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 .....Elliot Oram.....

Date .....07/05/2017.....

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 .....Elliot Oram.....

Date .....07/05/2017.....

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

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 inputs (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.

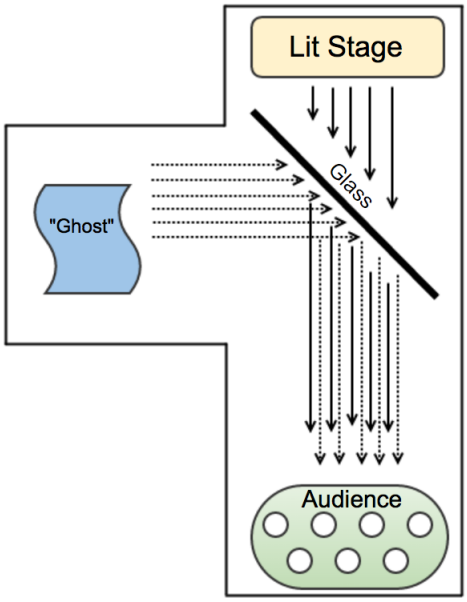


Figure 1.1: A basic diagram to display the Pepper's Ghost technique's use in theatre. In the diagram solid lines represent light from the lit stage and dashed lines as light from the "Ghost". The image has been taken from Brianne Costa's Blog on the website Comsol [1].

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 [3] and *Shane Warne - 'Cricket Found Me'* at the national sports museum in Melbourne [4].

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.

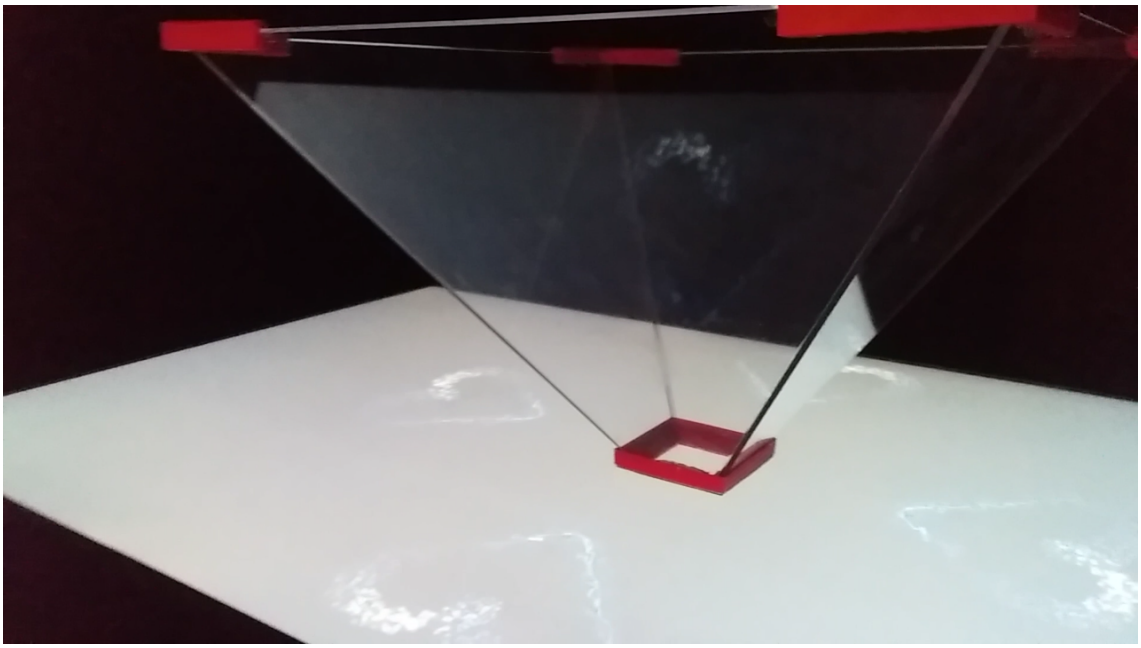


Figure 1.2: An image of modern day Pepper's Ghost Pyramid made from clear acrylic owned by the Aberystwyth University Computer Science Department.

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 utilised the Pepper's Ghost technique to give a speech to prospective voters in six cities simultaneously [5]. 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.

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 so it 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 acts 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 boasts 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 possible for users to 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 are unfamiliar to the developers. 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)"* [6]. 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 is shown in Figure 1.3.

Number	Description	Complexity (1-5)	Priority (1-5)	Dependencies	Prototype
1	Capture the video from the external camera	1	5	None	True
2	Subtract the background from the video feed	4	4	4	True
3	Rotate and scale video feeds where required	2	4	2,4	True
4	Output the video feed to a window for display	1	4	1	True
5	Display the home page of the website	1	1	None	False
6	Select the type of User you are (Viewer / Actor)	2	3	5	False
7	Select a phrase as Actor	2	3	6	False
8	Select a word from the phrase as Actor	2	2	7	False
9	Be able to log into a session/room using room code	4	3	8	False
10	Display word and genre information on Viewer interface	3	3	8	False
11	Allow Viewer to guess the current word	2	2	10	False
12	Allow Viewer to guess the phrase	2	3	10	False
13	Inform Users when the word has been guessed correctly	2	3	11	False
14	When word is guessed correctly prompt Actor for new word	3	3	12	False
15	Inform Users when phrase has been guessed correctly	2	3	12	False
16	Build list of possible phrases	1	2	None	False
17	Establish end game conditions	3	1	15	False

Figure 1.3: The full feature list for the project

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

The design for the system was completed in two major steps. First, the initial design was created before any development and secondly, 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. The diagrams displayed in this section will be the final full diagrams.

At the **Plan by Feature** level, design was performed at the start of each feature. A 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

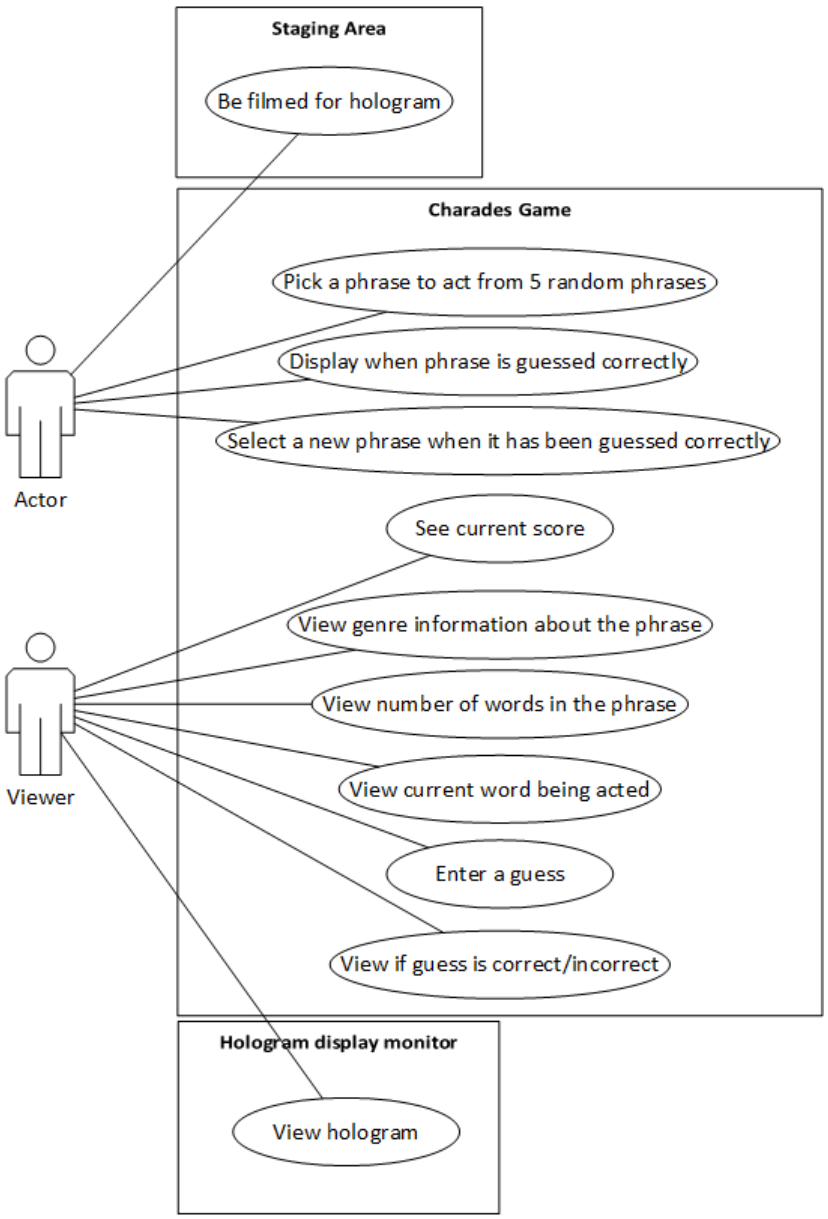


Figure 2.1: A use case diagram to show how the users will interact with the system and the functionality they require.

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 sub systems. This diagram was an insightful first step in the early development of the system as it gave an overview of the system as a whole, prior to any major implementation choices being required. Whilst the diagram was basic, it showed the interactions that users would have with the system. These interactions formed the input and output

requirements that needed to be addressed in the user interface design. In addition, the use case diagram showed that the Charades game functionality is dependant on the whether the user is an actor or a viewer. This in turn promoted a model-view-controller design pattern to be used for the system.

