MOVING PORTRAIT

To create and understand how users interpret and behave, in response to moving portraits reactions to movements and gestures.

UNIVERSITY OF
LINCOLN

James Bruce
BRU14476346
Computer Science
Project Supervisor: John Shearer
University of Lincoln, School of Computer Science
27th April, 2017

Introduction **3**

Aim 4

Objectives 4

Literature Review of related work and context **5**

Previous work/art 5

Software review / tools 7

OpenCV 8

Methodology **9**

Project management 9

Evaluation/testing methods 10

Design and Development **12**

Requirements 12

Software and Hardware Design 12

Implementation **13**

Introduction 13

Basic button press with waving 13

Basic motion detection 14

BUG/performance problem 14

Testing **17**

Deployment 17

Evaluation **18**

Methods 18

Deployment and Recruitment 18

results 19

Face Detection 19

Motion Detection 20

Analysis 21

Ethical Procedures 21

Conclusion **22**

Reflective Analysis **23**

References **26**

Introduction

With the Harry Potter books selling over 450 million copies, the movies making 6.8 billion dollars total worldwide box office, and with a spin off series, we can see that the series is a global phenomenon. The universe of Harry Potter explores magics such as charms, curses, potions and most importantly, moving portraits.

(Harry Potter: Prisoner of Azkaban., 2004)

Explained in the series, the magical moving portraits of Harry Potter are made possible through wizards placing enchantments on paintings they create, the more movement and life in the painting, the more powerful the wizard. Thanks to advancements in technology, magic in the Harry Potter universe is now possible in the real world, with the use of cameras and lots of pre-recorded footage.

Taking inspiration from the Harry Potter moving portrait, this project is going to take this idea and develop it into a physical artefact that shows a character that can recognise and respond to movements and gestures of users interacting with it. The project will be based around the integration of media (art and technology) and the creative development of them.

There are a few reasons for developing this idea into something real. The most important reason for this project is taking something that is magic in the Harry Potter universe and develop it into something in the real world. It would have been possible to create this idea in the past, but it would have taken a lot more time and effort than it does now. Advancements such as the microcomputer Raspberry Pi (rPi) and libraries such as OpenCV (OpenCV, 2000) have helped make this goal achievable . OpenCV is an open source computer vision software library, meaning that I can present a gesture and associate it with a response. The subject area that the project covers is mostly based around entertainment. The artefact is a creative media, unifying art and technology in an interesting way.

The overall background for the project, being from the United Kingdom I saw the connection between Harry Potter and the idea of a moving portrait. What I wanted to achieve with this in mind, was to use technology to emulate magic. The following document discusses the processes I went through creating my artefact of the moving portrait, starting with the aims and objectives of the development followed by the literature that I made use of, the steps I took in terms of design and development and finishing with an evaluation of my artefact.

Aim

To create and understand how users interpret and behave, in response to moving portraits reactions to movements and gestures.

Objectives

1. Build a Physical implementation with camera functionality
    - This step will make use of a raspberry Pi, once the device is set up Android will be installed to give an easy way to test, implement and improve the artefact quickly and easily.

2. Recording Actor(s) as the portrait, with movements and gestures
    - As for the recordings of actors, it would need to cover most common gestures, and keeping a balance between enough movements to look real but not too much to make it appear fidgety. A plan for each movement has been made for this as to not forget a gesture, if a gesture is forgotten then it can be hard to re-record since the lighting and stage need to be the exact same as the other clips, so that it transitions smoothly.

3. Make use of OpenCV to detect movement and faces to give a response from the portrait
    - Once familiar with OpenCV, it will be used to create responses to movement and gestures presented in front of the device.

4. Develop a program that can make use of the hardware built, that displays a portrait, that then moves and responds to gestures detected in the OpenCV library
    - Once the previous objectives are complete (and to a good standard), they will be compiled into a finished product that takes a user's gesture or movement and responds to it smoothly and realistically. The camera will display an input that then passes to OpenCV, if a gesture is found in the library then the program will respond with the correct gesture.

5. Testing of the Artefact and improving where necessary
    - Once I have a finished product I will then go about testing it with various methods for each function that it has, such as what happens if the user gestures again quickly or if there are 2 gestures at once. This will be a constant cycle at the end of development up until the deadline of completion in order to prevent unnecessary problems and easy to fix errors.

6. After the development process is complete, evaluate the artefact and gain user feedback
    - This is an important objective, as it revolves around putting the device to use, and user feedback is important for this
    - I will be able to improve my artefact with this feedback, because users may pick up on something that I am oblivious to.

From these aims and objectives the final goal of the artefact is to create a moving portrait, that reacts to users actions.

Literature Review of related work and context

In this section I am going to discuss the materials that I made use of during the stages of creating my artefact. Previous works that I had inspiration from as well as the tools and software that I made use of. I Will also discuss my usage of OpenCV and the other features of the software library.

