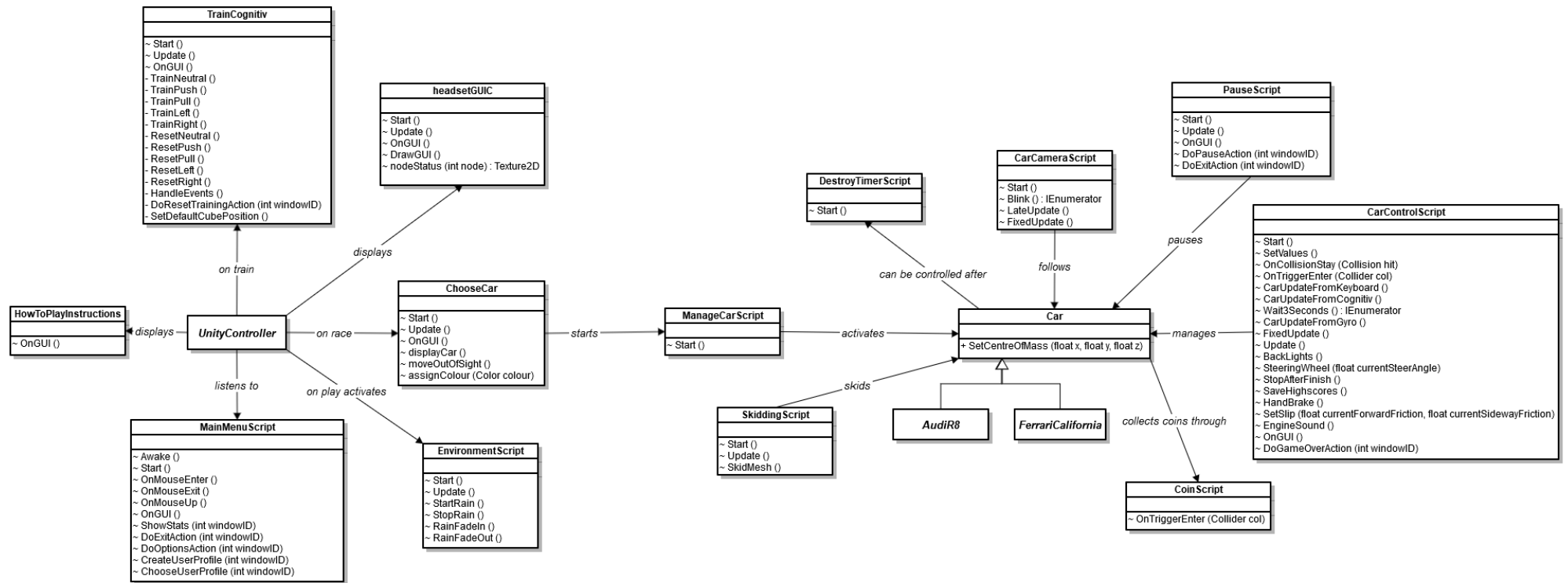


Figure 4.1: Domain class diagram for CogniDriver. The instance variables have been omitted for simplification.

# Chapter 5

## Implementation

This chapter is going through some of the implementation details throughout the development of the project.

### 5.1 Interfacing with Emotiv

The Unity 3D plugin package from Emotiv comes in a suite of 11 C# files which we will describe in a bit more detail. In order to get access to this data, an `EPOCManager` object was created. This object contains all the plugins in the package apart from `EdkDll`, `EmoState`, `EmoProfileManagement` and `LabelDraw`. In the following subsections, I am going to provide some very short code snippets since these might prove useful in the future for someone wishing to develop using Unity and Emotiv.

#### 5.1.1 EdkDll

Contains the implementation of the Emotiv API.

#### 5.1.2 EmoAffectiv

This class gives access to the player emotions described in section 3.1 through calls by directly accessing one of the public instance variables. For example,

Beginning of code

```
EmoAffectiv.frustrationScore
```

End of code

returns a value representing how frustrated the user is.

In the game, frustration is used to get the rain started once the level is percentage is higher than 50%.