2.1.2 Activity diagram

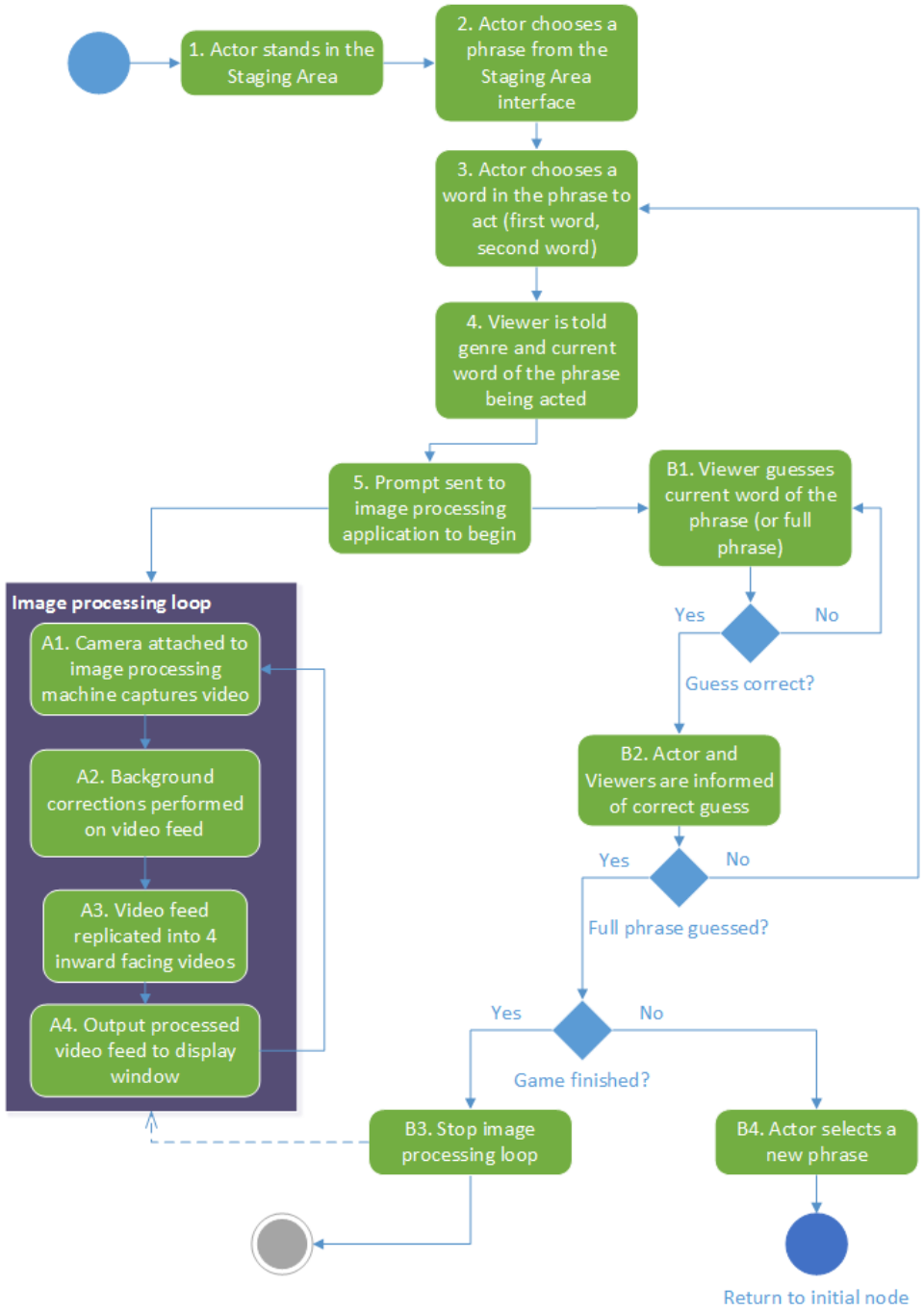


Figure 2.2: An activity diagram that describes the flow of information in the system. It includes the data flow for both the Hologram and Charades systems.

Given the system was designed 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) runs continuously in parallel with the game loop (prefixed with the letter B).

2.1.3 Component diagram

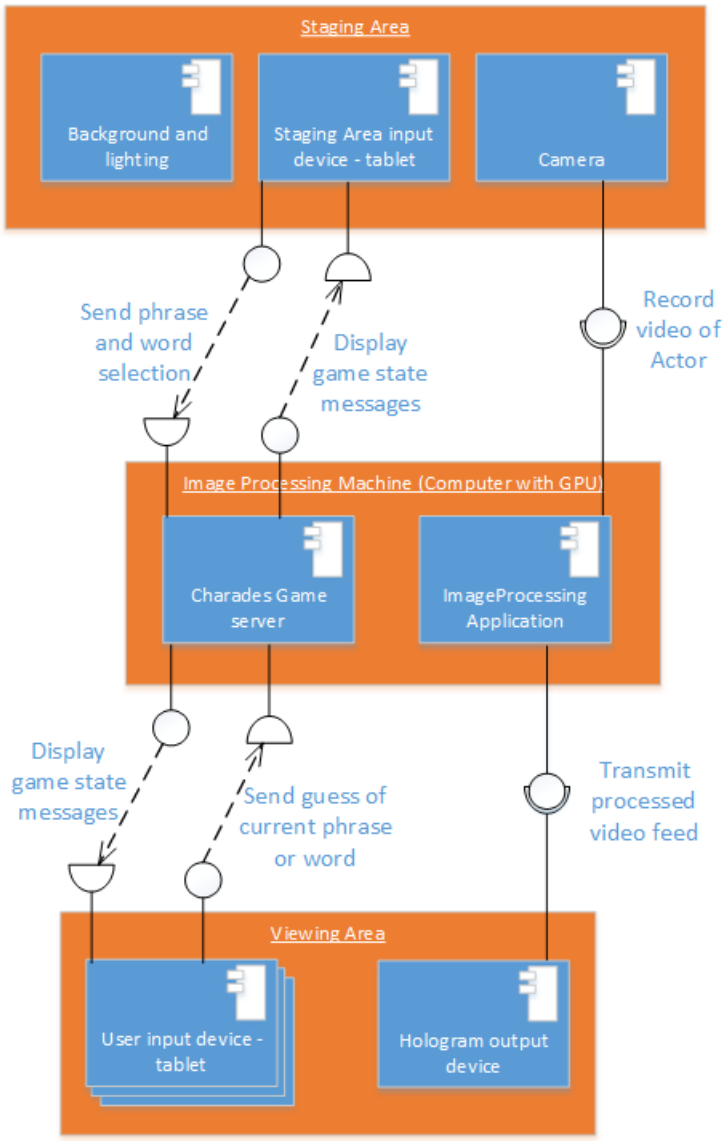


Figure 2.3: A component diagram that describes both the software and hardware components of the system, and how they interact with each other.

The system, in its totality, had to include multiple pieces of hardware in order 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 major entities in the project.

The **Staging Area** would be where the actor would be filmed. The camera directly connects to the Image Processing 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 receives 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.

The **Viewing Area** is the area where the Pepper's Ghost Pyramid and the monitor displaying the output of the hologram system are housed. The area will be a dark tent in order to produce the best quality (most visible) hologram.

2.2 Overall Architecture

2.2.1 Hologram creation system

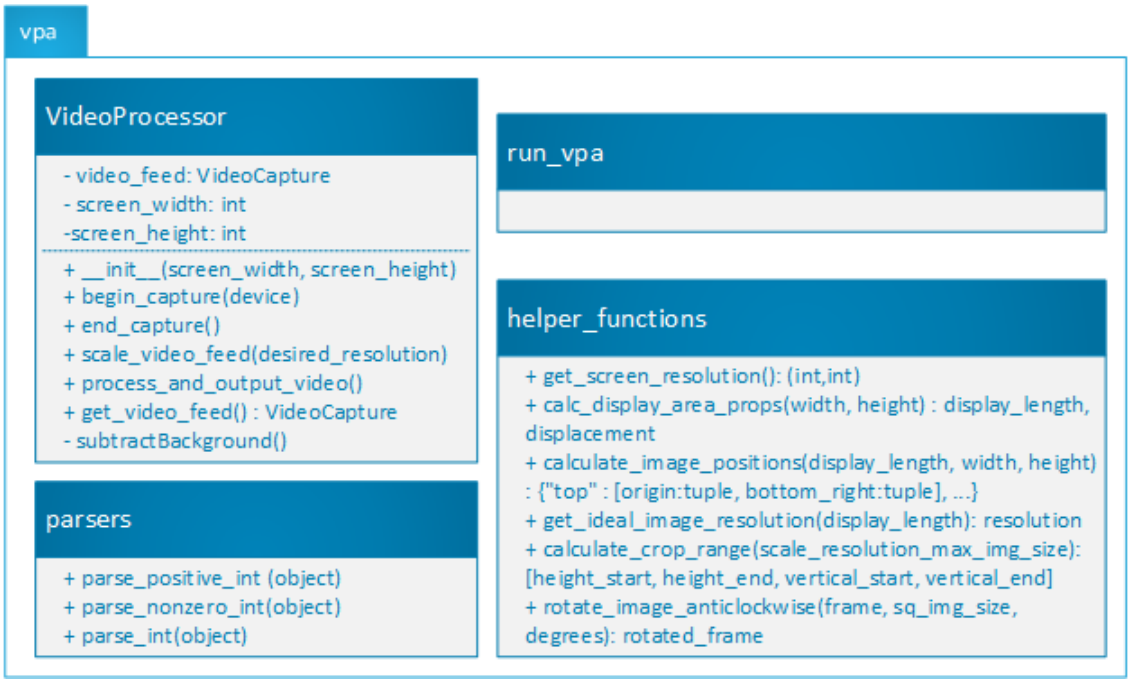


Figure 2.4: The UML class diagram for the hologram creation system. The diagram shows the vpa module and the enclosed subclasses and submodules.

- **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. It is the driver script for the VideoProcessor class and runs the required functions to 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 their own module to group them together. All functions in the parser module receive 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 important 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 functions that handle the scaling, rotation and positioning of the video feed. These functions are located in the helper_functions module.

- **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 providing simple helper functions, such as obtaining the screen resolution and calculating the size of the display window relative to the centre of the screen, the helper_functions module also aids with manipulation of the video feed. Functions used for calculating the location, cropping and rotation of the image are also in 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.

2.2.2 Charades Game

2.2.2.1 Model Design

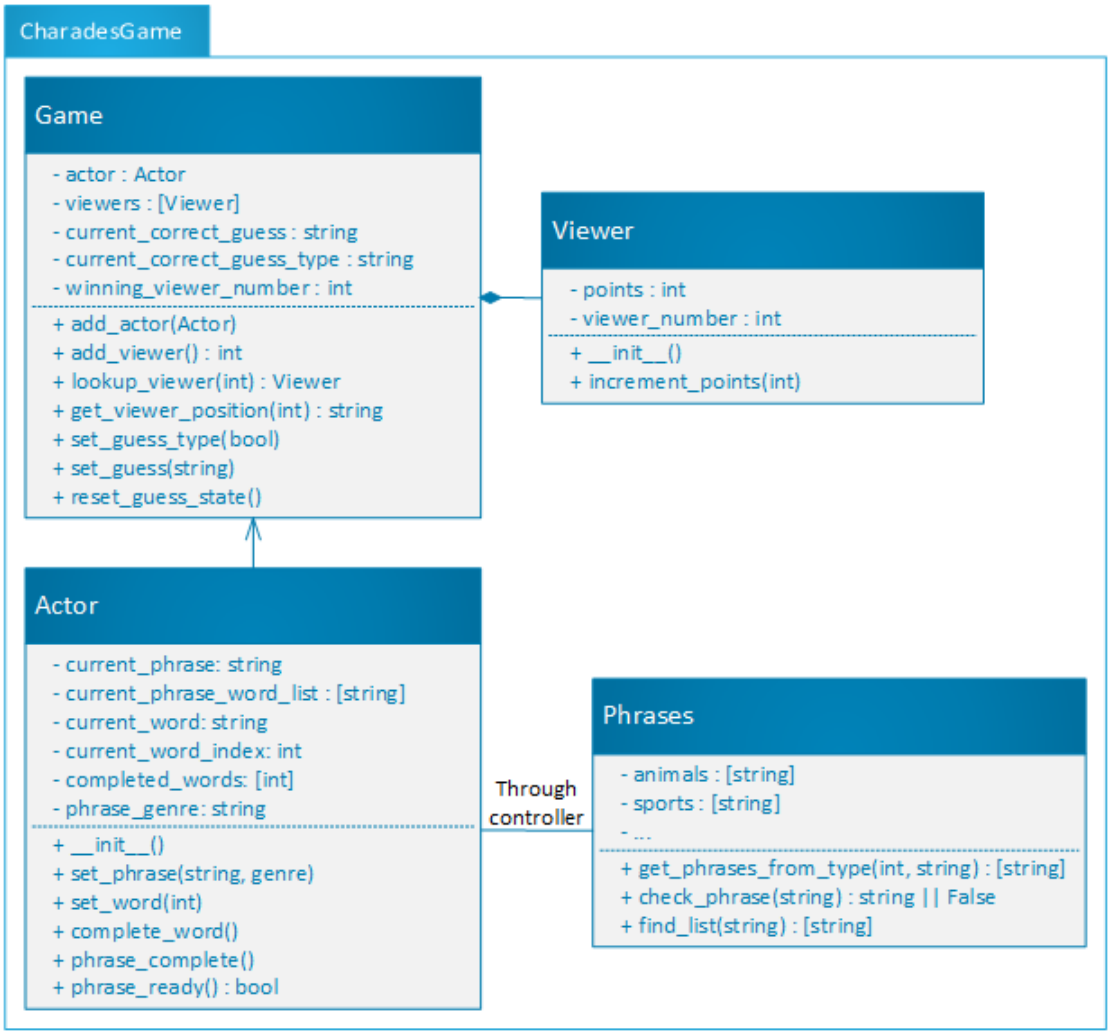


Figure 2.5: The UML class diagram for the model of the Charades Game.