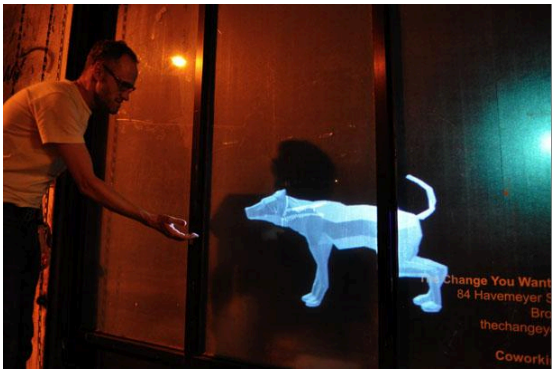
Previous work/art

Previously, some ideas were developed based around the same sort of idea; Good Boy Sammy (Perrone, 2016) and Sniff (George, 2012) took the same approach, but executed in different ways to each other.

Good Boy Sammy has a slightly morbid background to it, Perrone used footage of her dog that had died, in order to bring him back to life. Good Boy Sammy projects an image onto glass and using voice detection to get the device to respond as if the dog is alive and responding. This is very similar to the way that the Moving Portrait artefact works, only instead of voice, it makes use of movement and face detection. The main developmental process that Perrone went through were very similar to mine, they began with taking footage of a subject, and separating it into movements then attaching these movements to sounds, for me it was gestures.

(Perrone, 2016)

Sniff is an art project that makes use of a store window to project an animated dog, to give the appearance that it is moving around the room in 3D space. Sniff makes use of motion detection to let the device know where the user is standing. This is comparable to my artefact as I am going to be making use of motion detection as a method of activation.



(George, 2012)

Force of nature (FIELD and NIKE, 2015) was another technology based art piece. The product takes the idea of *"An interactive running experience that transforms you into a Force of Nature"*(FIELD and NIKE, 2015). As the user runs, their movements get projected onto a screen in front of them. Using the Kinect and a treadmill fitted with custom sensors, the installation takes the runner through a journey of immersive visual effects that amplify the feeling of getting into the flow of running. This can relate back to my artefact since it makes use of the user's movements, which is something that I aimed to do, in order to give back an output from a piece of software.

(FIELD and NIKE, 2015)

'Understanding the experience of interactive art'(Costello et al., 2005), the paper describes a study into the situated experience of interactive art, The study was conducted with audiences of the artwork Iamascope. The Iamascope(Fels, 2002). The Iamascope makes use of computer video graphics, vision and audio in order to create a kaleidoscope using the user's body and the movements that they make, the more they move the more sound is generated. The study looks into users and the experience that they have with interactive art, in conjunction with Beta_Space(Research.it.uts.edu.au, 2017). Beta_Space is an experimental environment where the public can engage with the latest research in art and technology. The exhibit is located at the University of Technology, Sydney and displays interactive artworks, the works presented can be in any stage of development, from beta to a finished product.

(Fels, 2002)

Beta_Space was a good find for the artefact that I developed since I would consider it a hub of all things connected to art and technology. This helped with generating ideas on how I could create my moving portrait, since similar ideas had been created previously by people taking part in the exhibit.

Software review / tools

GitHub was a useful resource for gathering literature to use for the Project. The repository 'Android-motion-detection'(Gehring, 2017) became useful for developing motion detection. The repository made use of another developer's work, 'Android-motion-detection' (Wetherell, 2017). Wetherell set the ideas in motion, demonstrating how movement capture can work using a camera. The code takes a picture every few seconds using the camera, then compresses the pixel values down, so that there is less to check against. It then compares to an image taken previously, If the images are different then motion has been detected, if it is the same then no motion has been detected. Wetherell's repository of code made use of various HTML files, making up around 90.8% of the total repository. Gehring developed upon this code, making it easier to call the functions into another class, also removing all the unnecessary HTML code, leaving behind just the Java, which was made use of in the project development. One of the main strengths of these repositories is that they work effectively with space management, for example the picture the camera takes is not in a high resolution, only 2 images are saved and compared at a time. This is useful, since the Portrait Project is developed on Android, which tends to not have a lot of space (around 60gb), compared to something like a computer (1TB).

Another set of repositories used consisted of multiple branches of developers. The HandGestureApp (Qin, 2017) takes the output of the camera, the user then selects where their hand is, so it can then find where the fingers are and then count them. This is a well-structured way of finding the hand, since it prevents the app from getting confused as to where the hand is in the image, since the user selects where it is. Another developer made a similar implementation of hand detection, Hand_finger_recognition_android(Azipurua, 2017). Azipurua demonstrated the idea differently, taking the entire shape of the hand rather than separate fingers from the hand and then counting the contours. A problem with this is that it can sometimes struggle to count fingers if they are not spaced apart. Another problem that the code had was that it was unreliable at counting 0 fingers, the 0 would sometimes be counted as 1. This is a limitation on the code usage, as an error like this can cause gestures to get mixed up and detect the hand incorrectly. OpenCV_HandFingerRecognition (Hery, 2017), forked the repository developed by A.Héctor, upgrading it to a newer OpenCV version and Android SDK version. The repository was not used in the development of the artefact but it helped to give a better understanding to how OpenCV can be used to develop a gesture/face detection application on Android.

