Using the Peppers Ghost Pyramid technique to create real-time holograms for a charades style game

Final Report for CS39440 Major Project

Author: Elliot Oram (elo9@aber.ac.uk)
Supervisor: Dr. Helen Miles (hem23@aber.ac.uk)

28th February 2017
Version: 1.0 (Draft)

This report was submitted as partial fulfilment of a BSc degree in Computer Science (G401)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, UK

Declaration of originality

I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the regulations on Unacceptable Academic Practice from the University's Academic Quality and Records Office (AQRO) and the relevant sections of the current Student Handbook of the Department of Computer Science.
- In submitting this work I understand and agree to abide by the University's regulations governing these issues.

Name .....

Date .....

Consent to share this work

By including my name below, I hereby agree to this dissertation being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name .....

Date .....

Acknowledgements

I'd like to thank Dr. Helen Miles for her continued support throughout the duration of this project which included advice and guidance on the project context and implementation. In addition I would like to thank the members of the Aberystwyth University computer science department for their assistance in setting up the stall for the prototype demonstration at the 2017 Aberystwyth Science week event.

Abstract

This report describes the process followed to develop a system to create real-time holograms using the Pepper's Ghost pyramid technique and an accompanying charades game. The system was first proposed to be used at Aberystwyth University's Science week in 2018, but is also suitable to be displayed at any appropriate outreach event. The Pepper's Ghost pyramid technique is an excellent tool to use for outreach as it is simple to understand, and in addition creates an impactful display. In order to improve the experience for participants, the charades game is designed to provide a way to interact with the holographic system.

This report will present the purpose, technologies and processes surrounding this system. The development process followed an adapted single person Feature Driven Development (FDD) plan driven methodology which will be discussed in detail below. The report will aim to focus on the software quality and testing performed throughout the development of the system.

Finally, the report will conclude with a critical evaluation of the system itself, the implementation choices and the development processes. This will include, but not be limited to, the programming languages and frameworks used, continuous integration, source control and testing tools, and prototyping and implementation testing.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Background & Objectives

1.1 The scenario

The project is to create a system to produce real time holograms from a camera feed, using the Pepper's Ghost pyramid technique. The system is intended for use at the Aberystwyth Science Week next year, however, there was the potential to display a prototype at the 2017 event held in mid-March. The project is intended to show case a technique, originally used in stage and theatre productions, and give insight into how it can be adapted, with the aid of computer science, to make an impactful visual display.

The system will take real-time data captured from a web cam in the staging area. The staging area will consist of a black background and appropriate lighting to illuminate the subject. The video feed will then be processed and displayed on a large monitor to work with a ghost pyramid of appropriate size.

To make the final demonstration more interactive, the display will have an accompanying system that allows users to play a charades style game. This would display a topic for the user to act out in the staging area and then allow those viewing the hologram to guess the activity being performed. This would require a system capable of taking multiple different string in puts (guesses) from users simultaneously, and feeding back to all users if a guess is correct.

1.2 Motivation and Justification

This project is appealing due to the uniqueness of the technology being used to produce it. Whilst the charades game can be considered as a generic system, the creation of real-time holograms is not heavily developed. Many examples of video footage which can be used with the Pepper's Ghost Pyramid are readily available online, however there are far fewer implementations that do real time manipulation for this purpose.

In addition to the project being interesting, it is designed with outreach events in mind therefore giving it value as a potential teaching aide. Pepper's Ghost is an excellent example for children to learn how basic physics and computer science can be used to create an interesting and engaging visual display. In addition, the charades game and real-time hologram system will help to further engage the audience with technique and visualise an impactful application of the technique.

1.3 Background

1.3.1 Pepper's Ghost Pyramid

The Pepper's Ghost technique was originally used for stage and theatre productions in the Victorian era to display holographic illusions to the audience. The technique, discovered by Dr. Henry Pepper, was first used in theatre in the 1860s [1] and was used primarily to create a ghost-like illusion.

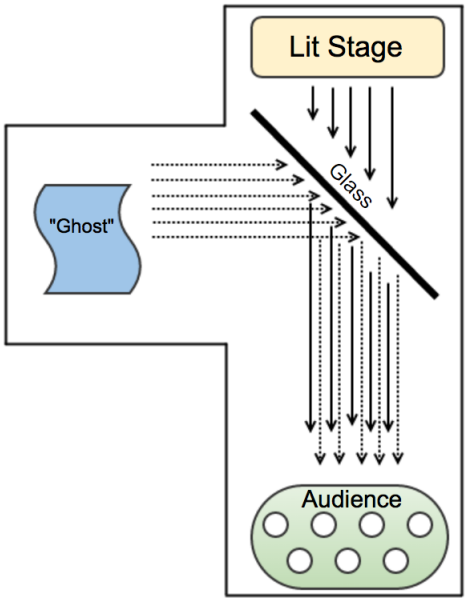

Figure 1.1: A basic diagram to display the Pepper's Ghost technique's use in theatre. In the diagram solid lines represent light from the lit stage and dashed lines as light from the "Ghost". The image has been taken from Brianne Costa's Blog on the website Comsol [1].

Figure 1.1 shows a basic example of how the technique produces holographic illusions. The lit stage that is parallel to the viewers line of sight will have actors and scenery as per a normal stage production. The "Ghost" is located off stage and is not visible by the audience. When illuminated, the ghost is displayed to the audience as the light rays travel radially from the "Ghost" and those that reach the glass will be refracted. The glass is positioned at 45° from both the audience and the "Ghost" to account for the 90° offset between the two.

This technique has been modernised since the 1860s, and has been used in different settings proving particularly popular in museums and exhibitions such as *An Audience with Sir Alex Ferguson* at the Manchester United Football Club museum [2] and *Shane Warne - 'Cricket Found Me'* at the national sports museum in Melbourne [3].

Within the last few years, there has been a resurgence in the use of the technique with consumers creating there own Pepper's Ghost illusions on mobile devices. Whilst the technique still uses Henry Pepper's original concept, it has been modified to display the hologram from multiple angles using a pyramid - rather than a single glass plane. This structure is commonly referred to as the Pepper's Ghost pyramid.

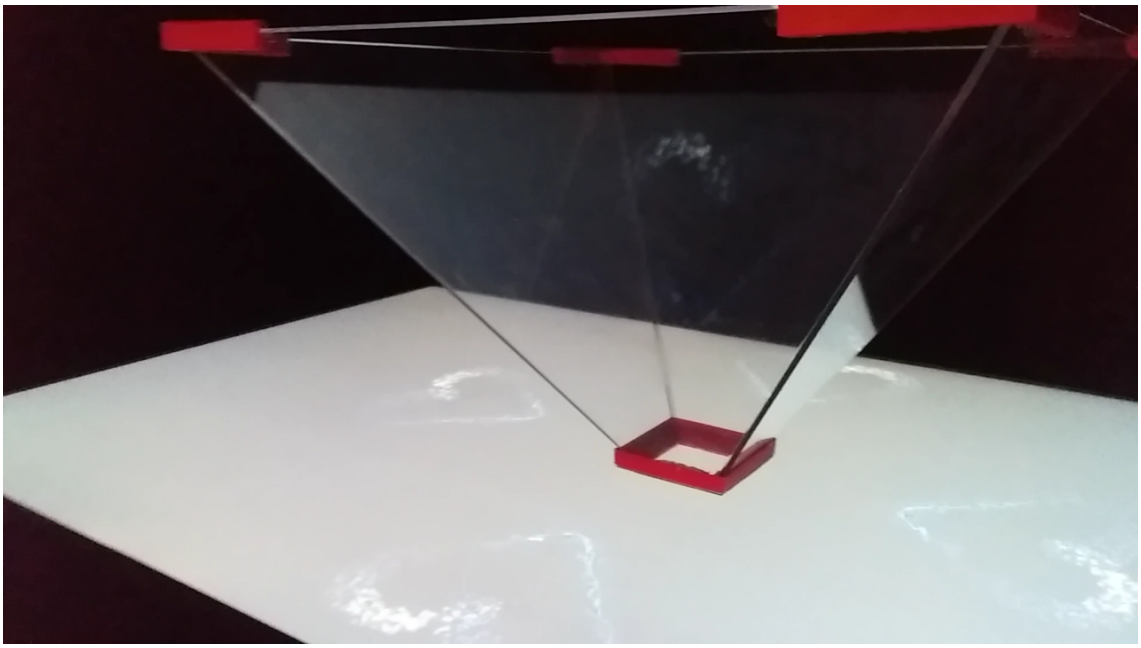

Figure 1.2: An image of modern day Pepper's Ghost Pyramid made from clear acrylic owned by the Aberystwyth University Computer Science Department.

Figure 1.2 shows an example of a Pepper's Ghost pyramid. The pyramid is square based, made from a transparent material (such as perspex, clear acrylic or glass) and is open at both the top and bottom. The technique for displaying holograms differs with the "Ghost" now being an image or video displayed on a screen. The pyramid is located on a large horizontal monitor that is being used to display the hologram input images. Furthermore, to create an illusion from all sides of the pyramid, four images are required (one for each side of the pyramid). This means that the 2D projection of the image can be seen from any side of the pyramid. This Pyramid structure will be used for the hologram creation part of this project.

1.4 Analysis

Whilst there have been numerous examples of applications using the Pepper's Ghost Pyramid as a computer visualisation technique, very few implementation use real time video feeds. This year, 2017, in the presidential election campaign in France, candidate Jean-Luc Melenchon has utilised the Pepper's Ghost technique to give a speech to prospective voters in six cities simultaneously [4]. Whilst no precise details of how this system works have been released, this project draws several parallels with the system. Both use the Pepper's Ghost technique with real-time video and therefore this example can be used as a feasibility case for this project. Furthermore, this highlights additional use cases in media and visual representation proving the technique to be of more use than simply an educational tool.

Whilst many have used the technique in an educational setting (such as museums and exhibitions), it is more often the case that the content of the hologram is used as an educational aid, rather than focusing on the technique itself. Due to its simplicity and visual impact, the Pepper's Ghost technique is ideal to help younger children understand light and refraction.