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 database and instead use a class representation for data storage. The reasoning behind this is that 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 its highest level of abstraction. As well as holding a list of the Viewers and Actors, the Game also holds information regarding the whether or not a correct guess has been made, the type of the guess and the viewer_number of the winner for the round. 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 unique viewer_number of that viewer. When a viewer logs in, the viewer_number is automatically generated and assigned to a Viewer object which is 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 Actor class holds the current phrase and word variables for the phrase that is being acted. 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”. 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 lists. In addition, the module contains several helper functions designed to check phrases exist in the list of known phrases and to acquire a phrase, at random, from the list.

The phrase section 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

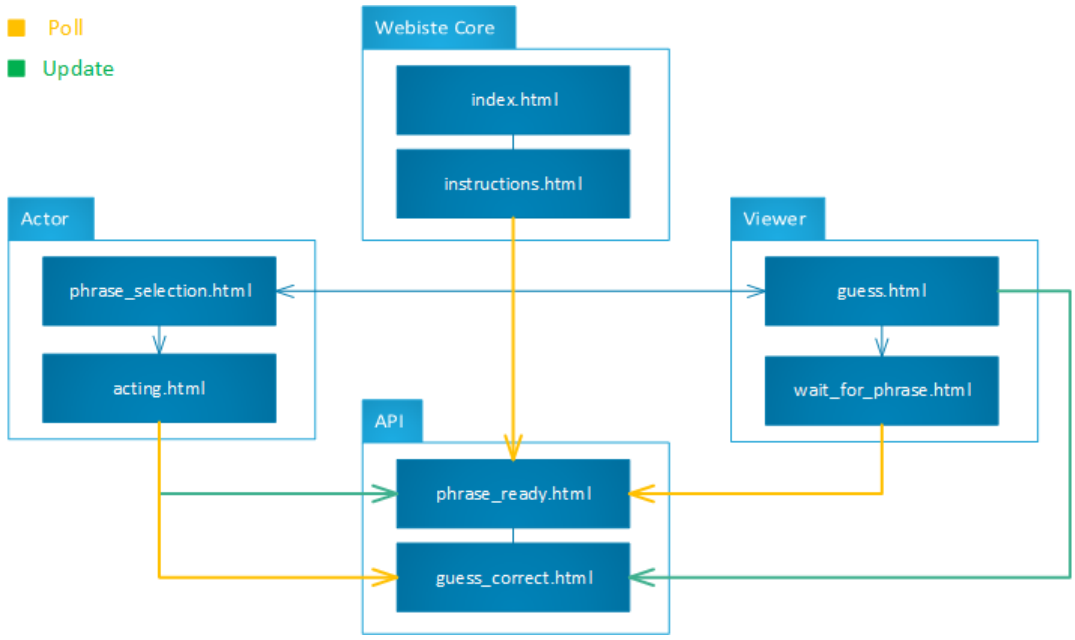


Figure 2.6: The website design for the Charades Game. The diagram shows the Template structure of the website, as well as when pages are updated from the API and when polling of the API takes place.

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 password (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 user (Actor or Viewer), and a 'continue' button for the Actor to continue to the phrase_selection page. In addition, for 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 the current phrase is being acted. The phrase_selection.html page 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, if the phrase contains multiple words, the actor is asked to choose a word from the phrase. 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 correct or incorrect guesses of the words or phrases. The guess.html page 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 viewers to submit their guesses

(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 (the API returns 'True') viewers are redirected 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. The phrase_ready.html API displays 'True' when the phrase is in a guessable state (i.e. the phrase and word have been selected and the phrase or word have not been correctly guessed) and 'False' when not guessable. The guess_correct.html API 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

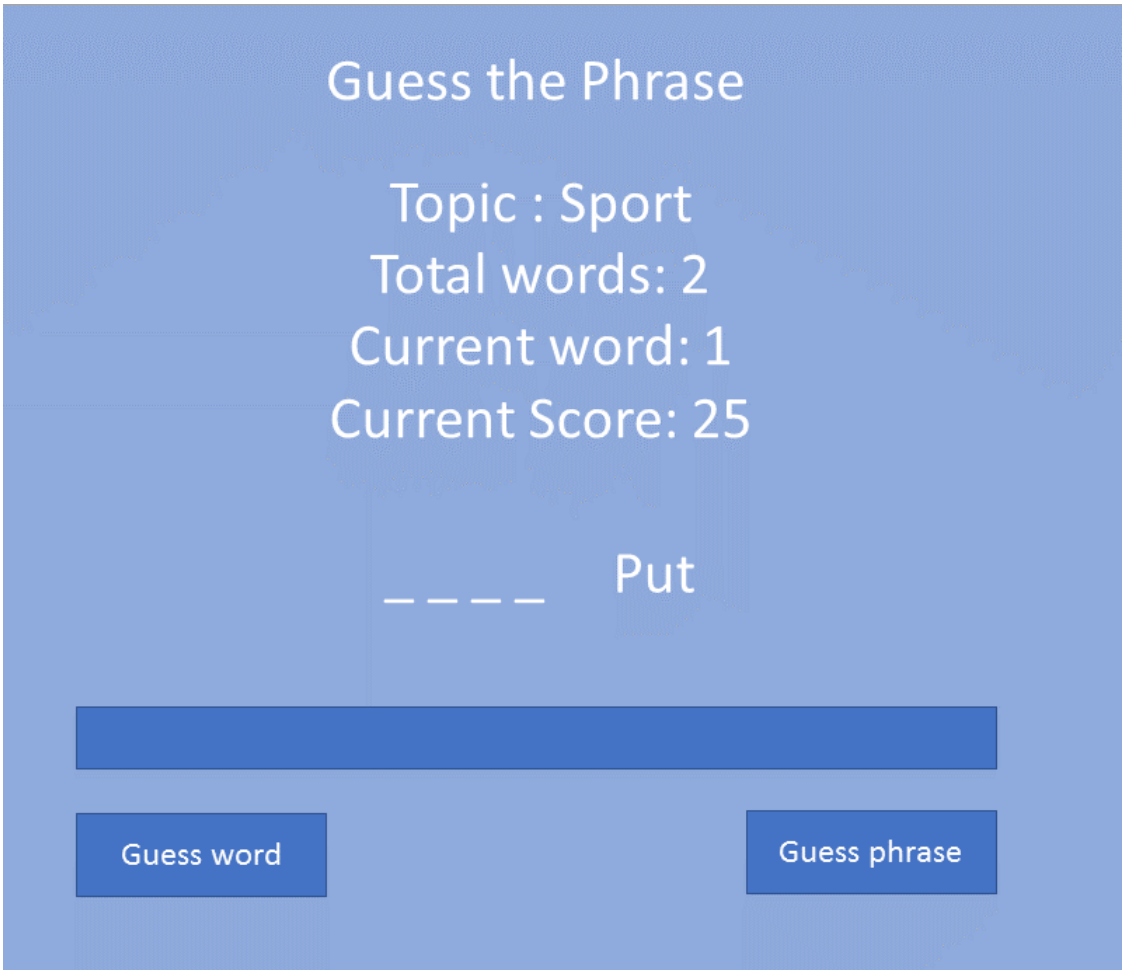


Figure 2.7: The UI design for the guess.html page. This page shows a user with 25 points and the phrase ”Shot put” is being acted. The word ”Put” has already been guessed, meaning the word ”Shot” is represented by ’_ _ _ _’ as it has yet to be guessed.

Figure 2.7 shows the UI design for the viewers guess.html page (All other UI diagrams are available in Appendix D). 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 2.5, 2.6). 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 [7]. 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 Visio

Microsoft Visio is a software package designed to aid in the creation of diagrams. It was used extensively in this project to produce all the design diagrams seen above. For more information see the Microsoft Visio product information page [**?**]

2.3.6 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>`

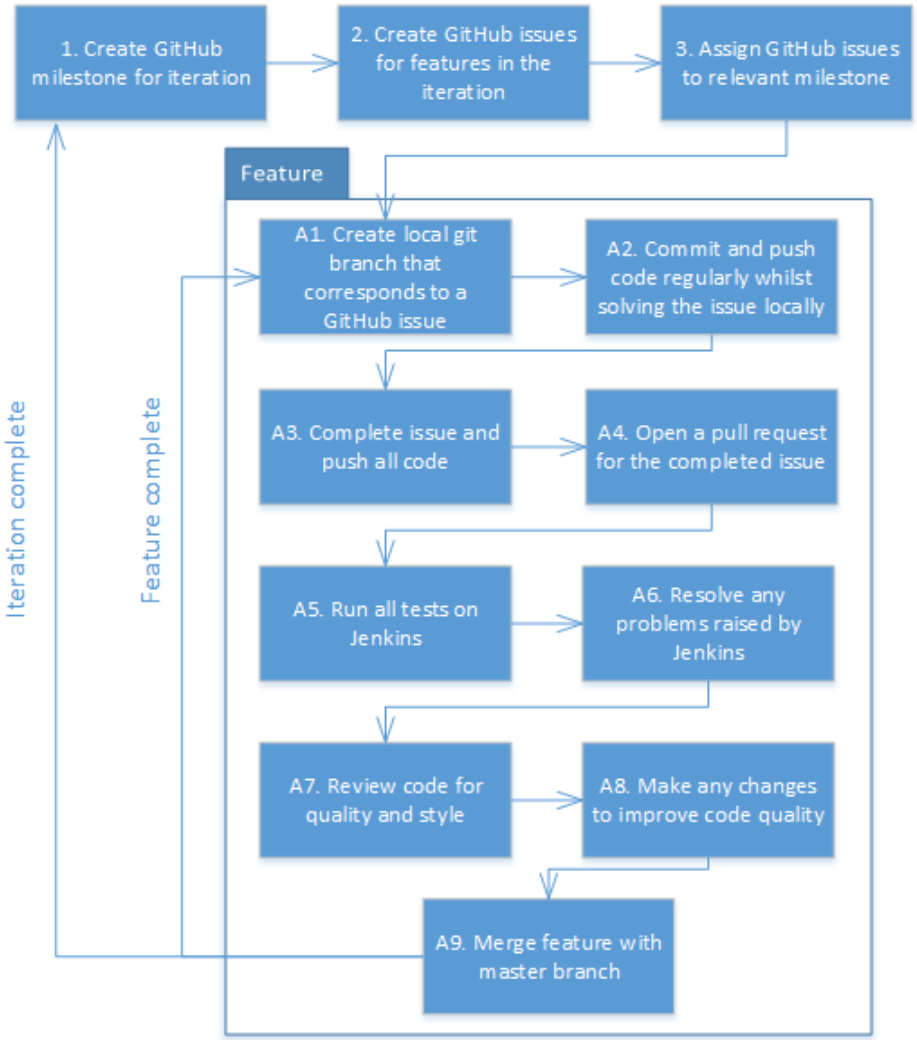


Figure 2.8: The GitHub work flow diagram used for development. The outer processes (1-3) are performed for every iteration. The inner processes (A1-A9) are performed for every feature.