The version of Android that I opted for was API version 21 (Android 5.0,LOLLIPOP), the reason for this was because I have used this version in the past. Another reason for opting for API 21 was because it is compatible with most Android devices, this is useful since the version of Android that I had working on the Raspberry Pi can run this version of Android applications. API 21 also worked with openCV

Some additional software that I made use of was Win32Diskimager (Disk Imager, 2016). I used this to image files onto an SD card so the Raspberry Pi could read them. SDFormatter (Sdcard.org, 2016) was another piece of software that I used, this was for cleaning the SD card in preparation for imaging Android onto it.

The Android image that I made use of was Android 5.1 Lollipop (Exton's International Blog, 2017), Lollipop is the most common version of Android and is used on most Android supporting devices. On the Pi it runs just the same as it would on a phone only it has a few issues with input lag and crashes.

OpenCV

OpenCV is a computer vision library that is written using C and C++. The library can run on operating systems such as Linux, Windows and Mac OS X. The library supports other languages, a few examples are Python, Ruby, Matlab, Java. The main objective of OpenCV is to provide a simple to use computer vision infrastructure that helps developers to build fairly complex vision applications quickly (Bradski and Kaehler, 2011). The OpenCV library contains more than 500 functions, spanning all areas of computer vision. OpenCV has been used in product inspection, medical imaging, security and robotics. Computer vision is the "*automatic extraction, analysis and understanding of useful information from a single image or a sequence of images"*(Bmva.org, 2017). Computer vision is based around taking an input image/video and manipulating it in a way that data can be extracted as needed. Some examples of this are; background subtraction, colour detection, face detection. OpenCV has many other uses, however these are the ones I chose to research.

Background Subtraction

Background subtraction is a technique used when an object of importance needs to be found in an image such as a human. Background subtraction takes a previous static image and then compares it to the current one, new objects will be revealed where as everything that didn't move will not appear. This could have been a useful implementation method for my artefact, it presents a visually clear way of seeing a user.

Colour detection

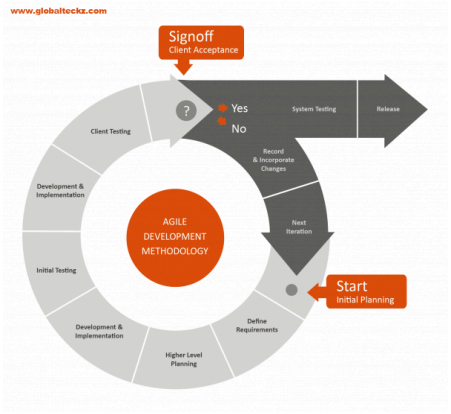
Colour detection takes an image and the colour pixel values of it, and then removes all of those that do not fit the desired colour threshold. This is used to detect objects based on their colour. I could have used this for my implementation as I could have taken a colour threshold that matches skin and returned the object that would be found, the user. One of the main problems with this is, how do I measure if there is a user in frame. Adding the pixels together could be a way of doing it but what if the user is wearing clothes that cover a lot of skin. They may not get detected by the artefact.

Face detection

Face detection is another method of OpenCV useage. Face detection works by taking the basic shapes of the face (eyes, nose, mouth and chin) and tries to find an object in the image that matches. Face detection is useful for the artefact since everyone has a face, and they tend to have the same features as one another.

Methodology

The main development methodology used in the Moving portrait cycle was Agile Development. The Agile methodology works by having multiple development cycles, as the development goes through multiple iterations.



(GlobalTeckz, 2017)

Project management

The reason I chose Agile as my methodology was because the artefact was to go through multiple iterations, from a basic button press to movement detection and then face detection. I also picked Agile over Waterfall because I can keep improving areas that I feel are lacking, whereas using Waterfall would mean being unable to go back on myself in terms of the development stages. Once a step is done in waterfall, there is no going back. Compared to waterfall, this seemed more appropriate since Waterfall is based around the idea of completely finishing a development step and never going back to it. Since my artefact was to go through multiple development cycles, Waterfall was not as suitable. There are many advantages to using Agile development;

