Progress Report

Game development project

sean khanna – Q11279516

Level 6

Contents

[Introduction 2](#_Toc2090570)

[Background 2](#_Toc2090571)

[Engine or Language to use 2](#_Toc2090572)

[Defenition 3](#_Toc2090573)

[Gaussian 3](#_Toc2090574)

[Gaussian Blur 3](#_Toc2090575)

[Morphological Transformations 4](#_Toc2090576)

[Dilation 4](#_Toc2090577)

[Erosion 4](#_Toc2090578)

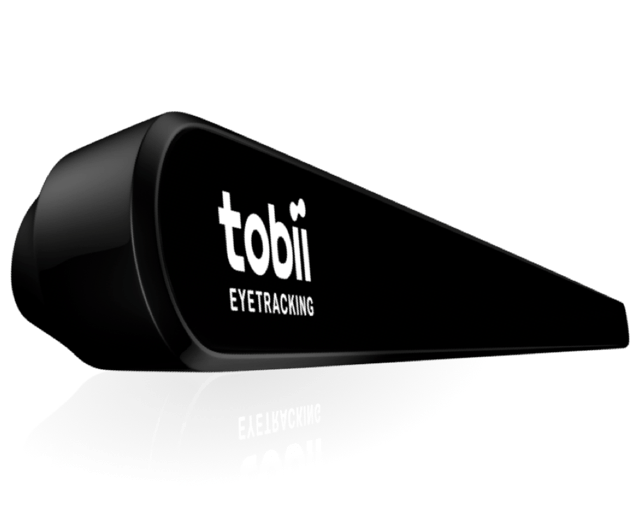
[Initial Design 5](#_Toc2090579)

[Project Plan 5](#_Toc2090580)

[References 7](#_Toc2090581)

[Appendix 8](#_Toc2090582)

# Introduction

This project started out as a plan to replace face recognition tools were you are required to buy extra hardware for it to work. The idea was that the tool was going to replace other existing tools and make it easier for the gaming community to create games which used a pre-owned camera to control parts of the game. There was already some competition out there such as tobii (2019). tobii is one of the lead eye tracking software out now, which uses their own cameras to track eyes. When creating tools or products using their products you are required to buy one of their cameras to use when creating any tool. Therefore, for this project the idea was to remove the part of having to buy hardware and use what people already owned, such as web cameras or laptop cameras.

tobii, 2009

After further research I found that there are quite a few other tools out, which allows you to use any camera and create your own programs with additional features. Some of these tools are very thorough and can use over 10 different forms of AI algorithms such as neural networks, Eigenfaces, Fisher faces and more which will be talked about later. Consequently, the product will now be a program that uses one of these tools, with a technical demo demonstrating what is going on and the program recognising certain events like: winking, the mouth opening and more.

# Background

There are many games on the market that have the capability of eye tracking unfortunately, my research came up short when I couldn’t find any released games or tools which use a simple web camera, all my findings lead to games that were created with the use of tobii (2019). There is a long list of games that use tobii, for example, Assassin’s Creed Syndicate and SOMA utilise the tobii camera for aiming using your eyes.

GameCrate, 2016

As said before there is no market for any games that use a person’s web camera to control the game therefore, this project will aim to fill that void.

## Engine or Language to use

After thorough research the program is going to be written in C++ because Unity only offers two official computer vision libraries that could aid the project the first was VisionLib. VisionLib is mostly used for Augmented Reality and helps more with model tracking, in layman terms you feed it a 3D model and you can use it with AR glasses or similar, to assist tracking. The second is OpenCV, however using it through Unity meant they must put a price tag upon it, of $65. Furthermore, I looked into using Unreal Engine, unfortunately, my research came up short and could not find any libraries to use with unreal except using OpenCV and using that within my code. C# also has a library which uses OpenCV, this is called EmguCV. This tool is cross platform, and wraps OpenCV in a .NET, allowing any .NET application, that are compatible, to use it, for example C#, VB, VC++ and any IDE that can compile it. This however, is time consuming for the project having to learn a new library and how to use a .NET wrap version of OpenCV. In the end, it came down to using OpenCV but, this can be used within three languages, Java, Python and C++. Unfortunately, my knowledge in Java and Python is little or next to none therefore, I have decided to write my program in CMD in C++.

* How do they work
* How suitable are they for your project

# Definition

The language chosen for this project is C++ using the OpenCV libraries, it has a wide range of functionality and the OpenCV website fully documents all of the functions, filters and other areas of its library. OpenCV is a very robust tool which uses many algorithms to help with face detection, in different ways. There are 4 main types of face detection that OpenCV offer: Eigen faces, Fisher faces, Local Binary patterns histograms and Haar cascades.