Chapter 3

Implementation

3.1 Initial work

Development was carried out in weekly iterations starting from Monday 27th February 2017. To enable a smooth start to development, several tasks were completed before this, starting with setting up the development and testing environment.

To develop the charades game, both python and OpenCV needed to be set up. Installing python on windows is done by downloading the windows installer from the python.org website and running it - following the instructions on the installation wizard. OpenCV requires the pre-requisites of NumPy (A mathematical library for Python) and SciPy (A scientific library for Python). Both of these were installed using the pip tool (A Package manager for Python). Once these were installed and verified to be working OpenCV was downloaded, extracted and the binary file added to the Python site-packages directory. For more information on setup and running the program, see the README.txt file packaged with the code base.

3.2 Iteration 1

Date: 27/02/2017
Features: 1, 3, 4
Overview: The first iteration focused on putting in place the ground work for the hologram creation application. In addition to developing an understanding for the OpenCV library, tasks for this iteration also included obtaining and displaying a video feed from a web cam, as well as rotating the image obtained from the web cam.

3.2.1 OpenCV API

Due to the diversity of the OpenCV library with regards to its platform deployment and versioning, the documentation for the library is fragmented making examples difficult to find. Most documentation is basic and the online API covers all programming languages. It was for this reason, that much of the first iteration was spent deciphering the way the library should be used. The online documentation did provide a tutorial for the implemention of capturing the output from a web

cam [8]. This tutorial was used as a starting point for the system (Feature 1).

3.2.2 Deep versus shallow copying

Originally, the feature list contained an additional feature to be implemented during this iteration. The feature was to create 4 copies of the video feed to display them on each side of the pyramid. Original spike work suggested that this could involve creating a deep copy of the VideoCapture output. The reason a deep copy would be required is that each instance of the video feed would need to be rotated independently of the others. If the default operation for copying a video frame in Python is a shallow copy, then both variables will share a pointer to a single object [9]. If this is the case, then rotating the video frame will rotate all the instances rather than just one. Through further investigation, it was discovered that each frame of the video was of type NumPy array. Furthermore, the output window, where the video feeds were being displayed, was also a NumPy array. This removes the problem of having to make a deep copy of the array as the numbers from the array can be directly added to the output array rather than copying the frames individually.

3.3 Iteration 2

Date: 27/02/2017
Features: 2, 5, 16
Overview: Most work in this iteration involved investigation and implementation of a method for background subtraction in the charades game. This feature was estimated to take 3 or 4 days due to its complexity, as such minor features from the charades game were also added to this iteration.

3.3.1 Background subtraction research

To begin this iteration, research took place to determine the best possible method for background subtraction from the video frame. The ”Comparative study of background subtraction algorithms” [10] provided an interesting summary and comparison of possible background subtraction routines. Given the need for real time execution of the background subtraction, it was decided to use the simple background subtraction technique. Simple background subtraction is performed by taking an image of the scene without the subject (foreground) present. For all frames that follow this, the background image is then subtracted from the frame which will leave only the foreground image.

3.3.2 Image subtraction

The simple background subtraction technique was implemented using the first frame captured by the web cam as the background image. Then when displayed, this image was subtracted from the real-time data from the web cam. Originally, this was implemented in full colour but due to the camera shake, auto focus and lighting adjustment made by the camera itself, the background image differed too much from the real-time foreground. To compensate for this, the background images was changed to be grey scale reducing the variance in colour from three channels to one. The grey scale background was then subtracted from a grey scale version of the web cam output and the resulting image was used as a mask. The mask could then be applied to the real-time video feed to only display the content that is outside of the mask. Using a mask improved the quality of the background subtraction, but not to the degree that was acceptable, as the mask that was generated was poor and contained holes due to the quality of the input video.

3.3.3 Green screen

The approach of using a green screen is commonly used in film and television production. This technique involves placing a green screen behind the object of interest, and then removing the green colour from the scene programmatically. The green colour is used for filming humans as it is a stark contrast to most skin tones. This implementation proved more effective in the quality of the output image produced. However, due to the video feed being captured in real-time, the function to remove the background needed fast execution to not reduce the frame rate of the display. The camera being used captured video at a frame rate of 60 frames per second (FPS). Applying the algorithm for removing the background reduced the frame rate to close to 10 FPS as each individual pixel had to be checked to find if the colour was part of the background or foreground. Despite attempts to optimise the algorithm, the best FPS achieved was 15.

The background subtraction algorithm implementation was left at this point to allow for other development to take place during this iteration. Planning allowed for work to continue this during the next iteration providing no set backs were encountered. This was an acceptable decision as a fall-back plan of using a black curtain as the background in the scene could be used if required.

3.3.4 Defining the phrases

Instead of further development on the background subtraction implementation, ideas for possible phrases to be used in the charades game were gathered. This involved research into the types of books and films that are popular with a younger target audience. Most of the phrases were selected from the *"100 best children's books"* list from BookTrust [11] and Sara Schmidt's *"15 Best Kids Movies Of 2016"* publish on screenrant.com [12].

Feature 5 was pushed back to the next iteration due to the setbacks with the background subtraction algorithm.

3.4 Iteration 3

Date: 06/03/2017
Features: Spike work for Charades Game
Roll-over features: 5
Overview: As well as completing the remaining work (Feature 5) from the previous iteration, this iteration covered the set up and spike work for android application development. Whilst in the end the Charades game was implemented as a web application, at this stage of the project, the design was for an android application. This decision will be discussed in more detail in **Iteration 4**. Finally, this iteration reopened the investigation into the optimisation of the background subtraction algorithm.

3.4.1 Android setup

Android Studio was the IDE chosen for Android development as it is one of the largest Android specific IDEs available. Furthermore, the online documentation for Android Studio provides multiple guides that help with the set up and implementation of a basic android app. After downloading and installing the IDE, the *"Create an Android Project"* guide [13] was followed to aid in the setup of a blank project. The tutorials that were available meant that spike work was completed quickly and yielded a basic prototype for the android application.

3.4.2 Android status bar

Figure 3.1: Shows a mock android application with the status bar open. The status bar is the black panel with the text "System UI" and the approve device information such as notification, signal, battery, ect. This image was taken from the "Hiding the Status Bar" tutorial created by the Android development team [2].

An additional task that was required alongside the creation of the welcome screen of the Android app was to remove the status bar from the Graphical User Interface (GUI). The status bar can be seen in Figure 3.1 and the *"Hiding the Status Bar"* tutorial [2] from the Android developer site,

gave an insight into how this could be removed from the app. When implementing the suggested changes from the tutorial, an error occurred routing from the call to the getActionBar() method that returns a pointer to the status bar. A stack overflow issue detailing this error [14], helped to diagnose the cause of the issue. The version of android being used was utilising the new

```
android.support.v7.app.ActionBarActivity
```

which needed to be accessed as a support status bar rather than the older style regular status bar. This could be done using the function:

```
getSupportActionBar()
```

3.4.3 UI testing

To ensure that all elements of the Android application were subject to some form of automated testing, UI tests were developed for the Android app. These tests included checking page contents as well as page transitions to ensure that the expected content was visible and the buttons functioned correctly. As described in the Android developers guide *"Testing Apps on Android"* [15], two types of test are commonly used for testing Android applications.

- **Local unit tests**: Unit tests in this case are similar to any standard unit test for a normal code base. They test function input and output given various states. These tests are run on the local machine using the Java Virtual Machine (JVM) and have access to the Android specific API functions.
- **Instrumented tests**: By contrast, the Instrumented tests run on a hardware device or emulator. They are designed to test the app while it is running rather than testing individual functions, like a unit test. This allows for tests that exercise button transitions or page content to be run in an automated way.

For the UI testing, Instrumentation tests were developed and run on a mobile device attached to the development machine.

Whilst running tests for page content was straightforward, attempting to test page transitions proved more complex. A stack overflow issue was found that helped in to determine how this operation could be achieved [16]. The answer to the issue detailed using an ActivityMonitor class to listen for an Android Intent (the commands issued at the time of page transition) being executed.

3.4.4 Background subtraction multi-threading

Finally, this iteration revisited the background subtraction algorithm. Having already implemented the green screen algorithm, the algorithm or execution of the algorithm needed to be optimised to produce an acceptable frame rate. To do this, investigation in Python multi-threading took place. Each pixel in the input image needed to be checked and if it was above a certain value of green, changed to be black. This operation was performed on every pixel independently one after another. To improve the speed of execution, instead of checking the full array of the image in the main thread, the image was divided into 4 quadrants and each processed in a separate thread.

Initially this was done in place in the array where each thread was given a start and end index to check. However, after implementing this solution, it was found to not have decreased the execution speed. This was the result of the array being locked while processing. To avoid multiple threads accessing the same NumPy array and potential corrupting the data, the array is locked while one thread accesses it meaning that the next thread must wait for the first to finish.

By creating new array objects that relate to each quadrant the lock problem was avoided, however only a slight decrease in execution time was found. More detailed investigation revealed that this was caused by the time taken to initialise new arrays with the data from each quadrant.

Again, as no solution was found for execution time issue, development of the background subtraction algorithm was halted. Instead it was decided to use a black background when filming. Whilst this solution offers less versatility in the location where actors can be filmed, it means that the frame rate of the hologram remains close to that of the camera feed.

3.5 Iteration 4

Date: 13/03/2017
Features: 6, 7, 8
Overview: Aberystwyth University Science Week was scheduled to begin on the 14th March 2017 and last a duration of three days. All the work required for the prototype (Feature 1-4) had been completed or alternative solutions found, as such it was possible to take the prototype to the event to test it with the target audience. In addition, this iteration concluded with a demonstration of the project so far on Friday 17th March.

3.5.1 Aberystwyth Science Week 2017

Figure 3.2: Shows the viewing area for the real-time hologram creation system at the Aberystwyth University Science Week 2017. Inside, part of the touch screen table (being used as the display monitor for the system) is visible.

The entities listed in the component diagram (the staging area, viewing area and image processing machine) were all constructed at the event venue. Figure 3.2 shows the viewing area (large black tent housing the monitor for the Pepper's Ghost Pyramid). To the right of the tent a camera attached to a computer was set up to record members of the audience. The camera was attached to the computer via a USB cable and the computer was attached to the external monitor in the tent via a HDMI cable.