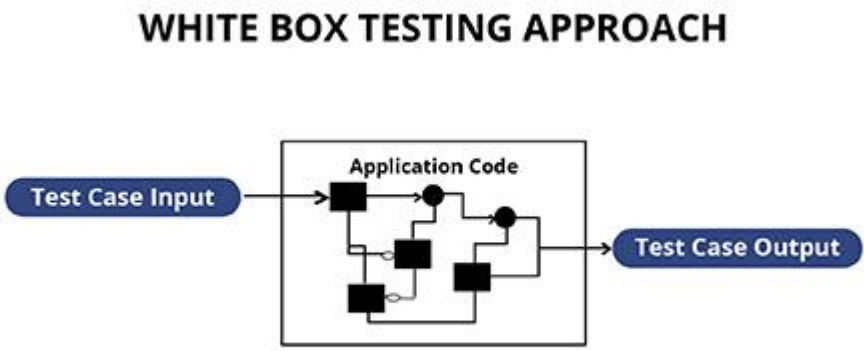
- Testing is integrated throughout the lifecycle, enabling regular inspection of the working product as it develops. This allows the product owner to make adjustments.
- Encourage active 'user' involvement throughout the product's development. This allowed for me to show others my artefact, then make improvements based on their feedback.
- In agile development, change is accepted. Sometimes an extra feature may be needed, or something may need to be taken out, Agile accommodates for this(Allaboutagile.com, 2017).

Disadvantages to Agile development;

- In case of some software deliverables, especially the large ones, it is difficult to assess the effort required at the beginning of the software development life cycle.
- There is lack of emphasis on necessary designing and documentation.
- The project can easily get taken off track if the customer representative is not clear what final outcome that they want (ISTQB, 2017).

Evaluation/testing methods

The testing method that I made use of was White box Testing. White box testing is a software testing method which the internal structure of the item being tested is known to the tester. The tester chooses the inputs and paths through the code to exercise and determines the appropriate outputs. White box testing is testing beyond the user interface and focuses on the internals of a system.



(Arvind Rongala, 2017)

Advantages to white box testing;

- Code optimization by revealing hidden errors
- Transparency of the internal coding structure which is helpful in deriving the type of input data needed to test an application effectively
- Covers all possible paths of a code thereby, empowering a software engineering team to conduct thorough application testing
- Enables programmer to introspect because developers can carefully describe any new implementation

Disadvantages to white box testing;

- Since tests can be very complex, highly skilled resources are required, with thorough knowledge of programming and implementation.
- Test script maintenance can be a burden if the implementation changes too frequently.
- Since this method of testing is closely tied with the application being testing, tools to cater to every kind of implementation/platform may not be readily available.

(Arvind Rongala, 2017)

Blackbox testing evaluates code without looking at the internal code structure. Black box testing focuses on inputs and outputs of the software system, without knowledge of the developed code. This method would be harder for me to do personally, since I had developed the code myself and I know how things should work.

(Guru99.com, 2017)

Advantages to black box testing;

- Unbiased tests because the designer and tester work independently
- Tester is free from any pressure of knowledge of specific programming languages to test the reliability and functionality of an application / software
- Facilitates identification of contradictions and vagueness in functional specifications
- Test is performed from a user's point-of-view and not of the designer's

Disadvantages to black box testing;

- Tests can be redundant if already run by the software designer
- Test cases are extremely difficult to be designed without clear and concise specifications
- Testing every possible input stream is not possible because it is time-consuming and this would eventually leave many program paths untested
- Results might be overestimated at times

(Arvind Rongala, 2017)

From the information that I have gathered about whitebox and blackbox testing, whitebox testing was the better choice. One of the reasons for this is that it would be a lot easier to do, black box testing would require an assessor that has no knowledge of the system and how it works. White box testing does not require this and can be done by the developer, meaning I can test at any time that the artefact may require it. Also white box testing covers all possible paths of code, meaning I can test all of the functions thoroughly, in the order I want them to be used.

Design and Development

Requirements

At the start of the design and development process the expectation and goals of the project are defined. This is a plan for what I want to achieve at the end of the development cycle. With this moving portrait idea, I wanted to create a physical implementation that can be easily set up as an art installation . The project required features that were capable of detecting users movements, and responding to them in a specific way. These features were important to the project as they give it life, since without it the project would just be a randomised video clip. What I wanted from the application was to feel as though it can respond to a user when they are detected.

Something that I wanted at the core of my development was the use of Android, and Android studio(Android studio, 2017), the reason for this was because I had used the software in the past so I had become familiar with how it worked. Another reason for using Android was because it became easy to test and deploy, for example once I had made an application I could publish it to my phone with the use of a USB connection.

Hardware that I wanted to make use of was the Raspberry Pi for a few reasons, the Pi is portable and easy to set up. This makes it better for me from a presentation standpoint, all I really need is a monitor to plug the device into and then play. I can make the application open on startup, meaning the device would not need a keyboard and mouse, just the camera peripheral.

Software and Hardware Design

After the requirements and goals are defined, work commences on designing the product. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture.