## Eigen faces

Eigen faces began out as the principal components of a distribution of faces and the idea was first created by Sirovich and Kirby in 1987. It uses a variety of face images to convert and relate individual faces in a big picture fashion.

### Steps

Eigen faces takes in a image and inputs each pixel into a vector, in the end once you have entered all the images you wanted into vectors you then end up with multiple sets of 2 dimensional date in one vector, these are called Eigen Vectors. In turn, each of these vectors put together can be treated as a matrix. With this matrix we apply PCA (see below) to it by, subtracting the mean, the mean here would be the average vector of all the images. Next you compute the covariance matrix of each eigenvector by finding the mean between the average eigenvector created before and the original eigenvector. Currently, we have a new set of eigenvectors that we can put back into an image and it will show how each image deviates from the average picture. Now, we can use the a different image, preferably one that is similar to one that was calculated into an eigenvector for best results, and do the dot product on that with each of the eigenvectors created, add the mean image and eventually, with the more dot products created and added, you will eventually get the an image which looks very similar to the input image, and finally, we can then use this to do face similarities.

#### PCA

Principal component analysis is a way of converting data that is possibly related into a lineal unrelated set of data which is called principal components, it allows as explained by Matt Brems(2017) “…we can drop the “least important” variables while still retaining the most valuable parts of all of the variables!” . Matt further talks about how we can use this data to be able to input new variables into the graph.

## Fisher faces

## Local Binary patterns histograms

## Haar cascades

* How will you use it to get the desired result
  + Filters

There are certain parts to OpenCV that allow me to do specific tasks such as reduce noise. To do this I will need to use some filters to help smooth out my image and make the information on screen more readable to the program, allowing for better results when tracking.

## Gaussian

Probability theory is a part of mathematics that deals with random events within quantities. Within probability theory the “normal distribution” or Gaussian is when there is a “random” anomaly. However, a guy called Galileo Galilei, questioned it and found that this phenomenon was symmetrical around a central value.

### Gaussian Blur

Gaussian blur is a way of processing an image and blurring out any part of an image, usually to reduce the noise.



OpenCV, 2019

## Morphological Transformations

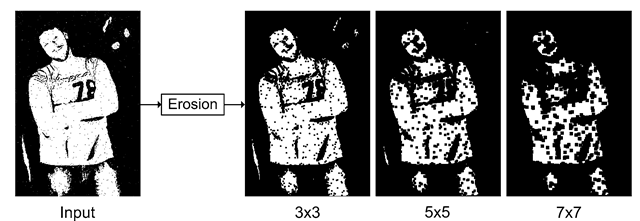
### Dilation

Dilation is when you take an image and brighten all the brighter parts of an image.

### Erosion

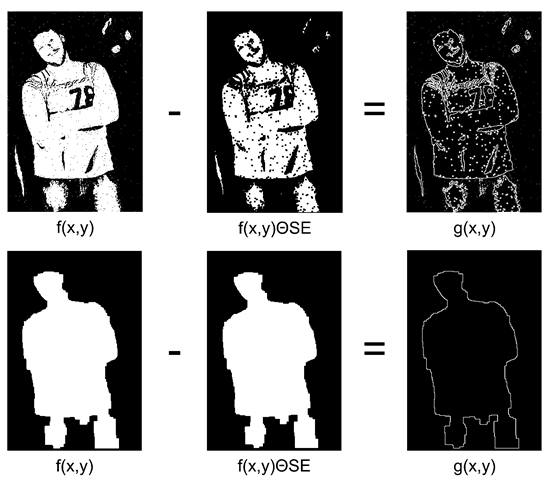
Erosion does the exact opposite and makes the image darker.

These can be used within images to reduce things such as reflections or bright lights in the background. While using erosion on a black and white image you can begin to define outlines of certain things see image below.



What-when-how, Unknown date.

Later we can use this to start doing edge detection called ‘boundary detection’ for example take the previous image and minus it from the input and we have our boundaries.



What-when-how, Unknown date.

# Initial Design

* Pseudocode or flowchart showing core functionality

# Project Plan

As Ben Aston (2017) says “Project management is important because it ensures what is being delivered, is right”. Therefore, as part of my rationale, I decided to write up user stories to help me build my project plan (see [Appendix C](#_Appendix_C_–)). User stories are a phenomenal tool which helps define what your product is going to achieve by stating what the user, programmer or even the client might want. One of the advantages of user stories explained by Kamlesh Ravlani (2017) is that “User Stories allow for easy addition and removal of features from the Product.” and with each user story completed the overall value of the product then rises.

From these user stories now created, we can begin defining the tasks and build a work break down structure (see [Appendix A](#_Appendix_A_-)) which displays each task and any sub tasks they may have and how long roughly, they might take to complete and their dependencies if any. Obviously, there are some limitations which I will further explain in my risk analysis however, all tasks should be achievable.