1.4.1 Objectives

Initially, the project scope covered the creation of a system to produce real-time holograms using a video feed. However, considering the time-line of the project and the work involved, the addition of an accompanying system for the Charades game was added to the objectives. The objectives that were agreed upon are as follows:

- To produce a system that can be used for outreach events to show case the creation of holograms using the Pepper's Ghost Pyramid.
- The system should use real-time video to capture a member of the audience and then project these images as holograms using the Pepper's Ghost Technique.
- An accompanying system should be produced to allow for a charades style game to be played using the hologram system to display the actor to the viewers.
- Given the target audience, the system should be interactive and engaging whilst also enabling the viewers to learn about the technique and how it works.
- The system should be easy to use and for the most part self explanatory.
- As it is designed for use at outreach events, the system should be robust and thoroughly tested.
- The system should be easy to run from the perspective of the client. Initial set up should ideally be straight forward and self explanatory.
- Given the target audience, the system should be robust and difficult to crash.

1.4.2 Problem decomposition

Following the agreement of the project objectives, it was possible to begin to decompose the requirements of the project into smaller sections. As the project is comprised of two systems (the real-time Pepper's Ghost creation system and the charades game), this seemed a natural divide for the project at the highest abstraction layer.

1.4.2.1 Real-time Pepper's Ghost

Following research into the Pepper's Ghost technique, it was decided that it would be feasible to construct a prototype of the hologram system for the 2017 Aberystwyth University Science Week event. The main tasks to produce the prototype system were:

- Analyse and produce a design for the display area (where the pyramid and monitor are used to produce a hologram).
- Analyse and produce a design for the staging area (where the actor is filmed).
- Design and implement a program capable of obtaining a live video feed from a web cam.
- Correctly orientate and position multiple copies of the video feed in a single display window.

- Scale the video feed is of the correct size for the output device.

The main consideration throughout the development process of the Pepper's Ghost application is the efficiency of the video feed manipulation. To have an effective display, the video should be running at no less than 30 frames per second. An ideal target would be 60 frames a second but this may not be achievable depending on the capture speed of the camera.

1.4.2.2 Charades game

The charades game forms the more interactive side of the project. The concept is to have a single actor (member of the audience) outside the viewing area with a mobile device. He or she selects a phrase from a choice of phrases and then act out the selected phrase. The acting is captured by the web cam from Pepper's Ghost system and displayed as a hologram in the viewing area. Whilst the phrase is being acted, the viewers (who will also have mobile devices) are given information about the phrase (such as the number of words and the genre of the phrase) and attempt to guess what is being acted. Once the phrase has been successfully acted, the actor is be prompted to select a new phrase and the viewers are allocated points. After three phrases, the winner is the viewer with the most points. The main tasks of the charades game are:

- Create a user interface for both the actor and viewer.
- Allow the actor to select a phrase and, if the phrase is comprised of multiple words, a current word to act.
- Allow the viewer to attempt to guess the current word or phrase.
- Inform both users when the word or phrase has been correctly guessed.
- Prompt the actor for a new word or phrase for them to act.
- Establish and implement how and when the game ends.
- Inform users of the winner.
- Create a session system to only allow those who have logged on to the website with the correct details to access the information.

Whilst a mobile application seems ideal for use at localised events, a web app implementation would allow users to access the system from their own mobile devices regardless of platform. However, using a web app raises additional security issues that should be taken into consideration. To stop malicious interactions from those outside the event, the web app must ensure that only those at the event can access the system. To ensure this, the main task list included adding a session key to the system. This session key will be distributed at the event, and only those who know the session key are able to access the system. The system also provides the ability to change the session key easily when required.

1.4.3 Alternative implementations

Implementing an android application over a web app boasts the increased security as devices could potentially connect via Bluetooth or a peer-to-peer connection. The shorter connection range offered by these technologies, would mean that those outside the event would not be able to join session and corrupt the system. Furthermore, it would stop users from attempting to change the information in the URL for each page and therefore reduce the number of checks required to ensure that the system data has not been tainted.

Despite this, a web application done correctly is more accessible to users as it is platform independent. In addition, it would be more likely that users can access the system from their own devices, such as smart phones, meaning more users can play the game at the same time.

1.5 Process

After consideration of different methodologies, the decision was made that this project would follow the Feature Driven Development (FDD) plan driven methodology. FDD is normally considered for larger projects as it provides a framework for distributed development. By dividing developers into smaller teams, FDD allows those teams to tackle features one at a time in parallel. Furthermore, the up front planning stage is generally more suited to projects that are more stable as, whilst it can be adapted throughout the process, the overall model is normally only added to, and the core architecture remains static. FDD follows five steps throughout the development of a software system. The first three steps regarding planning and designing an overall model, and developing a list of all the features that must be considered for the system to meet the criteria of the end users. The remaining two step form an iterative process of designing and implementing features one by one.

The steps required to complete this project are well defined and therefore would be well suited to having an up front design. Furthermore, FDD encourages continuous integration (CI) which offered a way to produce a functional prototype at various stages of the project. CI was a significant aid for producing a function prototype for both the mid project review and the 2017 Aberystwyth University science week event. FDDs **Design by feature** stage offers an ideal step to perform spike work for technologies that developers are unfamiliar with. Moreover, it was still possible to follow the five step process of FDD development in this project.

1.5.1 Single Person FDD adaptation

To successfully use FDD for this project, several adaptations to the normal processes of the methodology were made. Many of these changes mirror those discussed in S. Khramthchenko's thesis, *"A Project Management Application for Feature Driven Development (FDDPMA)"* [5]. The most notable being the abolition of developer teams in favour of a single developer. This required the developer to assume multiple roles throughout the development process and at each different stage as specified below.

1.5.1.1 Develop an Overall Model

Initially, a Domain Expert (Customer) was required to aid in the development of the overall model and feature creation. In this context, the project supervisor (Dr. Helen Miles) fulfilled this role and the developer acted as both the Chief Programmer and Chief Architect. For this project, the Domain specific language was shared by both the Domain Expert and the developer.

1.5.1.2 Build a Feature list

The feature list was produced with the objectives of the system and the main tasks for the project in mind. This was finalised with a discussion between the developer and the Domain Expert which verified all the features. In addition to a description, each feature also had a complexity and priority rating, as well as a list of dependent features and whether the feature was require for the prototype. The final feature list can be found online here:
https://github.com/ElliotAOram/GhostPyramid/blob/master/Features/FeatureList.md .

1.5.1.3 Plan by Feature

Steps such as establishing developer teams and scheduling developer teams' time throughout the project were no longer required. In their place, the features were given priorities to aid time scheduling and development order. Once the order was created, the features were assigned to separate week long iterations.

1.5.1.4 Design by Feature

Features were selected in dependency order. Once selected, features were exhaustively designed taking into consideration the functions required to fulfil the feature as well as how these functions should be tested. The project used GitHub for version control and issues were created which corresponded to a feature. The first action in an issue was to complete a design of the feature and update the overall design as required.

1.5.1.5 Implement by Feature

Features were implemented in the same GitHub issue as the design work. The project was developed using Test Driven Development (TDD) and, therefore, the test suite was updated before any code was implemented. Once the tests were created, an implementation was added which had to pass the tests to be acceptable. Whilst the tests could be run locally, there was also a Jenkins service for continuous integration to run the full test suite before it was merged with the master branch.

Chapter 2

Design

As well as describing the design adopted it must also explain what other designs were considered and why they were rejected.
The design should describe what you expected to do, and might also explain areas that you had to revise after some investigation.

The design for the system was completed in two major steps. First, the initial design was created before any development and following this, incremental changes were made to add more detail to the design with every relevant feature. The initial design formed the overall model for the system, and was only added to over the duration of the project. The UML class diagrams were designed to be basic initially and only contained major functions and classes. These then evolved as the features were developed.

At the plan by feature level, design was performed at the start of each feature. The feature would be analysed and the relevant functions and variables that were required would be added to the design first. Following this, unit (or system) tests would be designed to exercise the functions as well as the full requirements of the feature.

2.1 Informing the Overall model

2.1.1 Use case

To best identify the requirements of the users, the use case diagram (Figure 2.1) was produced. In addition to the functionality required by each type of user (Actors and Viewers) the diagram also divides the system into basic subsections. This diagram was a good first step in the early development of the system as it gave an overview of the system as a whole before any major implementation choices were required. Whilst very basic, it also showed interaction with the system that the users would require and acted as the base for the input and output requirements to be address in the user interface design. In addition, the use case diagram showed that the Charades game functionality was dependant on the type of user interacting with the system, which promoted an model-view-controller design pattern to be used.

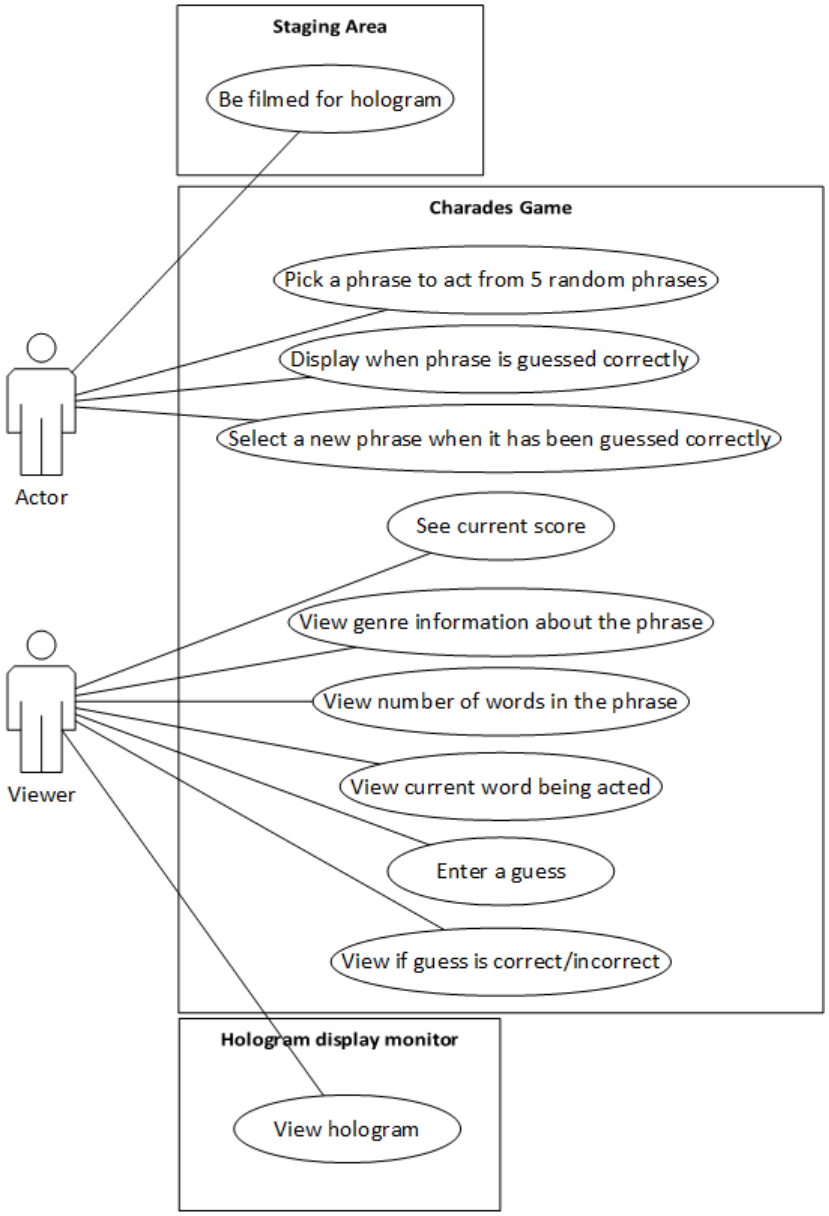

Figure 2.1: A use case diagram produced to show how the users will interact with the system and the functionality they require.