Two main pieces of feedback could be inferred from using the prototype at the event. The first is that it proved a great success, with the clear majority of visiting students appearing greatly interested and engaged by the impactful display. The second was that students were not given much time at the event (most schools were present for an hour), meaning that the time that students were able to spend at each stall was limited. Originally, the Charades Game was designed to be played where actors and viewers would swap every time a correct answer was given, as would be the rules in a conventional game of charades. However, after discovering the limited time window for each activity at the science week, this concept was adjusted going forward to better suit the use case. To do this, two changes to the game rules were made:

- Actors act three phrases, and a points system was used, giving viewers points for correct guesses. At the end of each round, the viewer with the highest points would become the actor. This change enabled more time to be spent using the system, and less time for the member of the audience to be swapping roles.
- The subject of the phrase was changed to make more phrases single words and, rather than using conventional genres such as books and films, genres were changed to be activities and animals. These new genres meant a large amount of phrases were shorter and avoided long titles with simple words. A good example of where long titles would have been a problem is with the book "The Lion, the witch and the wardrobe". This phrase could take some time to act out fully due to the large number of words and it is also difficult to act simple words like "The".

3.5.2 Mid project demonstration and conclusions

The mid project demonstration overall was a success and the demonstration of the software worked well. However, the question was raised as to why the charades system had been designed as an android application. The original reasoning for an android app was to promote the use of hand held devices for interaction with the system. Moreover, an Android app can potentially remove the reliance on the internet via the use of Bluetooth or locally connecting the devices together. This is an important factor to consider if, for example, an outreach event was being held in an area or building with poor reception. Furthermore, an android application removes problems that can occur with users attempting to manipulate the system via the URL which can be the case in some web apps.

Despite this, on re-evaluation of the project requirements and the final use case of the system, a web app was a better choice. The web app implementation removes many of the complications that could be faced with an android application which include:

- Enabling devices to communicate with one another: Whilst there are many technologies for performing communications between android applications such as over the internet, via Bluetooth or through a peer-to-peer system, these technologies can be difficult to implement and test.
- Limiting the number of devices that are available: Whilst the event organisers would normally be able to get several tablets and mobile devices to use with the system. If users wished to join in on their own devices this would require that both the device they are using is on the android operating system, and the app is available on the Google Play Store.

A web app solves the above problems as it is already hosted on the internet which will allow for implicit communication between device accessing the system. Furthermore, the web app will be platform independent as the system code is executed server side and users interact with it through a web client.

This decision was difficult as due to the events of the week, little progress had been made on the android application. Starting again with a web application would mean further setbacks, but at the time the decision was made that it would lead to a better product.

3.6 Iteration 5

Date: 20/03/2017
Features: 9, 10
Roll-over features: 5, 6, 7, 8
Overview: This iteration's first task was to adjust the features list schedule due to the decision to change to a web app from an Android app. Therefore, this iteration dealt with the roll over issues from the previous iterations, and Feature 9 and 10 (which were originally planned for this iteration) were redistributed among future iterations.

3.6.1 Django

As the hologram creation software already used the Python programming language, it was logical to use a Python framework for implementation of the website. Several Python web framework exist with the most popular being Django, Flask and Pyramid. The article *"Django vs Flask vs Pyramid: Choosing a Python Web Framework"* written by Ryan Brown [17], was a useful resource in making a decision regarding the best framework to use. In the Article, Django is identified as the most popular of the three as well as being best suited to mid scale projects like the Charades system. Furthermore, Django uses many familiar keywords and concepts to those ruby on rails which is a technology that the developer had used prior. Finally, an excellent tutorial for the Django web framework by Nigel George, *"Masting Django: Core"* [18] is available for free online.

3.6.2 Model

As described in *"Mastering Django: Core* [18], the initial web site setup was fast and straightforward. This process was made simpler by not having any requirement for a database and having models written as Python classes. However, using Python class meant that there was an additional requirement for tests to ensure the model functioned correctly. This would not have been required of a database as they are tested by the database supplier (for example SQL) before release. The tests for the Python class models were written as Python unit tests. As the model did not rely on the Django web framework, classes could be considered independent from the rest of the system and tested in isolation.

The model can be found in the directory /charades/charades/ directory. The model consists of actor.py, game.py, viewer.py phrases.py and strings.py.

3.6.3 Template

The templates in Django are written in html and can use embedded Python code to execute additional commands or access variables. The variables are pushed to the template through the controller and can then be accessed by surrounding them in double curly brackets:

```
{{ variable_name }}
```

Embedding python code also proved powerful for flow control and conditional statements. An example of both of these being used can be found in Appendix C section *Embedded Python*.

The template code can be found in the /charades/templates directory.

3.6.4 View

The view code is the controller for the website and acts as the link between the model (data) and the template (information displayed to the client). The views.py file is where this code is stored and consists of multiple view functions. The view functions must have take an input parameter of type request (which is a HTTP request header) and must return a request along with some rendered data (the template code). It is at the point where the request is returned that additional data can be provided to the template in the form of a python dictionary.

The HTTP request is handled by the urls.py file which holds the routing information between the URL and the controller. Figure 3.3 visually represents how these interactions take place.

The view code can be found in the /charades/charades/views.py file.

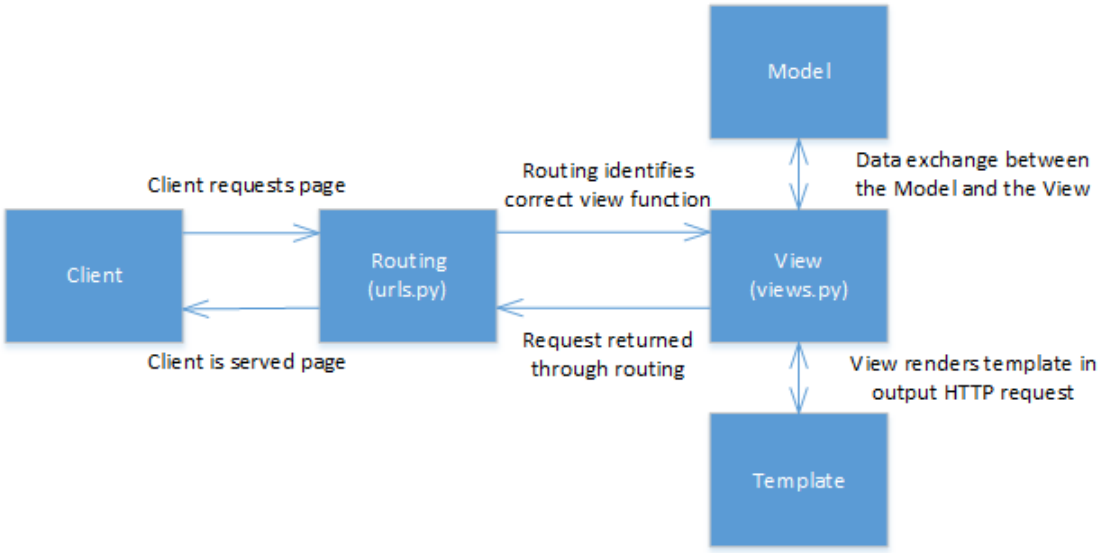


Figure 3.3: Show the interactions between the client, the urls.py (routing) file, the template, the view and the model in the Django web framework.

3.7 Iteration 6

Date: 27/03/2017
Features: 9, 10, 11
Overview: This iteration covered access and security for the Charades Game and dealt with functions to ensure that only those at an event would be able to log on. Furthermore, to improve the security of malicious users, redirects were put in place to stop access to certain pages depending on the game state.

3.7.1 Session password

To disable access from those outside an event (such as the Aberystwyth Science Week), a session password was added to the landing page of the website. When a user chooses a type (actor or viewer), they are also required to provide the session password to access the system. By default, the session password is stored in the view code as *"BSW18"* but this can be changed easily. If mobile devices capable of accessing the system were provided for an event where the system was in use, the organisers would be able to type in the session password before the arrival of any visitors. Alternatively, (if guests were to use their own devices) the session password could be displayed on a sign near the viewing area enabling users to log in themselves.

3.7.2 Cookies

When a valid session password has been provided, the password is then stored client side in a cookie. The Django online documentation [19] provided a tutorial which aided greatly in the implementation of a session cookie. In addition, the cookie also stores the type of user (actor or viewer) and, if the user is a viewer, the unique number used to identify them. This unique number is called the viewer_number and is generated by the system when a new viewer joins. The number can then be used to lookup the viewers score:

```
viewer_number = request.session['viewer_number']
...
viewer = GAME.lookup_viewer(viewer_number)
points = viewer.points
```

The above code is from the waiting_for_actor function in the charadescharadesviews.py file. The code returns the current score of the viewer accessing the waiting_for_actor.html page. The code first finds the viewer number from the session variable (cookie), and then calls the lookup_viewer function in the Game class, using the viewer_number as input. This will return the Viewer object relating to the viewer_number provided. Finally this object can be used to see how many points the user has accumulated.

3.7.3 Redirects

3.7.3.1 State based redirects

Each page of the website (excluding the landing page) queries a part of the model. It is possible that, without proper handling, either intentionally or not, a user can access a page that the system state does not yet access to as it is missing data required to render the page. An example of this is if a viewer attempted to access the guess.html page (where they are asked to guess the current word or phrase) before the actor has chosen a phrase. Accessing this page at this time could cause errors to be thrown and potentially corrupt the game state. To stop this from happening, state based redirects have been added to the website controller. These are functions that, instead of returning the page content that would normally be provided from the requested URL, redirect the user to a different URL. The different URL will then handle the request appropriately. In the above case of accessing the guess.html page before it is ready, the viewer will be redirected to the instructions page which displays a ”Please wait” message as shown in Figure ... .

** ADD FIGURE FOR REDIRECT ON GUESS.HTML **

3.7.3.2 User type redirects

Similarly, users can attempt to access webpages that are only designed for a different type of user. An example of this is that the phrase_selection.html page is designed for the actor to select a phrase to be acted. Once selected this data is then passed to the model and the viewers are able to guess the word or phrase. It should not be possible for a viewer to access the select_phrase.html at any time, therefore user type redirects have been added to the controller to ensure this is not possible. The user type redirects are designed to check the type of user attempting to access the content by searching for the 'user_type' entry in the session variable. If the incorrect user_type, or no user_type at all, is found in the session variable, the client will be redirected to the landing page, and an error message will be displayed as shown in Figure ... .

** ADD FIGURE FOR REDIRECT ON YOU'RE NOT AN ACTOR **

3.8 Iteration 7

Date: 03/04/2017
Features: 12, 13, 14
Overview: This iteration focused on allowing communication between user in real (or near to real) time. As well as implementing a solution to this, several other possible solutions were investigated and evaluated.

3.8.1 Real time updates

The charades game has multiple users logged in at the same time and has several scenarios where interactions need to take place in real-time. An example of a real time interaction, is that when a phrase is guessed correctly by a viewer, the actor must be informed and prompted to select a different word.

When the Django framework was created, it was designed to handle static website that were more common of those seen 5 or 10 years ago. In this respect, at its core Django is built around answering requests from the client, but does not have any methods to push data to the client by default. However, many technologies provide solutions to real time communications for the web and can be implemented in Django. The approach chosen for this project was to use an API and a polling system to mimic the pushing of data from the server side to the client.

An alternative consideration was made to use the new Django Channels technology. Django Channels provide a way to opening a bi-directional connection between the server and the client, allowing data to be pushed both ways (client to server and server to client) [20]. This technology utilises the new WebSocket protocol. Whilst this technology would provide a solution to the aforementioned problem, implementing the technology is a more complex task than a implementing a polling system. It is however a better and more long term solution but, the additional time it would have taken to implement would have set the project back further.