### 5.1.3 EmoCognitiv

It is a class containing methods to start or reset the training. It automatically handles events which start, reset, erase, declare successful or failed training. I have added accessor methods to obtain the current action, its power and the trained actions. I have also added a GUI method, `DoTrainingCompleteAction(int windowID)` which pops up the window asking the user whether the training should be accepted or rejected and it handles the response appropriately. Before starting the training though, the 4 actions (push, pull, left, right) have to be activated through a call such as:

```
_____ Beginning of code _____  
EmoCognitiv.EnableCognitivAction(EmoCognitiv.cognitivActionList[6],  
                                true);  
_____ End of code _____
```

where 6 represents the index of *right* action. At the end, the list of 4 actions has to be set through a call to

```
_____ Beginning of code _____  
EmoCognitiv.EnableCognitivActionsList();  
_____ End of code _____
```

The skill for the *push* action is obtained as follows:

```
_____ Beginning of code _____  
Single pushSkill = EmoEngine.Instance.CognitivGetActionSkillRating(  
    (uint)EmoUserManagement.currentUser,  
    EmoCognitiv.cognitivActionList[1]);  
_____ End of code _____
```

### 5.1.4 EmoEngine

‘Is the logical abstraction of the functionality that Emotiv provides in edk.dll.’[emo, 2014a]

### 5.1.5 EmoEngineInst

It is used most often to change the connection method from live data to simulated data (through the use of `EmoComposer` (subsection 3.1.3)).

### 5.1.6 EmoExpressiv

The `EmoExpressiv` class is obtained to get access to the player’s facial expressions through simple calls to the public instance variables, such as the following which checks whether the player did a left wink:

```
_____ Beginning of code _____  
EmoExpressiv.isLeftWink  
_____ End of code _____
```

In CogniDriver, left wink is used to change the camera view. Clenching teeth will action the car's handbrake.

### 5.1.7 EmoGyroData

Gives access to the head position in relation to the circle observed in figure. One may also obtained the x and y coordinates of the gyro. However, I could not find the maximum and minimum values, nor are there 2 online references which give the same numbers.

### 5.1.8 EmoProfileManagement

This class is useful to handle the player profiles. In CogniDriver, I have limited the number of player profiles to 10 because of the following reasons:

- The training for a saved profile can take a few MB;
- Unity does not contain a drop down list as a GUI element;
- Unity does not allow overflow of windows.

When the game starts, the list of user profiles (files saved with the .up extension) is loaded through a call to:

```
_____ Beginning of code _____  
EmoProfileManagement.Instance.LoadProfilesFromFile();  
_____ End of code _____
```

A new profile is added by calling:

```
_____ Beginning of code _____  
EmoProfileManagement.Instance.AddNewProfile(playerName);  
_____ End of code _____
```

Profile data is saved after training by a call to

```
_____ Beginning of code _____  
EmoProfileManagement.Instance.SaveCurrentProfile();  
_____ End of code _____
```

followed by

```
_____ Beginning of code _____  
EmoProfileManagement.Instance.SaveProfilesToFile();  
_____ End of code _____
```

On selection of a profile, this is set by calling

```
_____ Beginning of code _____  
EmoProfileManagement.Instance.SetUserProfile(selectedPlayer);  
_____ End of code _____
```

A profile is delete by calling

```
_____ Beginning of code _____  
EmoProfileManagement.Instance.DeleteProfile(selectedPlayer);  
_____ End of code _____
```

### 5.1.9 EmoState

Represents the emotional status of the user at a given time [emo, 2014a]. This class is used by `EmoAffectiv`, `EmoExpressiv` and `EmoCognitiv` mainly. I did not need to call any functions directly from this class.

### 5.1.10 EmoUserManagement

Keeps track of the current user, the current number of user profiles and also fires events when adding or deleting a user. I did not need to use this class directly apart from obtaining the current user by calling

```
_____ Beginning of code _____  
EmoUserManagement.currentUser  
_____ End of code _____
```

### 5.1.11 LabelDraw

I did not need to use this class while developing `CogniDriver`.

### 5.1.12 Gyro mode

Playing the game while using the headset's gyroscope is doable for forward/backward actions. However, most of the times the user is getting eyes off the screen meaning they may not see where and how the car is moving.

## 5.2 Scene description

In Unity, a scene contains all the objects for a part of the game such as a level or a different screen. For `CogniDriver`, I have split the game in 6 scenes. All of the scenes, apart from the splash screen, will display the sensor contact quality, current action and the power of the current action. This can be observed in figure.

See figure 5.1 for an example of how the scenes interact with each other.