2.1.2 Activity diagram

Given the system was deigned to be a game of charades, it follows a linear set of instructions that mirror the original rules of the charades game. An activity diagram was chosen to model the flow of activities in the system due to the well defined path of game play. Figure 2.2 shows an activity diagram that maps the basic flow of a single round of the game. The *Image processing loop*, shown in purple, describes the flow of the hologram creation. This process (prefixed with the letter A) run continuously in parallel with the game loop (prefixed with the letter B).

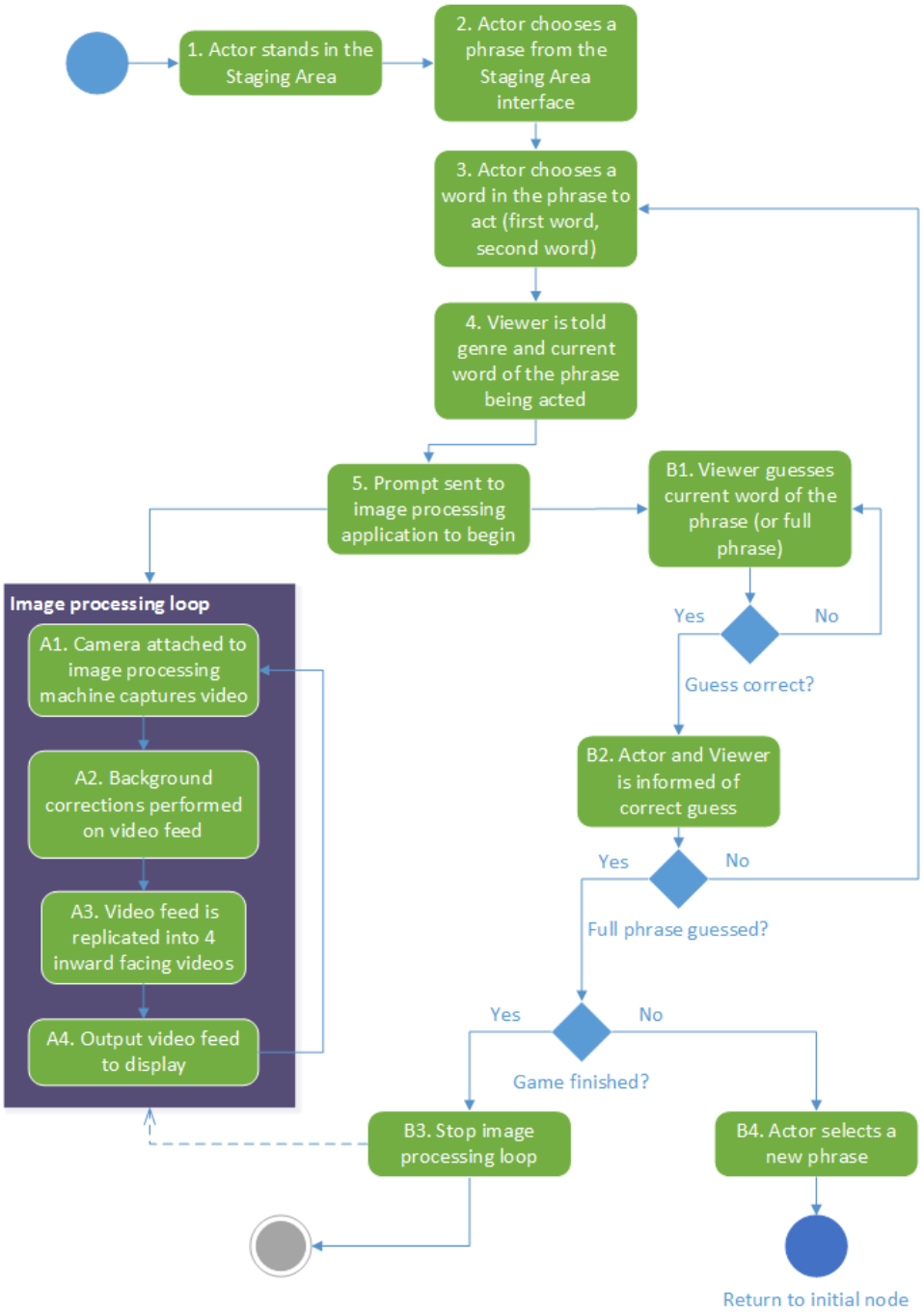

Figure 2.2: An activity diagram that describes the flow of information in the system. Includes the data flow for both the Hologram and Charades systems.

2.1.3 Component diagram

The system, in its totality, had to include multiple pieces of hardware to operate. The component diagram, shown in Figure 2.3, helped to combine the different hardware considerations in tandem with the software. Furthermore, the diagram confirmed the presence of three separate areas of the project.

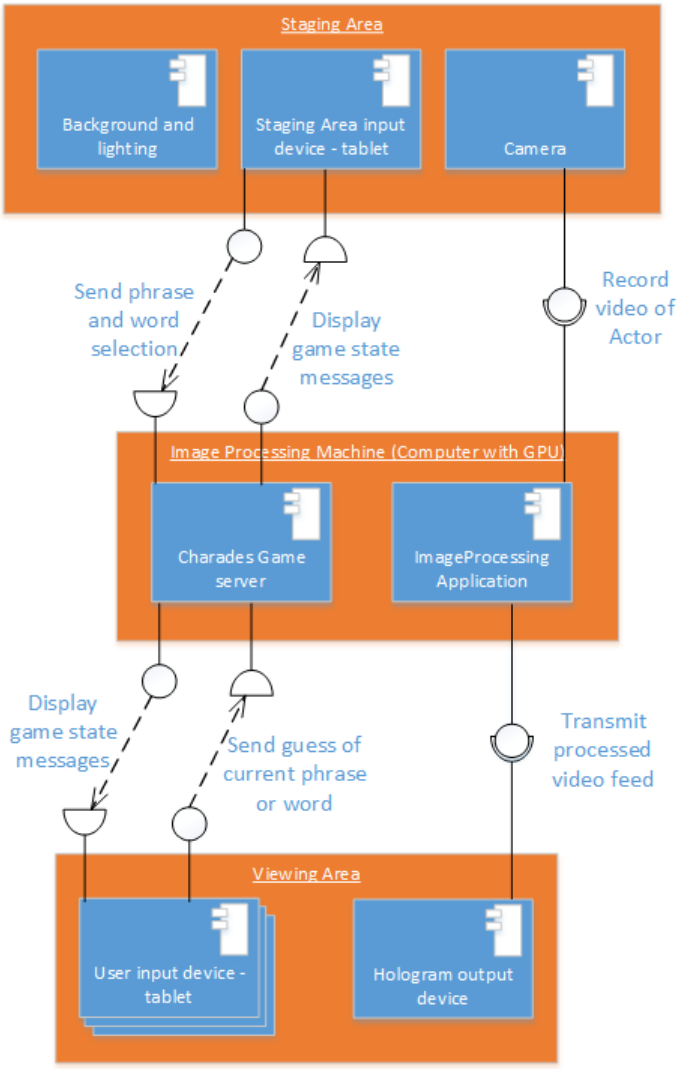

Figure 2.3: A component diagram that describes both the software are hardware components of the system and how they interact with each other.

The **Staging Area** would be where the actor would be filmed. The camera directly connects to the ImageProcessing machine via a usb cable for data transfer. The Staging Area input device will use a wireless connection to send and receive messages from the Charades game server that is hosted on the image processing machine.

The **Image Processing machine** is a computer that acts as the server for the website as well as running the image processing application for the holograms. Due to the nature of the video feed processing in real-time, the computer will require a Graphical processing unit (GPU). The Image Processing application has input of the raw video feed from the camera in the Staging Area and will process the video and output the result. The Charades Game server is hosted on the Image processing machine. This component will handle the flow between the users in the Viewing Area and the Actor in the Staging Area.