The idea of running it on a Raspberry Pi seemed to be the best approach since the Pi is a portable device designed to be easy to move and take up little space. I opted for the most up to date version of the Pi, the Raspberry Pi 3. With this version there are a few peripherals that come as a part of the board such as Bluetooth and Wi-Fi. The Wi-Fi is important since it makes it easier to connect to the internet and update. there were many advantages for using the Pi such as being able to image Android onto it, this was an important requirement since I built my idea using Android Studio, and with this I would not need to use a phone in order to present the app. Using Android also provided the advantage of additional peripherals that were need to test with, plus current Android phones have cameras meaning little to no set up

I chose Android and Android Studio for the bulk of my program. Android is built upon the usage of Java as its main code base. One of the reasons why I picked Android was because I have experience with it in the past. Android studio gives me the ability to look at what I create in real time, with visual feedback from the program. Java also makes use of the philosophy “Write Once Run Anywhere”(Janarthanan .S, 2017), meaning if I wanted to, I could take the code that I make in Android and make it into a Java application that can run elsewhere, such as on Windows or Ubuntu.

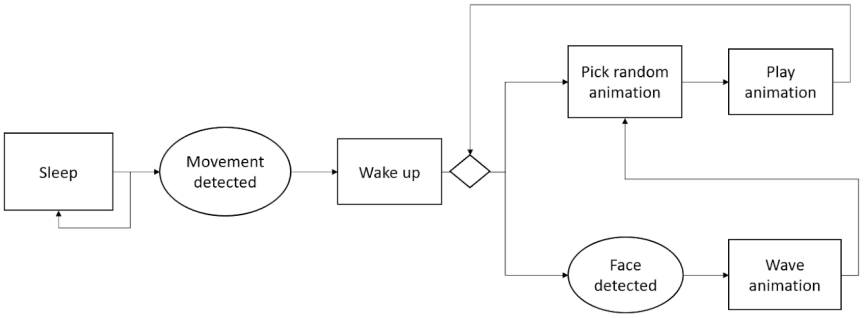
OpenCV was chosen for face and gesture detection since the library was already set in place, meaning as time went on I became more familiar with the program since another module was running that made use of the library. It was not until later on that I realized that Android made use of OpenCV in a different way to how I had used it in the past. Previously it was used in Python, where as this version of OpenCV made use of the language C++. I also did not realise was that OpenCV used C++ with Android at all, it did not become apparent until I followed some tutorials and he pointed out how to make use of C++ Native and how to change all the config files to allow it. This was a setback since this was another component to my program that I needed to learn how to use.

Implementation

Once the design stage is completed, coding of the artefact begins. This stage is the longest phase of the software development cycle.

Introduction

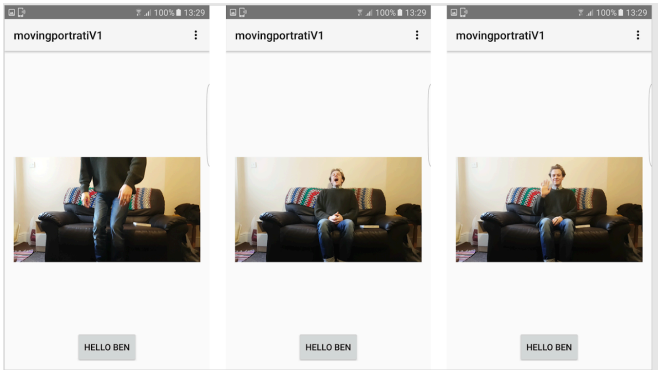
The implementation stage is where I spent most of my time. Since I was quite familiar with Android Studio and Java, it did not take long for me to figure out a starting point. Working with Android Studio, a lot of the UI design and implementation is done for you, it is mostly drag and drop the components that you need. Then using Java pass values to and from those added components. Below is a flow diagram illustrating how the final implementation will function.



(Bruce, 2017)

Basic button press with waving

To begin with, the development of my artefact started with creating a minimum viable product. Using features built into Android and Android Studio I developed a program that played videoclips at random, with a button that would play the wave animation when pressed. This is what I aimed to create at the end of development. Instead of using a button press to trigger a gesture, the function will activate when a motion/face is detected.