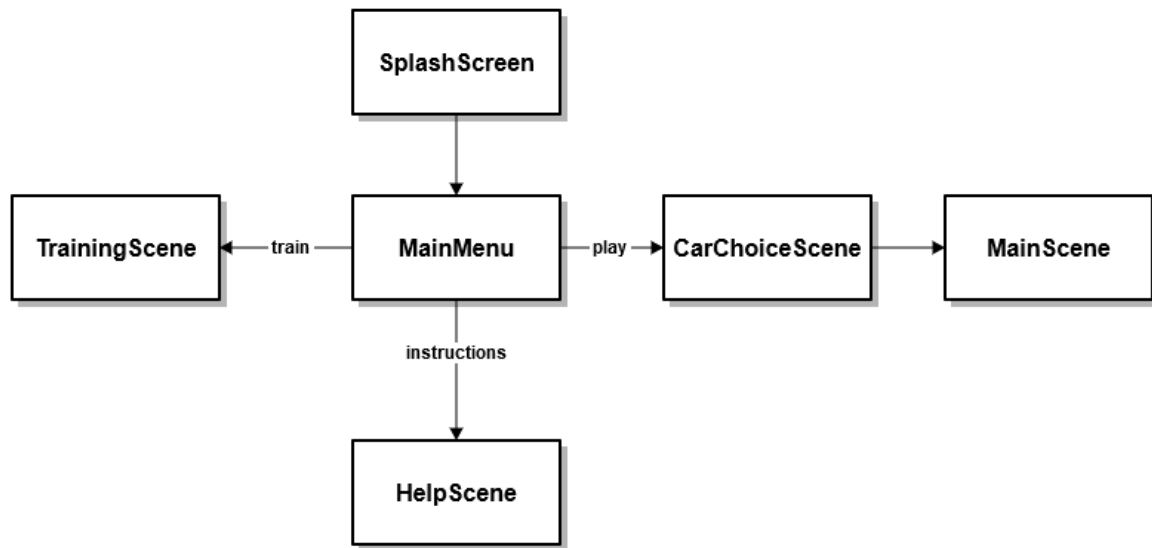


Figure 5.1: Scene interaction diagram from the moment the application starts loading.

### 5.2.1 CarChoiceScene

The purpose of this scene is to allow the player to choose a car model and colour. At the moment, I have only used 2 car models. I had three of them in mind but one of the models did not allow me to modify the grouping of its components.

### 5.2.2 HelpScene

It is simple instructions describing how the game is to be played. Figure shows an example of how the instructions are displayed to the player.

### 5.2.3 MainMenu

This scene allows the player to:

- manage the player profiles;
- start the race;
- train a player profile to be played in Cognitiv mode;
- see the highscores for Keyboard and Cognitiv mode in *Statistics* tab;
- read instructions about training and playing the game;
- adjust the sound effects and music volume, fullscreen enabling and select the play mode;
- exit the game.

Figure shows an example of how the main menu scene looks like.

#### **5.2.4 MainScene**

This is the proper game play where the player is controlling the car around the track.

#### **5.2.5 SplashScreen**

This represents the image the player is presented with during the initial loading of the game. The used image can be observed in figure.

#### **5.2.6 TrainingScene**

Its aim is to provide a place where the player can train the 5 actions: neutral (staying relaxed), push, pull, left, right. The training may also be reset for each action in turn. The user will receive visual feedback from the animated car on how well the training has been done. A screenshot can be observed in figure.

### **5.3 Interesting aspects of development**

I took the decision of having the Cognitiv training done inside the game so that a new player would not need to open another application, for example the Emotiv Control Panel, in order to train the profile.

For passing variables between different scenes, Unity's `PlayerPrefs` module has been very useful. One example of its use is in saving the highscores.