2.2 Overall Architecture

2.2.1 Hologram creation system

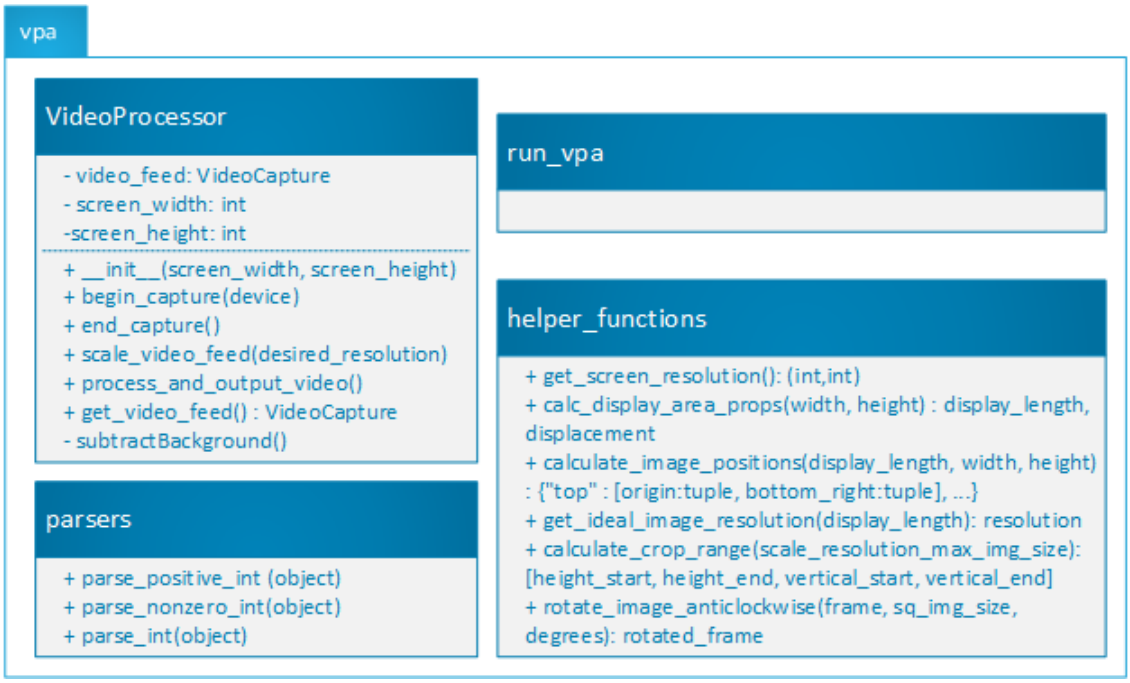

Figure 2.4: The UML class diagram for the hologram creation system. The diagram shows the vpa (video processing application) module and the enclosed subclasses and submodules.

- **vpa**: This module holds all the code for hologram creation system. vpa stands for *Video Processing Application* as the code in this module handles the processing of the video feed.
- **run_vpa**: This is a script and is the most simplistic in the module. This is the driver script for the VideoProcessor class and runs the required functions to create begin the video feed capture and display the manipulated video feed in the output window.
- **parsers**: This module holds several functions that are used to ensure that values provided in the system are expected and correct. As Python is not a strongly typed language, the parsers are required to ensure that no type errors occur during execution of the application. The parser functions have been separated into a separate module to group them together. All functions in the parser module have input of a single object (where object is the super class of all Python variables) and will raise a Python ValueError if they do not meet the criteria of the function.
- **VideoProcessor**: This class handles the main functionality and logic of the application. The most import function in the VideoProcessor is the process_and_output_video function. This function contains the main processing loop of the application. This loop will call the scale, rotate and position functions from the helper_functions module. The VideoProcessor has been designed as a class rather than a collection of functions so that the video_feed variable (which contains the video feed from the web cam) can be stored.

- **helper_functions**: This module was not present in the original design, but in order to improve the quality, readability and test coverage of the code, it was added to hold refactored code from the **VideoProcessor** class. In addition to simple helper functions such as obtaining the screen resolution, the helper_functions module also aid with manipulations of the video feed. Functions used for calculating the location, cropping and rotation of the image are also in the helper_functions module.

2.2.2 Charades Game

2.2.2.1 Model Design

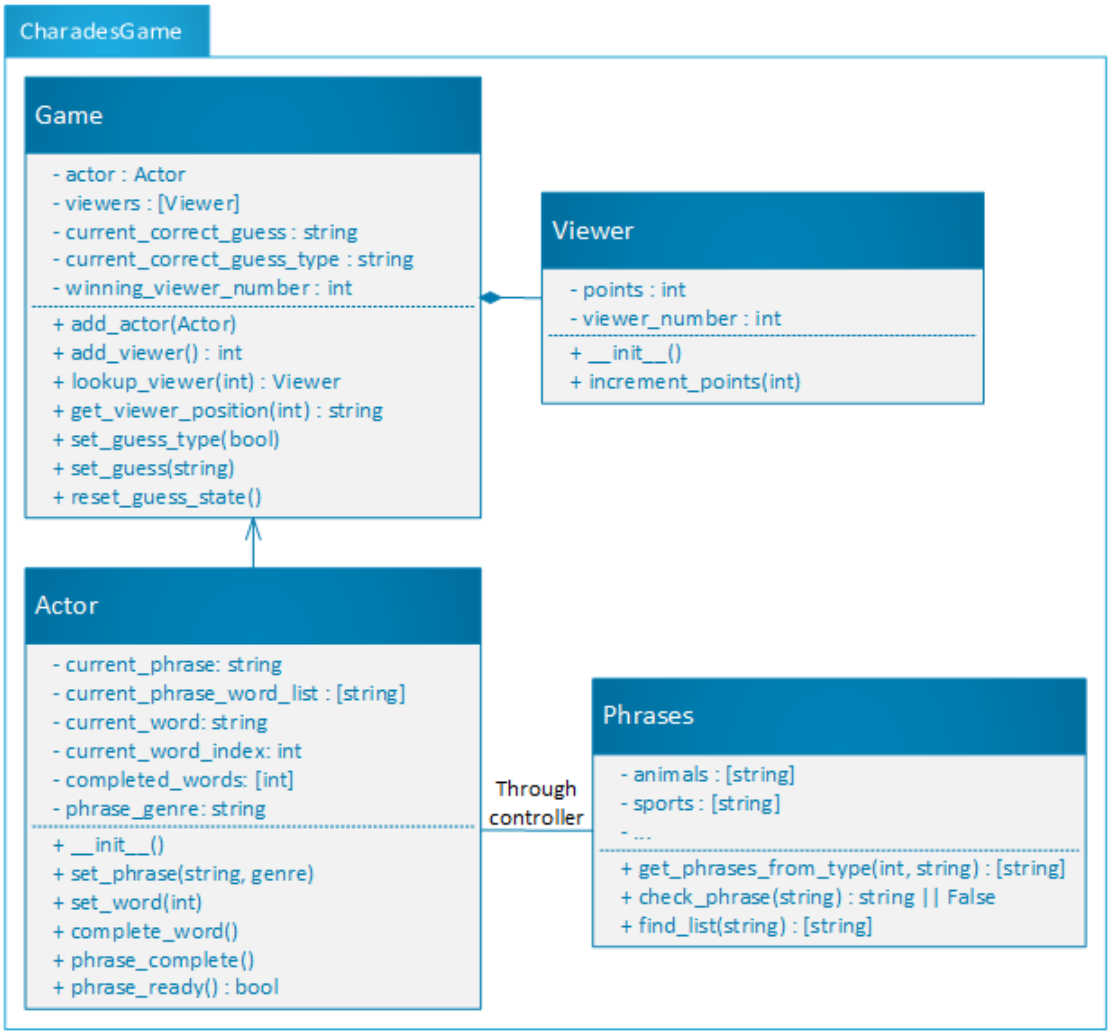

Figure 2.5: The UML class diagram for the Charades game. The diagram shows the Model for the charades game implementation.

Figure 2.5 shows the design for the Model used to represent data in the Charades game. For the Model, it was decided not to use a data and instead use a class representation for data storage. The reason behind this, is the flow of the charades game is dependant on the state of the system, and a class based model provides a better way to model state versus a database alternative.

The Model has been separated into four files:

- **Game**: The Game class holds the state of the game at it's highest level of abstraction. As well as holding a list of the Viewers and Actors, the Game will also hold information regarding the current rounds correct guess, the type of the guess and the winning viewer number. All of these values are used to update the game state and inform the users of the current state of the game via the APIs.
- **Viewer**: The Viewer is a simple class that holds the number of points accumulated by a viewer and the the viewer_number of that viewer. When a viewer logs in, the viewer_number is automatically generated and a Viewer object is generated and added to the Game object. In addition to the Viewer being created in the model, the viewer_number is also stored client side in a session cookie. This viewer_number is then queried when the client needs to display a viewers score.
- **Actor**: The model uses the concept of phrases and words for topics that an Actor will select to act out. In this scenario, a phrase is the whole topic that is being acted, for example "Shot put". However, a word is part of a phrase, for example "Shot" is a word in the phrase "Shot put". The Actor class holds the current phrase and word variables for the phrase that is being acted. In addition it has functions to set new words and phrases and complete words when they are guessed correctly.
- **Phrases**: The Phrases module holds all the phrases that can be acted by the Actor in Python dictionaries. In addition, the module contains several helper functions designed to add checking phrases and phrase acquisition. This part of the model could have been improved by using a database representation. This data, unlike that of the other classes in the model remains static and could therefore be more suited to using a simple database to hold the information. However, a possible caveat for this is the addition of another technology to the model. Whilst this technology would be simple and is well supported, it would add some additional complexity compared to using the same method as the class representation.

2.2.2.2 Routing and Website Design

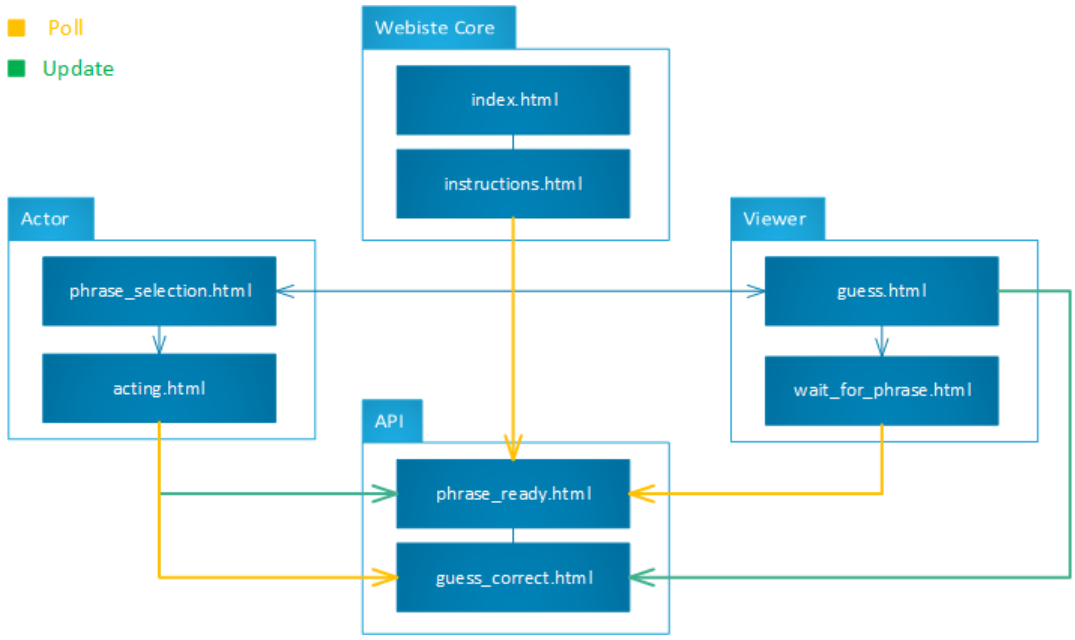

Figure 2.6: The website design for the Charades game. The diagram shows the Template structure of the website as well as when pages are updated from the API and when polling of the API takes place.