3.8.2 API

An Application Programming Interface (API) was created as a way to programmatically assess the current state of the system. As with any Django page, when a request is sent from a client (for example the user types the page URL into the address bar) the page is loaded and any data that is embedded in the template will be updated automatically. In the case of an API, simple data that represents a fact in the system can be displayed. In the case of phrase_ready.html, this page, when loaded, is given data from the controller. The controller function (See Appendix C, Section *phrase_ready API*) checks the current state of the system, and produces a True or False value as to whether the phrase is in a guessable state. The state of the system will change depending on the user interaction with the system. For example, if a phrase has not yet been selected by the actor, the phrase_ready API will display False. Once a phrase has been selected, the controller will receive the data from the client request, update the current phrase in the system and then, when the API is polled (loaded) it will have the new value of True.

3.8.3 Polling

As Django does not provide a built in methods of pushing data to the client (as discussed above), the client most poll the API to establish if the state of the system has changed. This is done using the JavaScript setInterval() function [21]. This function takes a parameter of another function and a time interval given in milliseconds. When that number of milliseconds has passed, the provided function will be executed. To poll the API the project used a function that was designed to retrieve information from the API pages using an HTTP GET request. This function (detailed in Appendix C, section *Polling function*) was the primary input to the set interval function. When the data is received from the API, conditional statements within the function decide what should happen. In most cases this is a page refresh and then the controller will supply new data to the page.

Depending on the polling function, the API is normally polled every second with some cases having larger polling intervals if an instantaneous answer is not required. This method of communication does have one small caveat, if two users were to guess the word within one second of each other, it is possible that both viewers would receive points. The reason for this interaction is that when a viewer guesses a phrase correctly, the API is updated and then all the viewers (on the guess.html page) are redirected to the waiting_for_actor.html page. During the time the other viewers are polling the API it is possible that a guess could still be submitted and, if correct, the controller would increment the score of the viewer who guessed correctly. This would result in the above interaction where two or more player potentially score points if they guess within a second of each other. Statistically, a full second window for another viewer to guess is the worst case scenario as it is likely that a viewer makes a correct guess mid way through a polling cycle. This would result in a 0.5 second window which was deemed acceptable.

3.9 Iteration 8

Date: 10/04/2017
Features: 15, 17
Overview: The final iteration covered two main tasks. The first was to decide on a end game condition and scoring, and the second to perform some refactoring of the of the Charades Game to improve readability and maintainability.

3.9.1 End game conditions and scoring

To ensure that the game was completed in a timely manor and the outcome was fair the scoring system and rules had to be decided upon. The rules, as previously mentioned were changed to include a scoring system as well as multiple rounds. The game lasts for a duration of three rounds, where a round is a single phrase being guessed. Although the main purpose of the game is to be an aid to showcase the hologram system, for the younger target audience, having a friendly competition would prove more interesting for them. The phrase dictionary consists of one or two word phrases, as such the whole phrase is worth 20 points, and a single word is worth 10. At the end of a round a new actor is chosen based on either a volunteer or the highest scoring viewer.

3.9.2 Refactoring

The controller (views.py) for the Charades Game had been continually added to throughout the development period. It contained all the functions for executing control code in a single file, and had some code duplication. To improve this, the code was refactored to firstly ensure that duplicated code was removed and secondly to make the controller easy to read. This was an important step as, providing this system would be used in the future, it would most likely require adaptation to remain current. Refactoring the code base also gave the advantage of allowing more tests to be added to the simpler helper functions which improved the overall test coverage of the system.

3.10 Implementation review

The major objectives set out at the beginning of the project have all been met with varying degrees of success. Whilst the main objectives regarding the actual system use cases itself have all been met, some of the other areas of the system could be improved relative to the criteria of the objectives.

The system should have been designed with a greater focus on the ease of use in mind. Whilst not tested explicitly on the target audience, with more effort put in to UI design, the system could have been easier to use. As a direct result of this, there is an instructions page for both sets of users built into the system, however, if the web app requires written instructions, then it could be argued that the system could be simplified.

Whilst the hologram system is thoroughly tested, there could be room for additional automated testing of the Charades Game. The view (controller) could be tested in an automated way rather than the manual testing it underwent.

However, the rest of the objectives have been met well, especially given the large setback in the middle of the project regarding the change from android app to web app.

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 test*.

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"* [22] from the Jenkins.io wiki and Steve's Blog *"Automated python unit testing, code coverage and code quality analysis with Jenkins"* [23], 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 [24] 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.

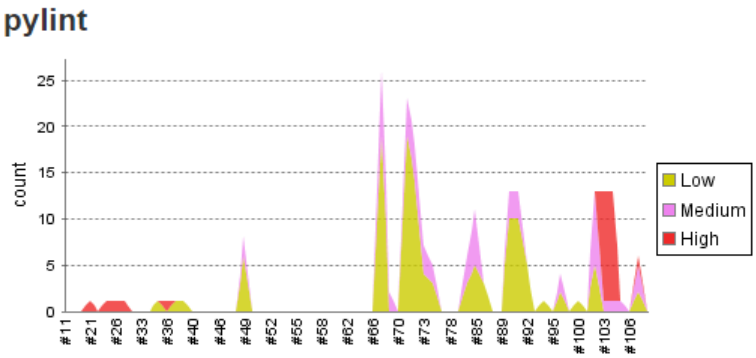


Figure 4.1: The output of the Pylint test run on the full code base rendered by Jenkins. Here, the number of warnings, and their severity, is logged by each build number.

4.2.2.2 Test coverage

An additional testing package, nosetest [25], 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. It 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).

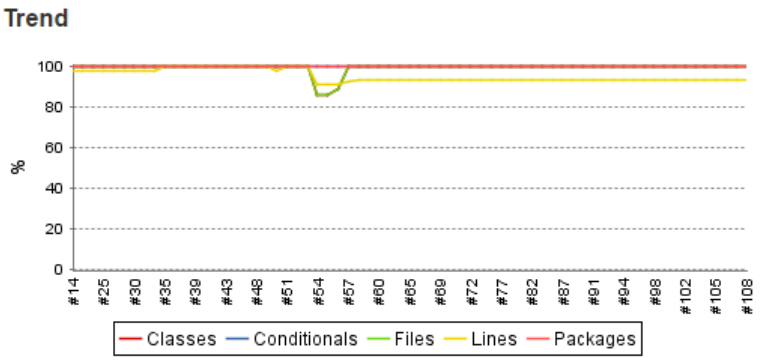


Figure 4.2: The rendered output of the test coverage report from nosetest in Jenkins. Note the incomplete test coverage for Lines (yellow) and see explanation in section 4.2.2.2.

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 [26]. 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 *Context file*) 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.

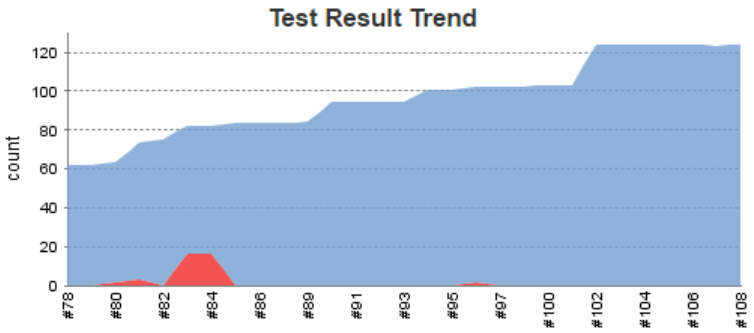


Figure 4.3: The rendered output of the unit test report in Jenkins. Blue indicates tests which pass, while red indicates those which failed.

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"* [27], 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, the decision was made to use only one Selenium web browser instance per class of tests [28]. 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 [28]. 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. 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, tests were carried out to ensure that information in the URL was sanitised and expected.

4.3.2 Hologram creation software

Manual testing was carried out on 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 projected 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

5.1 Methodology

Choosing FDD for the project proved useful from a structural and time management perspective. Whilst multiple adaptation needed to be made to ensure the methodology was suitable for a single developer, the adapted methodology still capture the core concepts of the original specification. Developing features one at a time and having a set work flow for developing those features, helped to schedule tasks, and avoid being overwhelmed by the scale of the project. In addition, as FDD provided an upfront design phase it was far easier to obtain insight into the size of the project, as well as identifying areas that would take longer to implement.

Whilst the methodology was a success overall, it could have been further improved with additional adaptations. Originally, it was planned that pull request would be left open on GitHub for a day, before they were reviewed to check for code quality. This in principal was a good idea but caused problems due to the dependencies between features. The feature list details that almost all features are dependent on the previous feature being complete. As such, delaying the code review for a day would cause a delay in development. This problem could have been resolved however by developing the charades system and the hologram system in tandem. By doing this, it would have been possible to leave time to review one systems code while writing the other (hence not losing any development time).

5.2 Requirements identification

Since the initial creation of the objectives stated in chapter 1 section 1.4.1, some additional considerations have been added. Firstly, there is no mention of security explicitly in the objectives. The justification for this is that neither system produced in this project stores any personal data. Therefore, from a data security aspect, there is no requirement to protect user data. However, there is other security aspects to consider such as SQL injection or Cross-Site Request Forgery (CSRF). These forms of security threat have been addressed in the project although not stated in the objectives and in hind sight, it would have been better to have security explicitly stated as a requirement.

An additional objective that was not stated and should have been added was to ensure that the

Charades Game UI was mobile compatible. The system itself was developed on a 17.3" monitor but rarely checked for how it would appear on a smaller (mobile) device. This would be an area of improvement that could have be made going forward and would greatly benefit the users. By using an adaptive design based on screen size, it would be possible to have a display that would be correct for any size device.

Aside from the above statements, the requirements for the project were well written and helped to focus the project and ensure it did not go out of scope. All the project objectives that were set have been addressed in some way in the project. This system has been well written and has a very high test coverage, which suggests that the system is very robust. The software includes error checking at runtime (in the form of input parsers), as well as logical checks to ensure that any user input is valid and expected. The robustness of this system was confirmed at the Aberystwyth University Science Week 2017, where the system functioned without error for the duration of the event.

5.3 Design decisions

The Charades Game was originally designed as an android application as discussed in the implementation section. This design decision was made due to language and technology familiarity for the developer. After further analysis, during the implementation of the project, the decision to switch to a web based implementation was made. This initially resulted in the loss of two weeks' work as the Android application so far had to be deleted. This decision was overall a positive change, but did potentially lower the quality of the software due to the time that was lost.

Neither the Android application or web application had a full design for the solution to communications. As this technology was relatively unknown to the developer, it was difficult the best method for implementation. Subsequently the communications design was vague. With more investigation into possible technologies for communication in the earlier stages of design, it would have been easier to make a more informed choice of the type of technology to use.

The design diagrams used to inform the overall model (section 2.1) proved very useful in helping to define the system. All the diagrams provided good insight into the requirements and structure of the system and were used throughout to help meet those requirements. Starting with a use case diagram helped to view the system at a very basic level of interaction. It helped to formalise all the operations that would be required of the system and ensured that no functional requirements were missed.