## **Chapter 6**

### **Results and analysis**

# **Chapter 7**

## **Testing and evaluation**

### **7.1 User testing**

# **Chapter 8**

## **Conclusions**

### **8.1 Achievements**

### **8.2 Personal development**

### **8.3 Further work**

### **8.4 Summary**



# Bibliography

- [emo, 2014a] (2014a). *Emotiv Software Development Kit User Manual for Release 2.0.0.20*. Emotiv.
- [emo, 2014b] (2014b). *EPOC User Manual*. Emotiv.
- [Adelson, 2011] Adelson, M. (2011). Experimenter epoc - an experimentation and mind-reading application for the emotiv epoc. [http://compmem.princeton.edu/experimenter/ExperimenterReport.html#\\_Toc290642075](http://compmem.princeton.edu/experimenter/ExperimenterReport.html#_Toc290642075).
- [Andrew T. Campbell and Raizada, 2010] Andrew T. Campbell, Tanzeem Choudhury, S. H. H. L. M. K. M. M. R. and Raizada, R. D. S. (2010). Neurophone: Brain-mobile phone interface using a wireless eeg headset. *MobiHeld/ACM*.
- [BioSemi, 2015] BioSemi (2015). Reference sensors. <http://www.biosemi.com/faq/cms&drl.htm>.
- [Carmen Vidaurre, 2009] Carmen Vidaurre, B. B. (2009). Towards a cure for bci illiteracy. *BMC Neuroscience*. doi:10.1186/1471-2202-10-S1-P85.
- [Emotiv, 2015a] Emotiv (2015a). Emotiv reference sensors. <https://www.emotiv.com/forum/forum4/topic3409/messages/>.
- [Emotiv, 2015b] Emotiv (2015b). Emotiv training. [http://emotiv.com/forum/messages/forum4/topic114/message392/?phrase\\_id=464885#message392](http://emotiv.com/forum/messages/forum4/topic114/message392/?phrase_id=464885#message392).
- [Francesco Carrino, 2012] Francesco Carrino, Joel Dumoulin, E. M. O. A. K. R. I. (2012). A self-paced bci system to control an electric wheelchair: evaluation of a commercial, low-cost eeg device. *Biosignals and Biorobotics Conference (BRC), 2012 ISSNIP*. doi: 10.1109/BRC.2012.6222185.
- [Gall, 1992] Gall, J. (1992). Method and system for altering consciousness. <http://www.google.com/patents/US5123899>. US Patent 5,123,899.
- [Inc., 2010] Inc., A. T. (2010). Emolens. [https://www.youtube.com/watch?v=E9\\_XZlHoSp0](https://www.youtube.com/watch?v=E9_XZlHoSp0).
- [Minkyu Ahn, 2013] Minkyu Ahn, Hohyun Cho, S. A. S. C. J. (2013). High theta and low alpha powers may be indicative of bci-illiteracy in motor imagery. *PLoS ONE 8(11): e80886*. doi:10.1371/journal.pone.0080886.
- [Monster, 2015] Monster, M. (2015). Car physics for games. <http://www>.

asawicki.info/Mirror/Car%20Physics%20for%20Games/Car%20Physics%20for%20Games.html.

- [Ning-Han Liu and Hsu, 2013] Ning-Han Liu, C.-Y. C. and Hsu, H.-M. (2013). Improving driver alertness through music selection using a mobile eeg to detect brainwaves. *Sensors*. <http://www.fermentas.com/techinfo/nucleicacids/maplambda.htm>.
- [Reinhold Scherer, 2012] Reinhold Scherer, Markus Prill, B. A. . G. R. M.-P. (2012). New input modalities for modern game design and virtual embodiment. *Virtual Reality Short Papers and Posters (VRW), 2012 IEEE*. doi: 10.1109/VR.2012.6180932.
- [Sam Fok, 2011] Sam Fok, Raphael Schwartz, M. W. C. H. J. Z.-T. S. D. B. E. L. (2011). An eeg-based brain computer interface for rehabilitation and restoration of hand control following stroke using ipsilateral cortical physiology. *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*. doi: 10.1109/IEMBS.2011.6091549.
- [Smirnoff, 2013] Smirnoff (2013). Mindtunes. <https://www.youtube.com/watch?v=PgfxKZiSCDQ>.
- [Studio, 2014] Studio, W. B. (2014). Unity tutorial. [http://walkerboystudio.com/html/unity\\_course\\_start\\_here\\_\\_free\\_.html](http://walkerboystudio.com/html/unity_course_start_here__free_.html).
- [Tutorials, 2014] Tutorials, F. (2014). Unity car game tutorial. <https://www.youtube.com/watch?v=c5Snsi68xzE&list=PL67XFC3MYQ6K0PXSad15xFrhxNL76r4Te>.

# **Appendix A**

## **Domain class diagram**