Once each sub task has been defined we can then put all of them into a Gantt Chart (see [Appendix B](#_Appendix_B_–)) were you can clearly see the critical path and how long I have, to do this project and how much slack time I have.

* Risks for critical tasks, with contingencies and mitigation plans includes thorough and timely mitigation steps
  + Risk Analysis
    - Risk
    - Solutions
    - Mitigation Steps

## Source Control

One of the major risks involved when creating any program is losing your work. It can happen very easily and might not be anybody’s fault but if people don’t back up their work it can be lost. However, using source control allows you to have a backup were ever you go, as Kemper (p.75, 2012) explains it “No matter what you’re working on, or what you’re using, you always want to keep some kind of backup”. In conclusion to this, GitHub will be used to fully backup this project furthermore, because of the large file size of the OpenCV library, GitLFS is needed on any computer that tries to pull these files from GitHub.

# References

EmguCV, 2019 [viewed 19/01/2019]. Available at:<http://www.emgu.com/wiki/index.php/Main_Page>

Gale, T., 2008. Normal Distribution [viewed 19/02/2019]. Available at: <https://www.encyclopedia.com/science-and-technology/mathematics/mathematics/normal-distribution#3>

GameCrate, 2016. CES 2016: Tobii Eye Tracking on Assassin's Creed Syndicate [viewed 25/02/2019]. Available at: <https://www.youtube.com/watch?v=6HHvCd_GPek>

OpenCV, 2019. Open Source Computer Vision Library [viewed 19/01/2019]. Available at: <https://opencv.org/>

OpenCV, 2019. Image Blurring (Image Smoothing) [viewed 25/02/2019]. Available at: <https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_filtering/py_filtering.html>

Siegmund, D.O., 2019. Probability theory [viewed 19/02/2019]. Available at: <https://www.britannica.com/science/probability-theory>

SOD, 2019. An Embedded Computer Vision & Machine Learning Library [viewed 19/01/2019]. Available at: <https://sod.pixlab.io/>

Tobii, 2019. Tobii [viewed 24/02/2019]. Available at: <https://www.tobii.com/>

VisionLib, 2019. Augmented Reality Tracking Library for industries [viewed 19/01/2019]. Available at: <https://visionlib.com/>

What-when-how, Unknown. Morphology [viewed 19/02/2019]. Available at: <http://what-when-how.com/introduction-to-video-and-image-processing/morphology-introduction-to-video-and-image-processing-part-2/>

Zhang, S., Turk, M., 2008. Eigenfaces [viewed 26/02/2019]. Available at: <http://www.scholarpedia.org/article/Eigenfaces>

Brems, M., 2017. A One-Stop Shop for Principal Component Analysis [viewed 27/02/2019]. Available at: <https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c>

Aston, B., 2017. Why Is Project Management Important? [viewed 27/02/2019]. Available at: <https://thedigitalprojectmanager.com/why-is-project-management-important/>

Ravlani, K., 2017. 6 Benefits of Good User Stories [viewed 27/02/2019]. Available at: <http://agileforgrowth.com/blog/userstory-benefits/>

Kemper, C., 2012. Foundation Version Control for Web Developers. Publisher: Apress