Creating UI designs helped to ensure that functions identified from the Use Case diagram were met. However, the diagrams could have been improved to help with system interaction. Being planned for use at outreach events, the system had to be designed with a young target audience in mind. In this respect, the UI design falls slightly short as the colour scheme could be more vibrant and enticing, and the interactions could have been designed in a way where instructions (displayed on the user and actor landing pages) are not required. Having not been able to test this system with the target audience, it is hard to establish if this is an effective UI.

5.4 Use of Tools

When the hologram system was being developed, it used the notepad++ text editor as the main editor for Python coding. However, when development of the Charades Game began, it required a better IDE that could help when developing code. The PyCharm IDE proved very helpful while developing the Charades System and on reflection would have been an excellent tool to use for development of the hologram system as well. Whilst it is unlikely that this would have further improved the code quality beyond its current state, it would have reached the state faster.

Throughout the project, Git and GitHub was utilised well making it easy to track progress, and organise features. The extensive use of Git along with sensible commit messages and correct issue number referencing made it easy to look back and see what development had been completed already. Furthermore, the use of GitHub templates for issue and pull request creation meant that a check list was already generated for each issue a pull request which contained the tasks to be completed. Although GitHub was an asset to the project, the way in which it was used, at times, was excessive and could get in the way of development. The GitHub workflow catered towards a team slightly more so than an individual. This meant that unnecessary tasks like assigning a developer to an issue or pull request had to be performed despite there only being one developer. Despite this however, Git and GitHub were an invaluable tool for version control, organisation, and backup throughout the project.

Jenkins was used as an environment to run the automated tests for the system as well as perform static analysis. Using a service like Jenkins helped to regulate the test environment and produce an easy to comprehend output. The decision to use Jenkins as the CI tool of choice was made due to past familiarity with the service, however, on reflection, this could have been done locally just as easily. Jenkins has many features that were not utilised in this project such as project builds. Project builds is a tool that creates a build copy of the software after a set of tests have been run in the pipeline - which can be a useful output for manual testing. As this project was written in Python, there was no build output (as Python does not require compiling in a similar way to other languages such as C). Project builds is just one example of how Jenkins could be considered a heavy weight solution to having a controlled testing environment. Whilst it was a success in this project, it could be replaced by a light weight alternative should the project continue.

5.5 Improvements

Keeping a diary throughout the project proved an invaluable resource when writing the report. The dairy helped with managing when development took place, the problems that were discovered and references that needed to be added. Were the project completed again, improving the diary even further would be an asset. As well as writing short bullet points that summarise issues that were dealt with that day, it would have been better to write a small paragraph. These paragraphs could then be used in the report to replace the overview at the start of the iteration. Furthermore, it would be better to reference features by name in the diary so that it can be more easily read without having to reference another document.

To obtain feedback on and improve the UI design, copies of the design could have been taken to the Aberystwyth University Science Week 2017. This would have given the opportunity to see if the target audience liked the design of the UI and what, if anything they might change. Collating these results then could have led to a better design that would be more fitting of the target audience.

5.6 Future work

There are many sections of the project that still require refactoring. Where this has been done already, the code is of very good standard, but sections of the Charades Game are not written as well as they could be. The main place where this is present is in the views.py file. This file holds all the controller functions for every page (including the API) and could quite easily be simplified through refactoring. The stages that would be done for this include, removing duplicate code used in redirects, moving simple code to other functions for both readability and maintainability, and using multiple files instead of just one to hold all the controller functionality.

An idea for a possible piece of future work is to enable the Charades Game and the Hologram creation system to interact with one another. The design for this would turn the camera (pointed at the actor) on when they were acting, and then display messages when either the actor is choosing a new word or phrase, or a viewer has guessed correctly. This could easily be done providing the hologram creation system and the Charades Game server are hosted on the same machine. The hologram creation system could poll a file on the machines hard drive, which contains what the hologram should currently be displaying (e.g. ”Camera”, ”Winner”, ”Waiting”). The website would then be able to update the file when the game state changes.

Currently the website for the Charades Game is not hosted anywhere online. Deploying this system to either a cloud service or on a server would not be a difficult task. The website can be installed and run using the installation instructions bundled with the project in the README.txt file. This would need to be done to use the charades system at future outreach events.

Appendices

Appendix A

Third-Party Code and Libraries

1.1 OpenCV

OpenCV is an open source computer vision library designed to aid in building computer vision systems. The library is original written in C++, but can also be compiled for C, Java, Python and MATLAB. OpenCV offers a wide range of functions and algorithms as well as a way to capture and represent a web cam feed in Python (and other supported languages).

OpenCV version 2.4.8 was used for this project. More information on OpenCV can be found on their website [29].

1.2 NumPy

NumPy is a mathematics library for Python and is a requirement for OpenCV. NumPy was used in this project for matrices that represented the image data obtained from the web cam.

NumPy version 1.8.2 was used for this project. More information can be found on the NumPy web page [30].

1.3 SciPy

SciPy is a scientific Python library that is require for some of the operations in OpenCV. Whilst this project does not explicitly use the library, for continuity this library is also mentioned as it is a dependency for OpenCV, and must be installed to ensure correct functionality.

SciPy version 0.13.3 was used for the project. More information about SciPy can be found on their website [31].

1.4 Django

A web frame work for creating Model View Controller style web applications using the Python programming language. Django works on a request based system where HTTP requests are routed

to the controller which will then query the Model and View to create a rendered web page. This is then display in the client-side browser.

Django version 1.8.3 was used as the web framework to implement the Charades Game within this project. For more information on Django, see the Django developer website [32].

1.5 jQuery

jQuery is a JavaScript library designed to help traverse and manipulate HTML content. This was used for the polling of the API in the Charades Game.

JQuery version 3.2.0 was used for this project. More information can be found on the JQuery web page [33].

1.6 Selenium

Selenium is a browser automation tool normally used for testing websites. Selenium offers the ability to mimic button clicks, text input and most (if not all) other website interactions.

Selenium version 3.4.1 was used for this project. More information can be found on the Selenium web page [34].

1.7 ChromeDriver

ChromeDriver is a tool for automated testing in a Chrome browser environment. It was used in tandem with Selenium during this project for testing the Charades Game.

ChromeDriver version 2.29 was used for this project. More information can be found on the ChromeDriver web page [35].

1.8 Django-Jenkins

django-jenkins is a plugin that captures the output of Django style unit and system tests and parses the output into a format that can be comprehended by Jenkins. This is a Django plugin that was developed by GitHub user kmmbvnr. This plugin is only a server side requirement so will be commented out from the final code base before final hand in of the technical work. Please see the README.txt file bundled with the source code to find out how to enable this functionality.

django-jenkins version 0.110.0 was used for this project. More information on the plugin can be found on the GitHub page [36].

1.9 Jenkins

Jenkins is a continuous integration tool that was used to manage and run tests over the duration of the project.

More information can be found on the Jenkins web page [37].

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

3.1 Embedded Python

The code shown below is embedded Python code taken from the acting.html template of the Charades Game. The code below is designed to change the colour of each word in the current phrase depending on if it has been guessed, it is the current word or it has yet to be used.

- The use of '% %' denote a Python block where the code inside is to be executed upon page load.
- 'forloop.counter' and 'forloop.counter0' are internal counters used to index the current position in the for loop. 'forloop.counter0' is a counter starting from 0 and 'forloop.counter' is a counter starting from 1.

```
<h1 class="page_title">
{% for word in word_list %}
{% if forloop.counter == current_word %}
<span class="current_word_span">
{% elif forloop.counter0 in completed_words %}
<span class="completed_word_span">
{% else %}
<span class="word_span">
{% endif %}
{{word}}</span>
{% endfor %}
</h1>
```

3.2 phrase_ready API

The Charades Game API function that displays if the phrase is currently in a guessable state within the system. The function is from the 'charades/charades/views.py' and is discussed in Chapter 3: Iteration 7.

```
def phrase_ready_api(request):
    """
    Asserts if the phrase is ready. Only intended for api access
    """
    phrase_ready = False
    if GAME is not None:
        if GAME.actor is not None:
            phrase_ready = GAME.actor.phrase_ready()
    return render(request,
           'phrase_ready.html',
           {'phrase_ready' : phrase_ready})
```

3.3 Polling function

The JavaScript and JQuery function used to retrieve data from the guess_correct API. The function is from the file '/charades/templates/guess.html' and discussed in Chapter 3: Iteration 7.

```
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js">
</script>
<script>
setInterval(function(){
    $.get( "/api/guess_correct", function( data ) {
        if (data.includes("Word") || data.includes("Phrase")) {
            window.location.replace('/waiting_for_actor/');
        }
    });
    poll();
}, 1000);
</script>
```

3.4 Unit test

The below code shows the success, edge and failure states that are tested by unit tests. Here 0 is the edge case, 10 is the success case and -1 the failure. This code is taken from the '/python/tests/test_parsers.py' file.

```
class TestPositiveParser(unittest.TestCase):
    """
    @class TestPositiveParser :: Tests that ensure the input parser
      only accepts postive integers

    """
    ###---------------Success cases---------------###
    def test_zero(self):
        self.assertTrue(vpa.parse_positive_int(0))

    def test_positive(self):
        self.assertTrue(vpa.parse_positive_int(10))

    ###-------------Failure cases-----------------###

    def test_negative(self):
        self.assertRaises(ValueError,
                          vpa.parse_positive_int,
                          -1)
```

3.5 context file

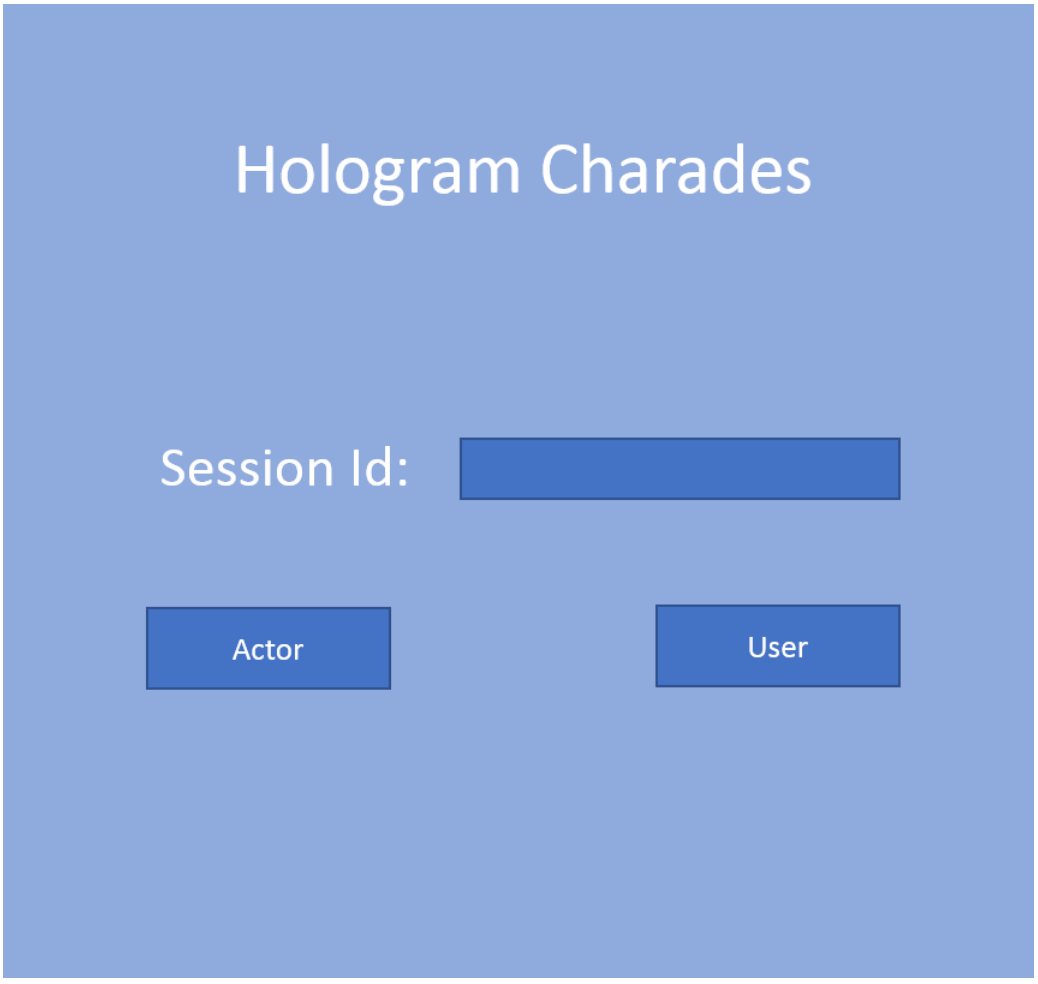
The context.py file is used to set the correct file path to import source code for the project. This allows project code to be executed in the tests. This code is from the file "/python/tests/context.py"

```
"""Allows tests to find the VideoProcessor module for testing
   This is required as the modules are in different directories"""
#pylint: disable=wrong-import-position,unused-import
#   Above pylint disables are required as this is a context file and
#   is only used by other files to set namespace and resovle modules.

import os
import sys
sys.path.insert(0, os.path.abspath(os.path.join(
os.path.dirname(__file__), '..')))

import vpa

#http://docs.python-guide.org/en/latest/writing/structure/
```