Figure 2.6 shows the design for the templates (html pages) of the website. The web pages are grouped into relevant sections that relate to their functionality:

- **Website Core**: The Website core includes the login (index) and instructions page. The index.html page is where the user will provide the session information (This would be provided on the day that the system is being used, or set up in advance for an event) and the option to login in as a Viewer or Actor. The instructions.html provides a dynamic set of instructions depending on the type of the user (Actor or Viewer) and a continue button for the Actor to continue to the phrase_selection page. In addition, as a viewer, the page will poll the phrase_ready API and, when true, display a button for Viewers to click to advance to the next section of the website.
- **Actor**: The Actor section of the website handles phrase and word selection as well as a default page for the actor to view while they act out the current phrase. phrase_selection.html provides the actor with a selection of five phrases and instructs them to select one. Once chosen, the actor advances to the acting.html page where the phrase is displayed and the actor is asked to choose a word from the phrase if that phrase has multiple words. Once the phrase and word have been selected, the phrase_ready API will be updated to display True.
- **Viewer**: The Viewer section of the website handles when the phrase or word is guessed correctly or incorrectly. guess.html is where the viewers attempt to guess the current word or phrase being acted. Information about the genre, number of words and current word is also displayed to the viewer on this page, as well as a text field for viewer to submit their guess (see Figure 2.7). This page both polls and updates the guess_correct API. When a correct

guess has been made, the API is updated with either ”Word” or ”Phrase” depending on the guess type. As multiple users will be on this page, the page also polls the guess_correct API to see if a correct guess has been made. When a correct guess has been made, both the viewer who guessed correctly and all other viewers are taken to the wait_for_phrase page. This page polls the phrase_ready API and when a new phrase has been selected (API returns ’True’) viewers are redirect back to the guess.html page to guess the next word or phrase.

- **API**: The API consists of simple pages that are used when polling the game state for information. phrase_ready.html displays ’True’ when the phrase is in a guessable state (the phrase and word have been selected and the phrase or word have not been correctly guessed) and ’False’ when not guessable. guess_correct.html displays ’None’ when no correct guesses have been made by the viewers, ’Word’ when the word has been guessed correctly and ’Phrase’ when the phrase has been correctly guessed.

2.2.3 UI Design

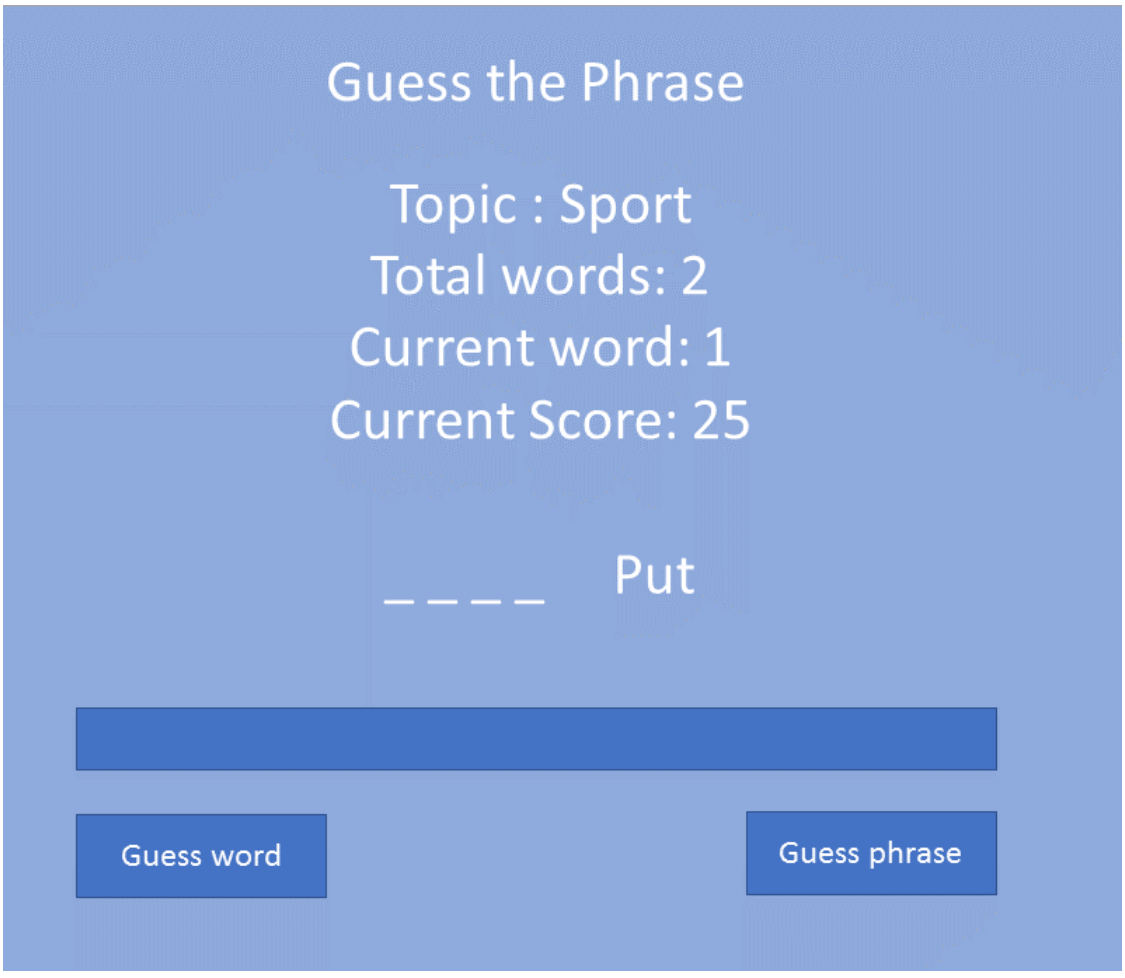

Figure 2.7: The UI design for the guess.html page. This page shows a user with 25 points and the phrase ”Shot put” is being acted. The word ”Put” has already been guessed meaning the word ”Shot” is represented by ”_ _ _ _” as it has yet to be guessed.

Figure 2.7 shows the UI design for the viewers guess.html page (All other UI diagrams are available in Appendix ...). The purpose of the UI diagrams was to ensure that all the functionality required for user interaction on a given page was met. In the case of Figure 2.7, the page had to do the following:

- Display the Genre information
- Display the total number of words in the phrase
- Display the current word being acted (when phrases contain multiple words)
- Display the viewers current score
- Provide a way for the viewers to submit a guess for the current word or the full phrase.

Whilst this criteria was never formalised at the point of design, given the knowledge of the system it was possible to infer this from the previous design documents (Figure 2.6, 2.5). The goal for all pages on the website was to keep them simple were possible. Given the age of the target audience, the focus of the interface was on ease of use and simplicity. These values have been reflected by the minimal buttons and small amount of text.

2.3 Implementation tools

2.3.1 Python

Python is a free, easy to learn and easy to read programming language. As a scripting language, it offers lightweight solutions to software problems without requiring long compile times. Python, whilst not a fully object orientated (OO) language, provides OO support and this approach is encouraged by the community for larger projects. For this project, Python version 2.7.11 was used. Whilst this is becoming a legacy version of Python, it is still well supported and a pre-compiled OpenCV binary file for this version of Python is available from the OpenCV website. Furthermore, the developer had more experience with the 2.7.11 Python language than the newer 3.6 variant meaning that less training was required for the project to get under way.

2.3.2 Pycharm

Pycharm is a Python Integrated development environment (IDE) used to aid with the development of programming projects written in Python. The application offers facilities for refactoring code, auto completion and creation of variable and function names, as well as on the fly testing and code execution. In addition, Pycharm supports a variety of community built plugins that help developers with style and following convention. For this project, the PEP8 style plugin was used to reduce the amount of errors and warnings raised in the Pylint linting tool (discussed in further detail in section 4.2.2.1).

2.3.3 OpenCV

OpenCV is an open source software library that provides solutions to common computer vision problems. It is an ideal library for handling image manipulation and offers easy solutions to image

display and rendering. OpenCV comes with precompiled binary interfaces for C++, Python, Java and MATLAB and is supported on Windows, Linux, Android and Mac OS [6]. Whilst OpenCVs application range from object and feature detection to real-time 3D model extraction, this library is only being used for the simpler case video feed capture and basic manipulation.

2.3.4 Django

Django is a free and open source web framework for the Python programming language. The framework helps developers to build web applications quickly without requiring in depth knowledge of web concepts such as middle-ware management and page routing. The framework uses a Model-Template-View (MTV) design pattern which almost mirrors the Model-View-Controller (MVC) pattern more commonly recognised in software engineering.

- Model: The Model manages the data and logic of the application. In this project, the model consists of Python classes representing the Actor, Viewer and Game.
- Template: Template is comparable to the View in the more standard MVC pattern. They display the data to the client (webpage) and are written in the HTML web language. In addition, data can be embedded into the templates using pythonic operations and variables that can be passed to the template from the View.
- View: The View is comparable to the MVC controller. This handles the communication between the Model and the template as it has access to both. The View pushes data to the template using a python dictionary and pulls data from the Model using either database querying or accessing model classes with functions.