# Appendix

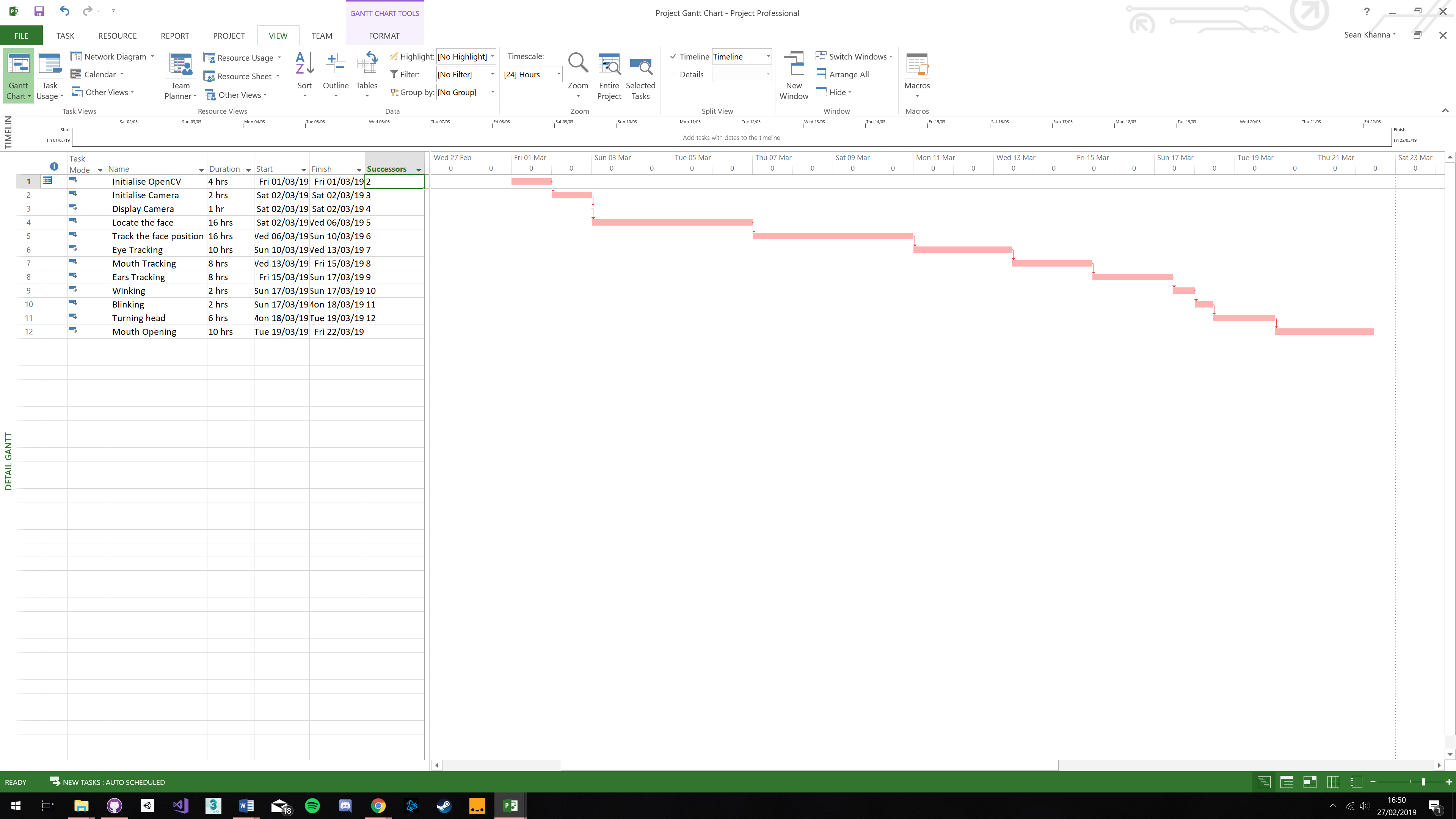
## Appendix A - Timeboxes

Note: All documentation includes evidence of testing

1. Setting up camera including initialisation of OpenCV - 6 hours
   1. Initialise OpenCV - 4 hours
      * Test it works – 3 hours 30 mins
      * Document – 30 mins
   2. Initialise Camera – 2 hours
      * Test – 1 hour 30 mins
      * Document - 30 mins
      * Dependencies: 1.1
2. Display camera – 1 hour
   1. Show camera on screen – 1 hour
      * Test - 45 mins
      * Document – 15 mins
      * Dependencies: 1.2.
3. Getting camera to recognise where the face is – 35 hours
   1. Locate the face – 16 hours
      * Dependencies: 1.2.
   2. Track the face position – 16 hours
      * Dependencies: 3.1.
   3. Testing – 2 hours
   4. Document – 1 hour
4. Getting the camera to recognise where the eyes, mouth and ears are – 26 hours
   1. Eyes - 10 hours
      * Test - 9 hours
      * Document – 1 hour
      * Dependencies: 1.2.
   2. Mouth – 8 hours
      * Test – 7 hours
      * Document – 1 hour
      * Dependencies: 1.2.
   3. Ears – 8 hours
      * Test – 7 hours
      * Document – 1 hour
      * Dependencies: 1.2.
5. Use this to calculate if a player is winking, blinking, turning their head etc – 20 hours
   1. Winking – 2 hours
      * Test – 1 hour 30 mins
      * Document – 30 mins
      * Dependencies: 4.1.
   2. Blinking – 2 hours
      * Test – 1 hour 30 mins
      * Document – 30 mins
      * Dependencies: 4.1.
   3. Turning head - 6 hours
      * Test – 5 hours
      * Document – 1 hour
      * Dependencies: 3.2.
   4. Mouth open – 10 hours
      * Test – 9 hours
      * Document – 1 hour
      * Dependencies: 4.2.

**Grand total:** 88 hours.

## Appendix B – Project Gantt Chart



## Appendix C – User Stories

* As a player I want to be able to turn my head to make my character move because I want to explore.
* As a player I want to be able to use my eyes by winking or blinking to do certain actions because I want to be able to communicate with NPCs.