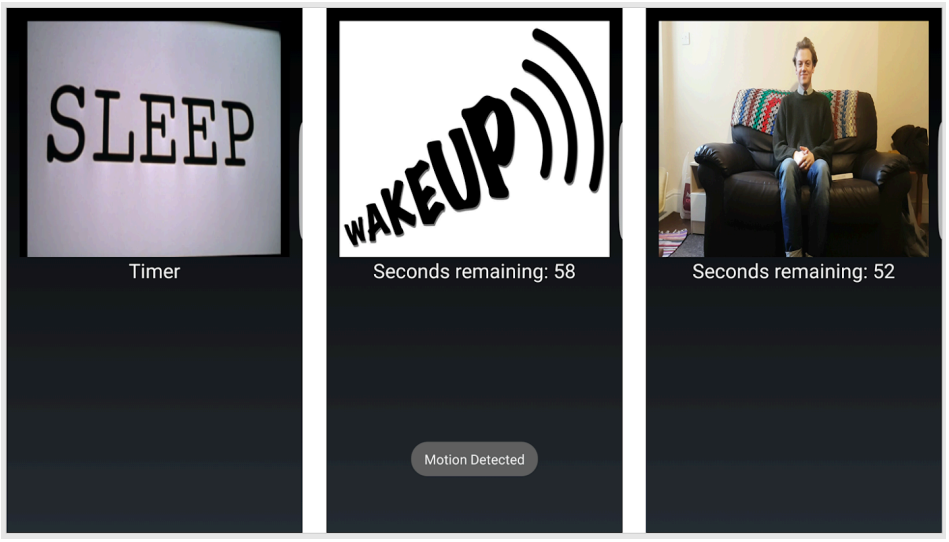


(Bruce, 2017)

From the screenshots above, you can see the basic ideas of the moving portrait. When the application loads, the actor comes into frame and sits down, from there the animations cycle randomly, until the application closes, or the user presses the ‘Hello Ben’ button. Once the button is pressed the wave animation is played, when the clip is finished, the function for random videos is then called.

Basic motion detection

Once I had a basic Program, I started to develop one of the basic functions. First, I implemented motion detection, making use of a GitHub repository, 'Android-motion-detection'(Gehring, 2017). The repository had multiple Java class files built to implement motion detection into any Android application. The class files only need to be added to the project files and then called accordingly to work. This helped me get the basic principles of the moving portrait idea and make basic implementation.



(Bruce, 2017)

From the screenshots shown above, the basics of motion detection is presented. in the first 2 slides, the words sleep and wakeup are pictured, in the final development these would be video clips of the actor, these videos where something that I thought about adding later on in the development process. My reason for adding these is that I thought about what the application will be doing when no one is in front of it, rather than the screen be blank, something could be happening. With those clips in mind, the flow of the application would be a lot smoother and transition better. The motion detection essentially wakes the application up and begins cycling the character animations, once time has passed the application will restart, preparing for the next user.

BUG/performance problem

Originally, I defined gesture detection as a main method of interaction, but a problem with this was that it only really worked well when placed in front of a white background, meaning it would need specific background and lighting conditions in order to work well. What I wanted to achieve was to count each finger on the hand, then give a valid response on how many there were. For example, 5 fingers would be considered a wave, 2 fingers in the middle, a peace sign, 2 fingers on the outside, rocker sign, and 1 in the middle, for something offensive. This only worked when the hand was right in front of the camera so it would not prove effective in a real scenario. The scenario I had in mind was to set up in a hallway, in a way that users will be walking through the vision of the artefact, and trigger when the conditions are met. The reason for changing the hand detection to something else is that the hand can be difficult to detect when the device does not have a lot of time to do so.

Face detection with OpenCV

OpenCV was probably my biggest weakness in this stage, I had little to no experience prior to using it and it took some time to become familiar with how it worked. The first problem came when trying to setup OpenCV with Android Studio was figuring out how to make the library a dependency. There were guides online but a lot of them did not go into too much detail on how to do all of the steps, making OpenCV into a dependency required some folder manipulation and directory changes. Once I managed to get OpenCV working with Android, I did not anticipate that it needed C++ Native NDK to function, this delayed me for some time until I managed to find another guide, a video series on YouTube, Dtdzung (Dtdzung, 2016), that taught everything related to Android Studio and OpenCV. I went through a few of these tutorials, beginning with set up, then passing values from C++ to Java, and finishing with face detection. If it was not for these tutorials it would have taken much more development time to implement OpenCV with my program, possibly resulting in a change of method.

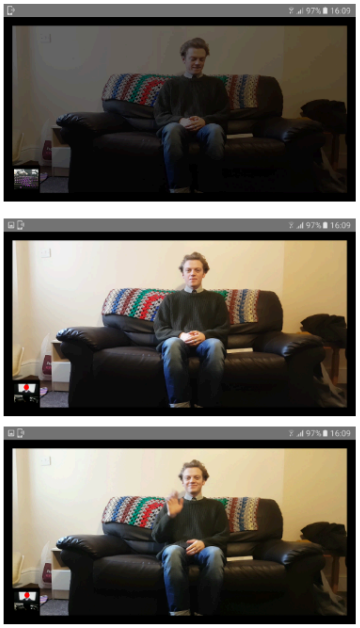
Once I had become familiar with OpenCV development began to get a lot easier. I made use of the tutorials that I followed from the Youtube account Dtdzung and worked them into my final implementation. The artefact made use of face detection, and reacted based on time, rather than specific scenarios. For example I intended for my application to detect movement, which would then wake the character up, once woken the program could then trigger gestures. This was to keep a clean flow to the application, since without this then it could jump around from being asleep, to doing a gesture with no wake up animation in between. In the end I changed the way that I wanted a person to be detected to face detection because I felt like it achieved the same thing plus I had a working prototype that made use of this method. The way that it works now is, it detects a face (as if a user has taken a look at the screen/camera) then the character will wake up, if they remain looking at the screen then it will recognise that the person is not just passing by but stopped and looked, triggering the wave animation. I also ensured that the program would not execute the same wave over and over again, once wave has been called, it does not go back to that point again until the program restarts. The way the program restarts is, by waiting until a face has been out of frame for longer than a minute or so then restarts the activity, putting it back to before wave is called and before it wakes up.