2.3.5 Git and GitHub

To maintain version control the project used git and GitHub. Git was used to help organise code as well as being a method to back up the development that had been carried out. With regards to organisation, this refers to setting milestones to measure progress, creating stable release branches for storing working versions of the code and creating pull requests to allow features to be reviewed before merging. In addition, a default template for GitHub issues and pull requests was created which added check lists to the issue and pull request descriptions. The templates helped to ensure that all required elements of the feature were considered and checked.

When developing with git, a git work flow was used to ensure that iterations and features were handled in git correctly. Figure 2.8 shows the work flow diagram followed during development. In addition to the rules written at each stage, some additional rules were applied at certain stages:

- **A1**: The local branch follows the naming convention of:

  `<issue_number>_feature_<feature_number>`

  An example of this for issue number 60, feature number 14 would be:

  `60_feature_14`

- **A2, A3**: When code is pushed, it should reference the issue number so that the issue online shows the commits history. This is done by finishing the commit with:

  `Refs #<issue_number>`

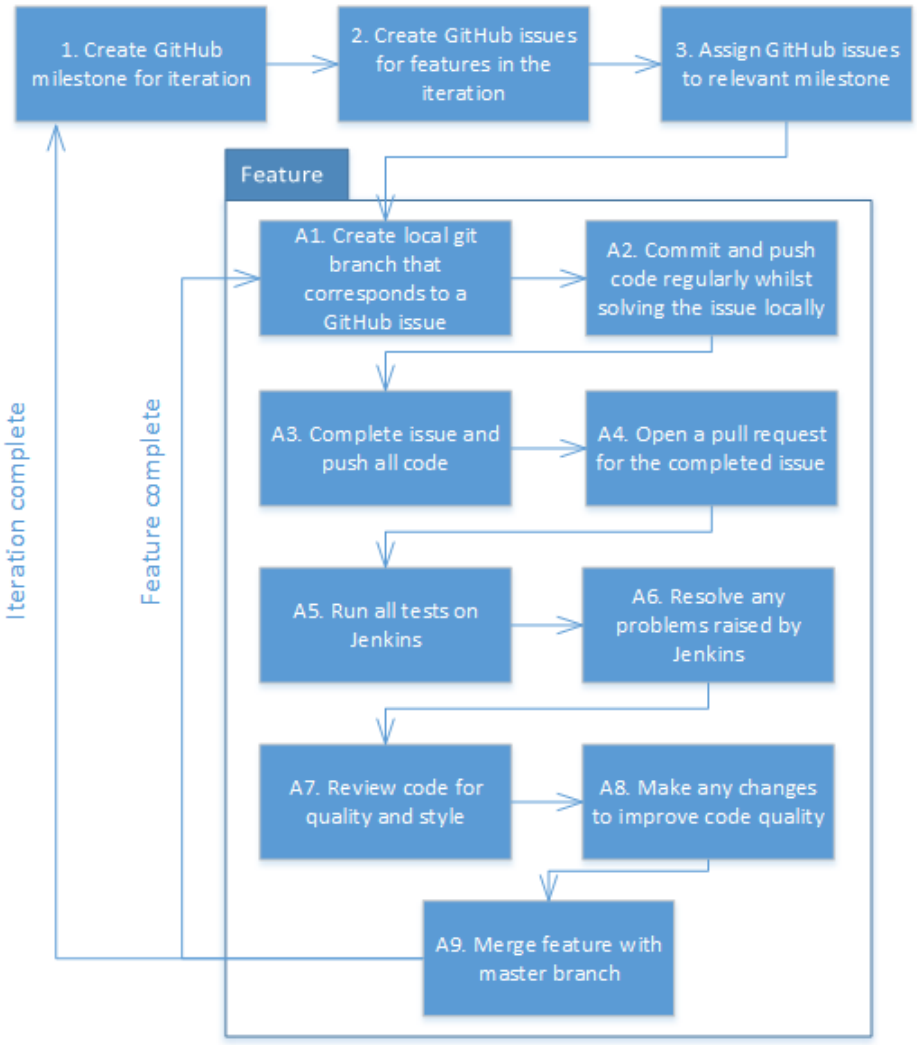

Figure 2.8: The GitHub work flow diagram used for development. The outer processes (1-3) are performed for every iteration. The inner processes (A1-A9) are performed for every feature.

Chapter 3

Implementation

The implementation should look at any issues you encountered as you tried to implement your design. During the work, you might have found that elements of your design were unnecessary or overly complex; perhaps third party libraries were available that simplified some of the functions that you intended to implement. If things were easier in some areas, then how did you adapt your project to take account of your findings?

It is more likely that things were more complex than you first thought. In particular, were there any problems or difficulties that you found during implementation that you had to address? Did such problems simply delay you or were they more significant?

You can conclude this section by reviewing the end of the implementation stage against the planned requirements.

Chapter 4

Testing

The project requirements state *"Given the target audience, the system should be robust and difficult to crash."* In order to achieve this, testing should be a priority to ensure a functional system. Furthermore, the testing of this project (as proven below) is comprehensive and attempts to cover all eventualities. With a combination of automated, manual and user testing, a reasonable degree of confidence can be stated with regards to the robustness of both the hologram and charades systems.

Whilst data security did not need to be considered for this project (as no sensitive data is stored) security against cyber attacks and malicious use have been considered throughout the testing and development process. For the most part, manual testing has been used to uncover potential flaws in the security of the website and, when discovered, additional automated tests have been added to safeguard against these issues in the future.

4.1 Overall Approach to Testing

The project used TDD as part of the process for software implementation. As well as designing features, tests were also designed for the required functions before any implementation. There were two main types of tests for the website: unit tests, which test the functionality of the model and controllers, and system tests, which use Selenium to check the flow of the system and simulate the actions of a user. Where appropriate, tests for functions were designed to test three different cases:

- Success case: Simulates either an expected input parameter for the function or an expected state that the system should be in to run a function.
- Edge case: Checks the maximum range of the functions input parameters. For example, if the function was only designed to allow 5 viewers to participate in a game, then the test case would check that the when the viewers total 5 the function still passes.
- Failure case: Ensures that given incorrect parameters or system state, the function will fail in the expected way.

An example function and corresponding test cases displaying the above approach to testing can be found in Appendix C, section (...).

Unit tests were organised into classes where each test class would exercise a single class or module of code. Similarly, system tests were designed to test a single Django template. All test classes are held in test files prefixed with *test_* and these can be found in the *python/tests* and *charades/charades/test* directories. The requirement for adding code to the master branch is that all unit and system tests (past and present) must pass, and the static analysis tools must produce no errors.

4.2 Automated Testing

4.2.1 Jenkins

To improve the consistency of the code base, the GitHub repository was linked to a local Jenkins server. This server was set up at the beginning of the project and was used as part of the development life cycle of each feature. The Jenkins server ran as a background process on the development machine. The set up for the Jenkins server followed two guides: The official Jenkins Windows Service installation guide *"Installing Jenkins as a Windows Service"* [7] from the Jenkins.io wiki and Steve's Blog *"Automated python unit testing, code coverage and code quality analysis with Jenkins"* [8], a self hosted technical blog from a software development consultant specialising in Python.

Whilst Jenkins offers the ability to run tests periodically, this feature was not used and tests were triggered manually by the developer. The reasoning behind this is that time based builds were not required. Had the project been developed by a team, the need for timed builds is more justifiable as many developers will be pushing code changes simultaneously. In a single person developer team, code is pushed one feature at a time and the work flow is linear, hence there is no justifiable requirement for periodic builds. Despite this, the use of Jenkins is still valid in a single developer project, as it allows for centralised building of the project in a controlled, repeatable, environment. Additionally, the formatting of results produced by Jenkins offers an easy way to quickly identify problems within the code base.

4.2.2 Static analysis

4.2.2.1 Pylint

Pylint is a linting tool that follows the PEP8 standards [9] for Python coding. The linter runs static analysis on the code base and ensures that all code meets with the standards specified by PEP8 (and produces warning and errors were this is not the case). The PEP8 standards include guidelines for style and syntax, as well as documentation (in the form of doc strings). Whilst several alternative standards for the Python coding language are available, PEP8 was chosen as it is promoted by the developers of the Python language and is the most popular amongst the community. An example of the Pylint output rendered by Jenkins is displayed in Figure 4.1. In this Figure, the peaks correspond to when pull requests are initially pushed to Jenkins and troughs are were the Pylint warnings have been resolved.

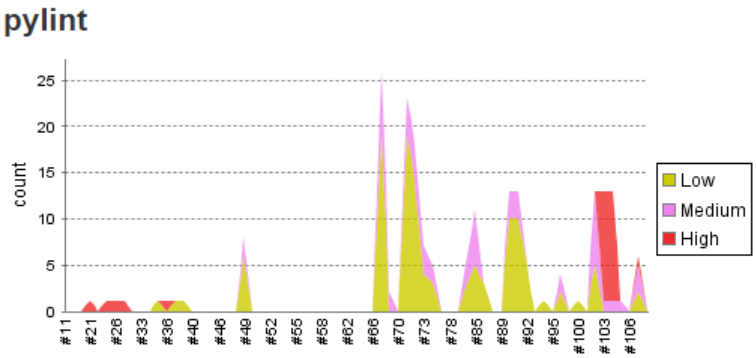

Figure 4.1: The output of the Pylint test run on the full code base rendered by Jenkins. Here, the number of warnings, and their severity, is logged by each build number.

4.2.2.2 Test coverage

An additional testing package, nosetest [10], was used to run Python tests for this project. This package produces reports that are interpretable by Jenkins for both unit tests and test coverage. The test coverage displays graphically the proportion of the code that is run in tests. Although basic, this helps to highlight potential conditional statements, function and even lines that are not being tested. As the web based charades game was written in Django (which has its own testing framework), nosetest could only be used for the hologram creation software.

An example of the test coverage output rendered by Jenkins is displayed below in Figure 4.2. Figure 4.2 shows that whilst the majority of the code base is covered by testing, several lines are excluded. These lines had to be omitted from tests as their functionality was to perform the video processing loop. Whilst the functions that are called within the loop are sufficiently tested, the loop itself can not be tested easily as it creates an output window. The output window is only successfully closed via mouse click on the window close button (an operation that is not easily replicated within unit tests).