Appendix D

UI diagrams

4.1 Index



The landing page for the Charades Game.

4.2 Instructions

The instruction screen for the actor and viewer.

4.3 Select_phrase

The selection screen for the actor to chose a phrase to act.

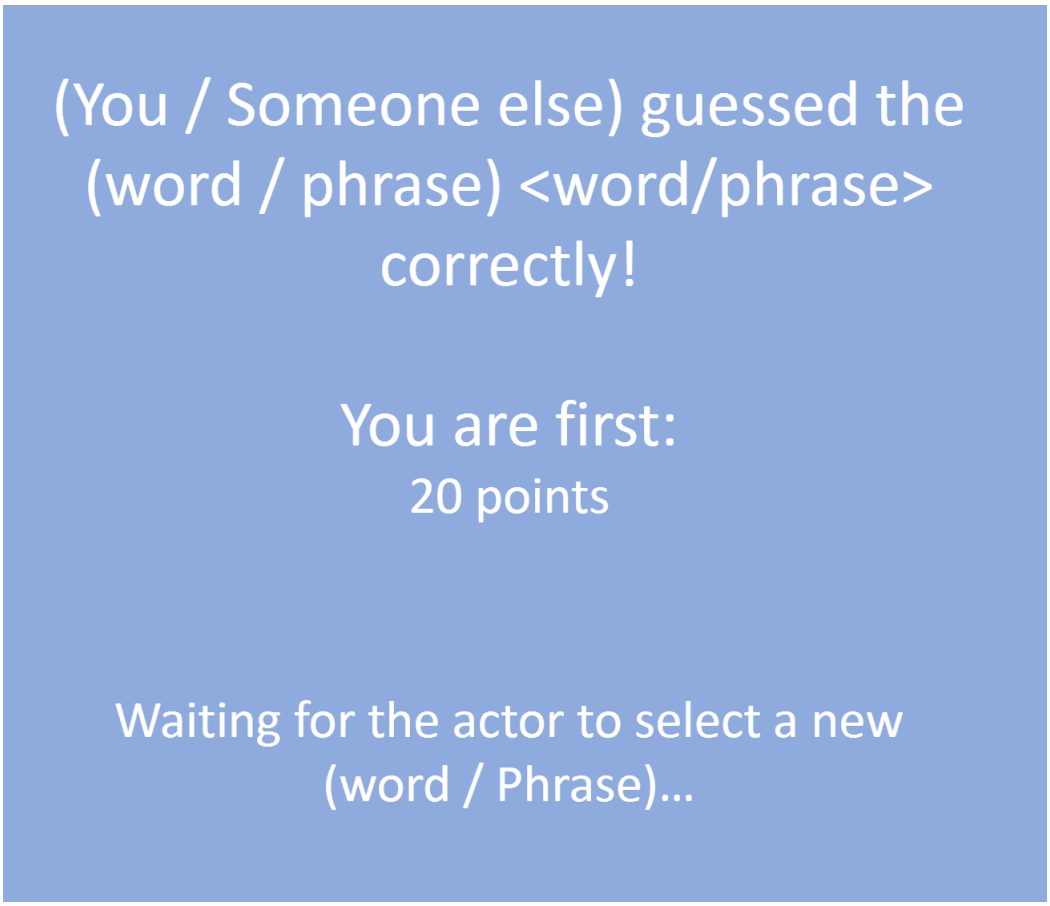
4.4 Acting_single_word

The acting screen for when there is only one word in the phrase.

4.5 Acting_multi_word

The actors word selection screen when the phrase is comprised of multiple words. Green is if the word is complete. Yellow if for the currently selected word. Blue/white is unselected words.

4.6 Waiting_for_actor



The page that is displayed while the viewer waits for the actor to select an new word or phrase.

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] Android Development Team, "Hiding the Status Bar," https://developer.android.com/training/system-ui/status.html, 2016.

A tutorial included in the documentation on the Android developers site that discusses managing the System UI, in particular, the removal of the status bar from the header of an Android app.

[3] 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.

[4] 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.

[5] M. Nunez, "French Presidential Candidate Jean-Luc Melenchon's 'Hologram' Is Not Actually a Hologram," http://gizmodo.com/french-presidential-candidate-jean-luc-melenchons-holog-1794450283, 2017.

An article describing how the Pepper's Ghost technique was used during the french presidential elections in 2017 to give campaign speechs in multiple locations at once.

[6] 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.

[7] 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.

[8] Alexander Mordvintsev, "Capture Video from Camera," http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_gui/py_video_display/py_video_display.html, 2013.

A tutorial for obtaing the output images from a webcam using python and openCV.

[9] net-informations, "Explanation of Deep and Shallow Copying," http://net-informations.com/faq/net/shallow-deep-copy.htm, 2016.

An explaination of the difference between deep copying and shallow copying in computer science. Whilst this reference describes the differnece using the .net programming language, the concept applies to other languages as well.

[10] Yannick Benezeth, Pierre-Marc Jodoin, Bruno Emile, Helene Laurent, Christophe Rosenberger, "Comparative study of background subtraction algorithms," Journal of Electronic Imaging, Society of Photo-optical Instrumentation Engineers,, 2010.

A summary of multiple background subtraction techniques and their effectiveness given different types of data. The Article includes a variety of techniques ranging from the simple to the complex and many of the techniques are already implemented in the OpenCV module.

[11] Book Trust, "The Ultimate List: Our 100 best children's books," http://www.booktrust.org.uk/news-and-blogs/news/222/, 2013.

Contains a list of the top 100 books in certain age brackets.

[12] Sara Schmidt, "15 Best Kids Movies Of 2016," http://screenrant.com/best-kids-movies-of-2016-zootopia-moana-sing/?view=all, 2016.

The top childrens movies that came out in 2016.

[13] Android Development Team, "Android: Getting Started," https://developer.android.com/training/index.html, 2016.

An initial hello world style android project guide provided by the creators of the Android Studio IDE.

[14] User: ViVekH, "Hiding the Status Bar," http://stackoverflow.com/questions/28144657/android-error-attempt-to-invoke-virtual-method-void-android-app-actionbar-on, 2016.

A stack overflow post describing the incorrect use of the getActionBar() method when attempting to obtain a pointer to the status bar in an adroid app using support.v7.app.ActionBarActivity.

[15] Android Development Team, "Getting Started with Testing," https://developer.android.com/training/testing/start/index.html, 2016.

A tutorial included in the documentation on the Android developers site that discusses how to create automated tests for an android application. The tutorial details the types of testing as well as how to implement them.

[16] User: jacktrades, "Test if a button starts a new Activity in android JUnit," http://stackoverflow.com/questions/9405561/test-if-a-button-starts-a-new-activity-in-android-junit-pref-without-robotium, 2012.

A stack overflow post describing the how button transistions between pages could be tested using Instrumented testing. The post details how an ActvitiyMonitor can be used to listen to the app and check for changes when they occur.

[17] R. Brown, "Django vs Flask vs Pyramid: Choosing a Python Web Framework," https://www.airpair.com/python/posts/django-flask-pyramid#6-frameworks-in-action, 2016.

An online article that details and compares the differences between three of the most popular python web frameworks, Django, Flask and Pyramid. This article was used to help make the decision as to which would be used for the project.

[18] N. George, "The Django Book," http://djangobook.com/, 2016.

A online copy of a book that teaches Django concepts and implementation. The book teaches how Django can be used and how it works giving examples throughout. The book was used to help build the website for the Charades Game.

[19] D. D. team, "How to use sessions," https://docs.djangoproject.com/en/1.11/topics/http/sessions/.

A section of the online documentation produced by the developers of the Django framework which details the implementation of sessions using cookies.

[20] J. Kaplan-Moss, "Real-Time Django Is Here: Get Started with Django Channels," https://blog.heroku.com/in_deep_with_django_channels_the_future_of_real_time_apps_in_django, 2016.

An online blog describing Django Channels, how they work and how to implement them.

[21] W. Schools, "Window setInterval() Method," https://www.w3schools.com/jsref/met_win_setinterval.asp, 2017.

The W3 Schools webpage explaining the javascript setInterval function. This function was used in the project to poll the API.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.)

[28] 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.

[29] OpenCV Team, "OpenCV," http://www.opencv.org/, 2017.

The home page for the OPenCV software library.

[30] NumPy developers, "NumPy," http://www.numpy.org/, 2017.

The home page for the NumPy mathematical Python library.

[31] SciPy Developers, "SciPy," https://www.scipy.org/, 2017.

The home page for the SciPy scientific Python library.

[32] Django Software Foundation, "Django," https://www.djangoproject.com/, 2017.

The home page for the Django Python web framework.

[33] The jQuery Foundation, "jQuery," https://jquery.com/, 2017.

The home page for the jQuery JavaScript library.

[34] Selenium development Team, “Selenium HQ,” http://www.seleniumhq.org/, 2017.

The home page for the Selenium browser automation library.

[35] Chromium and WebDriver development teams, “ChromeDriver,” https://sites.google.com/a/chromium.org/chromedriver/, 2017.

The home page for the Chrome Driver library.

[36] GitHub user: kmmbvnr, “django-jenkins,” https://github.com/kmmbvnr/django-jenkins#egg=django-jenkins, 2017.

The GitHub page for the django-jekins Django plugin.

[37] J. development Team, “Jenkins,” https://jenkins.io/, 2017.

The home page for the Jenkins continuous integration service.