Another implementation that I played around with was to use skin detection, as the colour of skin is easy to set apart from everything else. My reason for not going with this was because it would require a more complicated setup than what face detection. The way that it would work by measuring how much ‘skin’ is on the screen and then figure out if there is enough to be a person there. Some problems with this were that if the user was wearing gloves or a long-sleeved top, there would be less skin meaning it would be harder to detect a human.

One way that I thought of doing the user detection was via movement, for example, when a user moved from one side of the screen to the other, the program would recognise that as someone passing by, therefore there is no reason to respond. If there was movement and it stopped in frame, then this could mean that the user had stopped to look at the device, prompting a response. My reason for not going with this method was the inherent flaw in the idea, what if someone stops in frame, but are not looking in the right direction, causing the app to wake up, but the user would not see it. Also with this method, what if the background changes or something gets moved, it might recognise that as a person and constantly loop through the same part until the frame gets reset. Also, what if multiple people pass by, how could it respond if one person passes and the other stays in frame.

I started designing the idea with gesture detection in mind at first, but as the development stage went on, I realised how difficult that can be for someone of my skill level. I looked at a few examples that users had created making use of OpenCV and hand gestures. Most of them made use of image thresholding then counting the indices, which then can be used to calculate how many fingers there

are. What I realized from their examples is that this idea only really works up close with the hand in frame, with my artefact users would not really get close enough to make use of this feature, and would then cause the portrait to not work as I intended. If this did work, then I could have made a wider array of responses to the users inputs, since multiple conditions could change with the user's



hand.

(Bruce, 2017)

From the screenshots above you can see the basic functionality of the moving portrait. The actor is asleep to begin with, but once a face is detected the actor then wakes up, if the face remains in frame then the actor waves at them.

Testing

After the code is developed it is tested against the requirements to make sure that the product is actually solving the needs addressed and gathered during the requirements phase.

I made use of Whitebox testing, mentioned previously. Once a implementation cycle was over testing began on that iteration. Testing was ongoing throughout development of the artefact. one way that I tested the device was to set it up at home, and test everything using a White Box method, making sure to cover motion detection, from different movement locations (Towards, left to right, right to left and away from the device). I also tested for face detection, using the same kind of movements. What I was looking for from this was if the response animations conflicted at all, and if the right one played for each method.

Once I was happy with this, I then fixed any errors that were found. The first error found when the app would start and wait for a movement, it would only detect a person if the sleep video clip was at the end, before it began to loop again. This was a minor error that did not take long to fix, all it required was to take the function calls out of an onCompleteListener. The onCompleteListener waits for the video to finish, before going onto the next fuction of the app.

Testing Android on the Raspberry Pi was difficult to do. Most current Android images that can run on Raspberry Pi have proven to be slow and buggy(Androidpi.wikia.com, 2017). Android requires hardware acceleration, from looking at developer's comments, this looks to be difficult to implement. This could be a huge problem for my artefact therefore, I opted for running the application on a standard Android phone. This was because the support was there, and I could not really change the programing language to accommodate for the Pi lacking compatibility, as it was too far in development.

Deployment

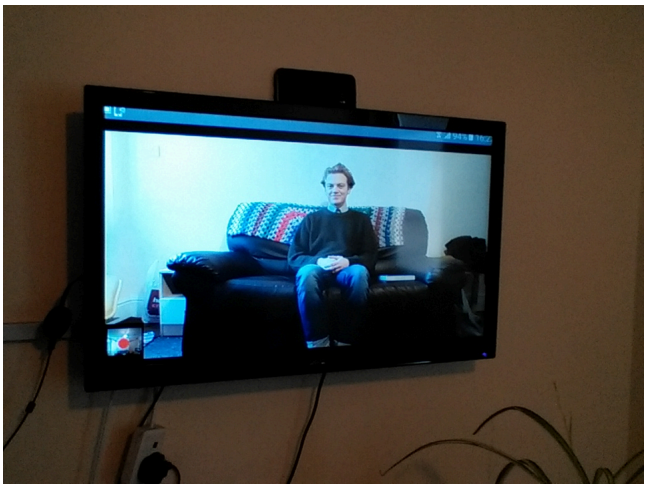
My method of deployment consisted of making a physical implementation that I can set up and run efficiently. I intended to deploy my artifact in the University of Lincoln in an empty office. This is the scenario that I have prepared the artefact for, at the side of a long corridor, where lots of people will walk past.