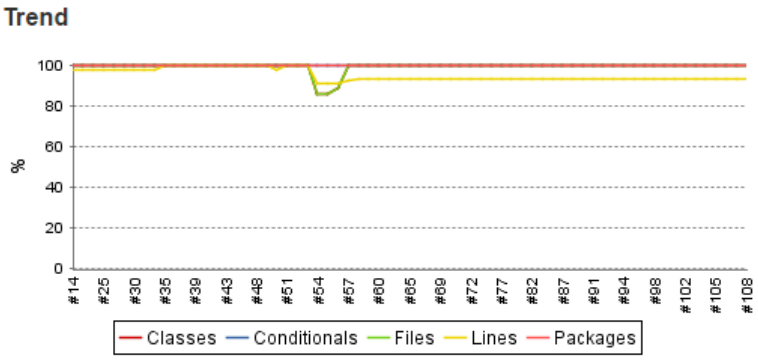

Figure 4.2: The rendered output of the test coverage report from nosetest in Jenkins. Note the incomplete test coverage for Lines (yellow) and see explanation in section 4.2.2.2.

4.2.3 Unit Tests

To ensure the functionality of the system, unit tests were created to test functions. The unit tests were written using the Python unit test framework provided by the Python language [11]. The unit tests are designed to be atomic and test a single part of functionality per test. As such, more complex functions that are dependant on the state of the system, or have multiple return values, will have multiple tests associated with them, designed to exercise every line of the code. Unit tests were written in classes where one class will test only one Python class or module.

The tests for the hologram creation software are stored in a different directory to the source code, meaning the functions can not be directly called from the unit tests. Python's solution to this is to import the source code modules from the other directory using the keyword

```
import
```

This, however, is only possible if the shared parent directory of both the tests and source code is a Python module itself. If this is not the case, a context file should be used to resolve the issue. The context file (Appendix C, section ...) is used to import the module by first setting the correct directory path for Python to be able to find it, then importing the module from the current working directory. The context file is then imported at the start of every test.

Figure 4.3 shows the graph created by Jenkins and nosetest that represents the pass and failure state of each unit test for every build. The graph shows an upwards tread as the number of tests being created in the project increase with the creation of additional features. To be accepted into the master branch of the GitHub repository, all tests must pass.

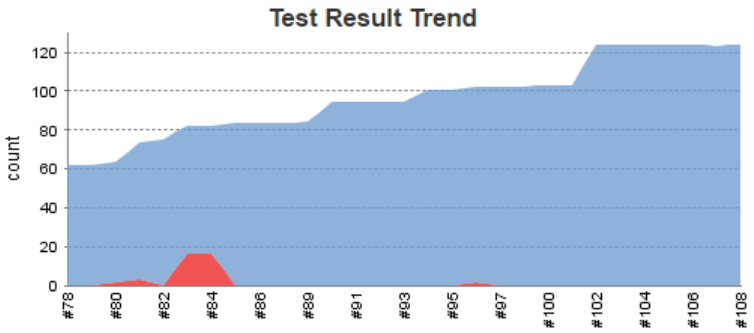

Figure 4.3: The rendered output of the unit test report in Jenkins. Blue indicates tests which pass, while red indicates those which failed.

4.2.4 User Interface Testing

The Python Selenium module was used for all tests of the user interface. The Selenium library allows for interaction with a web browser in a testing environment. By providing a URL for a website, Selenium will load that web page and then allows common user input actions to be performed programmatically. These actions include, but are not limited to, button clicks, filling in text fields and changing the current URL of the page. All user interface tests, for this project, have several generic test cases are always implemented. These are:

- Page elements: The generic view of a page is tested to ensure that the elements on the page are what the test expects. This is mainly used as a test of the Django template code, written in HTML, and the embedded Python.
- Navigation: Most pages of the website will either have a button to take the user to another page, or poll the API until the conditions are met to redirect the user. In both cases, a test case will exist to ensure the page transition is correct.

In addition to these tests, the website had to be tested to ensure that the URL could not be manually changed to affect the state of the game. An example of this would be attempting to visit the phrase_selection.html page (where the actor selects a phrase) as a viewer. In a scenario like this, the viewer is redirected to the index page with an error message displayed.

4.2.4.1 Selenium start up time

For continuity between tests, much of the online documentation that was used in this project, such as *"Selenium with Python : Using Selenium to write tests"* [12], proposed running a new instance of the Selenium web browser for each test. The reason for doing this is that it will restart the web browser and remove any cached data or session information. When run locally on the developer machine, Selenium took on average 6 to 8 seconds to create a new instance of a web browser. Hence, with a growing number of Selenium test, the time taken to run the full test suite was no longer practical.

To remedy this, it was decision was made to use only one Selenium web browser instance per class of tests [13]. The Selenium instance was initialised in the class setUp function and closed in the class tearDown, and then the cached data was manually removed from the web browser instance. Whilst this does leave scope for having some cached data remain in the web browser between tests, this reduced the number of web browser reinitialisations to be equate to the number of test classes (8) as opposed to the number of test cases (27), hence saving over a third of the time when executing the tests.

4.2.4.2 Using Selenium with Django and Localhost

During the development of the project, the website was hosted on the local development machine using local host. In order to test the website with Selenium, a hosted version of the website must be accessible. For manual developer testing, Django offers the ability to run a server locally and host the website using the command:

```
python manage.py runserver
```

to start the server. The website is then accessible from http://localhost:8000 (by default). To allow Selenium tests to run successfully, a method of starting the server before tests were run needed to be implemented.

A possible way of doing this was to run the above command to start the server before tests were run. Whilst this does resolve the problem, it forces the tests to be reliant on the correct port (default 8000) being used to access the website. This port can be specified upon starting the

server, but this could produce conflicts in different development environments if, for example, the development machine had other services running on the same localhost port.

A more permanent solution was discovered in the Django documentation [13]. This section of the documentation described the use of extending the StaticLiveServerTestCase rather than extending Python UnitTest. The extra functionality provided by extending this module means that when the test class is run, it launches a live Django server to host the Django web content stored in the same project. This service defaults to running on http://localhost:8081, but the full URL is accessible during test execution with the Python code:

```
self.live_server_url
```

This above code returns a string value that matches the correct port address on localhost (where the server is running). Using the LiveServerTestCase resolves both problems mentioned above as firstly, there is no longer a need to hard code the URL for the localhost server address and secondly, the server exists only for the life span of the test meaning there is no need to start a separate server manually before the tests are executed.

4.3 Manual Testing

4.3.1 Charades Game

In addition to testing the web site code with automated unit tests, during most features, a manual test performed by the developer was carried out. Manual testing was designed to be a supplementary testing phase to ensure that functionality was correct and no exceptions or errors were raised unexpectedly. In the case that problems were discovered, these issues would be written as a formal test case and added to the test classes to ensure the functionality is checked in the future. The Manual testing followed multiple patterns of use.

- Normal user work flow: The expected user work flow is followed to ensure that a typical user performing valid, and expected, operations causes the expected behaviour of the system with no failures.
- Unexpected behaviour: Users will not always follow the expected work flow of the website and can occasionally enter values incorrectly. This testing was performed on new features user interfaces and ensured:
  - No invalid entries could be put into text or number entry fields.
  - Submitting blank fields in forms does not produce errors.
  - Pressing the back button on a web browser does not cause errors for either the user or the system.
- Malicious behaviour: Tests were carried out to ensure that the website could not be compromised deliberately by a user. Some of the information passing in this website uses the HTTP GET protocol and subsequently, that information is stored in the client side URL. It is possible that a user with malicious intentions may attempt to taint the information provided in the URL to corrupt the system. As such, test were carried out to ensure that information in the URL was sanitised and expected.

4.3.2 Hologram creation program

Manual testing was carried out a smaller scale for the hologram creation program. As discussed in section 4.2.2.2, several lines of code within the main display loop could not be unit tested. To ensure the functionality of this code, manual testing was performed to ensure correct functionality. This was the only manual testing performed on the hologram creation program as there was no user input and output required.

4.4 User Testing

User testing took place at the Aberystwyth Science week event in March 2017. The event lasted for three days and gave the opportunity to test the prototype of the hologram system with the target audience. A black tent was set up in the event hall and this housed the touch screen table being used as an external monitor for the holographic display. The web cam for the video feed input was set up outside the tent and used to record viewers. When new viewers entered the viewing area, initially a test video would be running and then, after explaining the technique, audience members were given the opportunity to be filmed to create a hologram that was displayed to their peers.

4.4.1 General findings

The stall at the event generated much intrigue amongst the audience. Many thoroughly enjoyed seeing their peers project holographically, but there was a clear lack of purpose for those being filmed other than waving to a camera. This helped to confirm the need for the charades system to be used alongside the hologram creation system.

4.4.2 Limited time

Given the events size, the number of stalls and the limited time that each visiting school had at the event, a new consideration needed to be made regarding the amount of time required to play the game. Whilst the initial idea had been to swap users who were acting after every correct guess, this would invest a lot of the time that users had at the stall with swapping players rather than playing the game. To accommodate for this, the game flow was amended to have 3 phrases acted by the same actor and points accumulated by the viewers.

4.4.3 Camera distance

The charades game will require the actors whole body to be visible on camera while they are acting. At the event only the head of the actor was filmed using a standard web cam. Whilst this was in part because there was no advantage to filming the rest of the body given the state of the prototype, this was affected by the amount of space in the event hall. To resolve this issue at the next event, a wide angle camera orientated vertically would allow for a larger amount of the body to be filmed in same amount of space. However, this approach may require adaptations of the hologram creation software to avoid over cropping the video feed.

Chapter 5

Evaluation

Examiners expect to find in your dissertation a section addressing such questions as:

- Were the requirements correctly identified?
- Were the design decisions correct?
- Could a more suitable set of tools have been chosen?
- How well did the software meet the needs of those who were expecting to use it?
- How well were any other project aims achieved?
- If you were starting again, what would you do differently?

Other questions can be addressed as appropriate for a project.

Such material is regarded as an important part of the dissertation; it should demonstrate that you are capable not only of carrying out a piece of work but also of thinking critically about how you did it and how you might have done it better. This is seen as an important part of an honours degree.

There will be good things and room for improvement with any project. As you write this section, identify and discuss the parts of the work that went well and also consider ways in which the work could be improved.

In the latter stages of the module, we will discuss the evaluation. That will probably be around week 9, although that differs each year.

Appendices

The appendices are for additional content that is useful to support the discussion in the report. It is material that is not necessarily needed in the body of the report, but its inclusion in the appendices makes it easy to access.

For example, if you have developed a Design Specification document as part of a plan-driven approach for the project, then it would be appropriate to include that document as an appendix. In the body of your report you would highlight the most interesting aspects of the design, referring your reader to the full specification for further detail.

If you have taken an agile approach to developing the project, then you may be less likely to have developed a full requirements specification. Perhaps you use stories to keep track of the functionality and the 'future conversations'. It might not be relevant to include all of those in the body of your report. Instead, you might include those in an appendix.

There is a balance to be struck between what is relevant to include in the body of your report and whether additional supporting evidence is appropriate in the appendices. Speak to your supervisor or the module coordinator if you have questions about this.

Appendix A

Third-Party Code and Libraries

If you have made use of any third party code or software libraries, i.e. any code that you have not designed and written yourself, then you must include this appendix.

As has been said in lectures, it is acceptable and likely that you will make use of third-party code and software libraries. If third party code or libraries are used, your work will build on that to produce notable new work. The key requirement is that we understand what is your original work and what work is based on that of other people.

Therefore, you need to clearly state what you have used and where the original material can be found. Also, if you have made any changes to the original versions, you must explain what you have changed.

As an example, you might include a definition such as:

Apache POI library - The project has been used to read and write Microsoft Excel files (XLS) as part of the interaction with the client's existing system for processing data. Version 3.10-FINAL was used. The library is open source and it is available from the Apache Software Foundation [**?**]. The library is released using the Apache License [**?**]. This library was used without modification.

Appendix B

Ethics Submission

Application Number: 6775

AU Status
Undergraduate or PG Taught

Your aber.ac.uk email address
elo9@aber.ac.uk

Full Name
Elliot Oram

Please enter the name of the person responsible for reviewing your assessment.
Reyer Zwiggelaar

Please enter the aber.ac.uk email address of the person responsible for reviewing your assessment
rrz@aber.ac.uk

Supervisor or Institute Director of Research Department
cs

Module code (Only enter if you have been asked to do so)
CS39940

Proposed Study Title
MMP Using the Pepper’s Ghost Pyramid technique to create real-time holograms for a charades style game - A system that takes a real time webcam feed and creates a hologram using the Pepper's ghost pyramid. The system is accompanied by a website that hosts a charades style game so viewers can guess what the person in front of the camera is doing. This project is designed to be a outreach tool.

Proposed Start Date
30/01/2017

Proposed Completion Date
08/05/2017

Are you conducting a quantitative or qualitative research project?
Mixed Methods

Does your research require external ethical approval under the Health Research Authority?
No

Does your research involve animals?
No

Are you completing this form for your own research?
Yes

Does your research involve human participants?
No

Institute
IMPACS

Please provide a brief summary of your project (150 word max)
The hologram system uses real-time data and whilst the system is designed for outreach use, no images or data of the participants is stored and human participants are not required for the creation or testing of this system. The website does not require login details or personal information as a session id will be provided at events to log into a 'virtual room' to play the charades game. The website does not store any data other than potentially

the names of participants used in a leaderboard system.

Where appropriate, do you have consent for the publication, reproduction or use of any unpublished material?
Not applicable

Will appropriate measures be put in place for the secure and confidential storage of data?
Yes

Does the research pose more than minimal and predictable risk to the researcher?
Not applicable

Will you be travelling, as a foreign national, in to any areas that the UK Foreign and Commonwealth Office advise against travel to?
No

Please include any further relevant information for this section here:

If you are to be working alone with vulnerable people or children, you may need a DBS (CRB) check. Tick to confirm that you will ensure you comply with this requirement should you identify that you require one.
Yes

Declaration: Please tick to confirm that you have completed this form to the best of your knowledge and that you will inform your department should the proposal significantly change.
Yes

Please include any further relevant information for this section here:

Appendix C

Code Examples

For some projects, it might be relevant to include some code extracts in an appendix. You are not expected to put all of your code here - the correct place for all of your code is in the technical submission that is made in addition to the Final Report. However, if there are some notable aspects of the code that you discuss, including that in an appendix might be useful to make it easier for your readers to access.

As a general guide, if you are discussing short extracts of code then you are advised to include such code in the body of the report. If there is a longer extract that is relevant, then you might include it as shown in the following section.

Only include code in the appendix if that code is discussed and referred to in the body of the report.

3.1 Random Number Generator

The Bayes Durham Shuffle ensures that the psuedo random numbers used in the simulation are further shuffled, ensuring minimal correlation between subsequent random outputs [**?**].

```
#define IM1 2147483563
#define IM2 2147483399
#define AM (1.0/IM1)
#define IMM1 (IM1-1)
#define IA1 40014
#define IA2 40692
#define IQ1 53668
#define IQ2 52774
#define IR1 12211
#define IR2 3791
#define NTAB 32
#define NDIV (1+IMM1/NTAB)
#define EPS 1.2e-7
#define RNMX (1.0 - EPS)
```

```
double ran2(long *idum)
{
  /*-------------------------------------------------*/
  /* Minimum Standard Random Number Generator        */
  /* Taken from Numerical recipies in C              */
  /* Based on Park and Miller with Bays Durham Shuffle */
  /* Coupled Schrage methods for extra periodicity   */
  /* Always call with negative number to initialise  */
  /*-------------------------------------------------*/

  int j;
  long k;
  static long idum2=123456789;
  static long iy=0;
  static long iv[NTAB];
  double temp;

  if (*idum <=0)
  {
    if (-(*idum) < 1)
    {
      *idum = 1;
    }else
    {
      *idum = -(*idum);
    }
    idum2=(*idum);
    for (j=NTAB+7;j>=0;j--)
    {
      k = (*idum)/IQ1;
      *idum = IA1 *(*idum-k*IQ1) - IR1*k;
      if (*idum < 0)
      {
        *idum += IM1;
      }
      if (j < NTAB)
      {
        iv[j] = *idum;
      }
    }
    iy = iv[0];
  }
  k = (*idum)/IQ1;
  *idum = IA1*(*idum-k*IQ1) - IR1*k;
  if (*idum < 0)
  {
    *idum += IM1;
  }
```

```
  k = (idum2)/IQ2;
  idum2 = IA2*(idum2-k*IQ2) - IR2*k;
  if (idum2 < 0)
  {
    idum2 += IM2;
  }
  j = iy/NDIV;
  iy=iv[j] - idum2;
  iv[j] = *idum;
  if (iy < 1)
  {
    iy += IMM1;
  }
  if ((temp=AM*iy) > RNMX)
  {
    return RNMX;
  }else
  {
    return temp;
  }
}
```

Annotated Bibliography

[1] B. Costa, “Explaining the Peppers Ghost Illusion with Ray Optics,” https://www.comsol.nl/blogs/explaining-the-peppers-ghost-illusion-with-ray-optics/, 2016.

Comsol blog that describes the Peppers Ghost Pyramid implementation used for this project.The blog details in brief how the technique works and explains this with ray tracing. Furthermore, the blog mentions the history of the technique.

[2] Manchester United Ltd., “Meet Sir Alex - the hologram,” http://www.manutd.com/en/News-And-Features/Club-News/2007/Dec/Meet-Sir-Alex--the-hologram.aspx, 2007.

News bulletin announcing new hologram of Sir Alex Furgeson in Machester United Museum.

[3] The National Museum, Melbourne, “Shane Warne - ’Cricket Found Me’,” http://www.nsm.org.au/Exhibitions/Shane%20Warne%20Hologram.aspx.

Exhibition details of a hologram of Shane Warne discussing his career in Cricket.

[4]

[5] S. Khramthchenko, “A project management application for feature driven development (fddpma),” http://fddpma.sourceforge.net/help/fddpma_thesis.pdf, 2005, accessed August 2011.

Details the authors adapted FDD methodology used when creating a project management application for FDD for a thesis at Harvard university. Not only does the author discuss the methodology he followed, (Chapter 2) raise good points for consideration this projects methodology.

[6] OpenCV team, “Installing Jenkins as a Windows service,” http://opencv.org/about.html, 2017.

The OpenCV about page that describes the support for different operating systems and programming languages.

[7] e. Erik Ramfelt, “Installing Jenkins as a Windows service,” https://wiki.jenkins-ci.org/display/JENKINS/Installing+Jenkins+as+a+Windows+service, 2013.

The official installion guide for installing Jenkins as a service on a Windows machine. The guide is hosted and written by the developers of the jenkins continuous integration platform.

[8] Steve, "Automated python unit testing, code coverage and code quality analysis with Jenkins," http://bhfsteve.blogspot.co.uk/2012/04/automated-python-unit-testing-code_27.html, 2012.

A technical blog produced by a software development consultant who specialises in Python. This particular blog includes details of setting up static analysis tests for Python and Jenkins. The blog is written for Linux (Debian) but has been adapted to aide in the set up of a Jenkins service for Windows.

[9] Python Software Foundation, "PEP 8 – Style Guide for Python Code," https://www.python.org/dev/peps/pep-0008/, 2012.

The contents page for the PEP 8 style guide for python. This is used by the Pylint linting tool to ensure code conforms to standards.

[10] J. Pellerin, "nosetest," http://nose.readthedocs.io/en/latest/, 2015.

A Python unit test exension that provides additional functionality and reporting to Python unit tests.

[11] Python Software Foundation, "Python: unittest - Unit testing framework," https://docs.python.org/2/library/unittest.html, 2017.

The documentation for the standard python unit test framework. this was used at various stages of the project to help in the creation of unit tests for python code.

[12] B. Muthukadan, "Selenium with Python: Using Selenium to write tests," http://selenium-python.readthedocs.io/getting-started.html#using-selenium-to-write-tests, 2016.

The online documentation provided by the developers of Selenium. Section 2.3, 2.4 comment on the use of webdriver.Firefox() (starting a new instance of the Firefox web browser in the setUp function that runs before each test.)

[13] Danjo Development Team, "Selenium with Python: Using Selenium to write tests," https://docs.djangoproject.com/en/1.8/topics/testing/tools/#liveservertestcase.

The django official documentation page that discusses the LiveServerTestCase implementation. This test case is used to to start an instnce of the live server for django to run on before starting tests.