Advantages of scenario:

- Artefact is secure in the room
- Hallway is Well lit
- Clear threshold between background and users
- Difficult to tamper with device
- Chance for lots of interaction
- Less interference from other objects

Disadvantages of scenario:

- Difficult to get user attention
- Users can walk by before the artefact begins to react
- Requires the user to stand fairly close to detect their face



(Bruce, 2017)

Pictured above is an example of how the device will be set up, with the camera resting above the screen so so faces can be picked up easily, as opposed to being below the screen. From this set up I found that it was fairly easy for the device to pick up faces, with little to no errors. The way that the phone is connected to the screen is via Chromecast (Google, 2017), by casting the Android phone screen to the device. This makes for an easy set up since the phone can be placed anywhere, just as long as it is on the same network as the Chromecast. A limitation to this method is that the Chromecast can not connect to certain network connection security types, those that require an external website in order to connect. This created limitations when it came to deploying the artefact on the university campus, since the network connection is one of these types.

Evaluation

Methods

My method of deployment involved setting up a physical build and placing it in a designated location. I chose to deploy the artefact in the front living room of my house, pointing out the window into the street outside. initially I had planned to to set the device up in a secure office, pointing towards a hallway. I was unable to do this as a secure room was not easily available, posing a problem due to time constraints. Using the street was a good compromise for deployment since no additional risks were raised, this method essentially achieved the same result as being set up in the office.

Deployment and Recruitment

To deploy my application I used the simple method of setting it up to point into the street. I made use of a multi-condition study, between how the face detection picked up users, and how the motion detection picked up users. I observed the how the device was working, as well as, and more importantly how the users were responding to the device. I observed the device by staying in the room myself, and watching as people walked by, focusing on their response, and the device's response to them.i categorised my data by: person; what the device did; and what the user did. Initially I deployed the artefact for 30 minutes for each function(motion detection and face detection). Another thing I made use of was running a few tests myself, in order to tell if it was set up as I intended. Setting the device to point into the street created some problems, users are not likely to look at the device, since they have things to do. There can be many more distractions outside, as opposed to a plain hallway, this is something that I did not realise was a problem until after deployment

results

Face Detection

Test1
Interaction: Tester stands in front of camera
Device Response: Face detected, Wakes up, continue with other animations
User response: N/A

TEST 2
Interaction: Tester stands in front of camera
Device Response: Face detected, Wakes up, continue with other animations
User response: N/A

USER 1
Interaction: Walks past device, does not look
Device Response: Device stays asleep
User response: User did not see anything from the device

USER 2
Interaction: Walks past device, Looks up at device
Device Response: Device does not recognise face, does nothing
User response: Proceeds to carry on walking

USER 3
Interaction: Walks past device, Does Not Look at device
Device Response: Does nothing
User response: Did not see the device, so they kept walking

Motion Detection

TEST 1
Interaction: Movement in front of camera
Device Response: Motion detected, Wake up and wave, continue with other animations.
User Response: N/A

TEST 2
Interaction: Movement in front of camera
Device Response: Motion detected, Wake up and wave, continue with other animations.
User Response: N/A

VEHICLE 1
Interaction: Van parks in front of camera
Device Response: Motion detected, Wake up and wave, continue with other animations.
User Response: N/A

VEHICLE 2
Interaction: Car drives past
Device Response: Motion detected, Wake up and wave, continue with other animations.
User Response: N/A

USER 2
Interaction: User walks past device
Device Response: Motion detected, Wake up and wave continue with other animations.
User Response: Carried on walking past the camera

VEHICLE 3
Interaction: Car drives past camera
Device Response: Motion detected, Wake up and wave continue with other animations.
User Response: N/A

USER 3
Interaction: User walks in front of device, looks, carries on walking
Device Response: Motion detected, Wake up and wave continue with other animations.
User Response: User no longer in front of device

Analysis

From the results gathered, it is clear to see that motion detection was the better implementation method for waking up the device and reactions. Face detection would not activate on the device, in this scenario, it depends upon the user stopping and looking at the screen something that users tend not to do, if they have somewhere to be. This method of implementation would better suit an art gallery of sorts, something where users will be stopping to look, and be close enough for the device to recognise them. Whereas face detection did not work as well as I wanted, motion detection worked almost too well. The motion detection would sometimes get triggered when a vehicle drove past the camera. This is not too much of a problem, since in other scenarios vehicles most likely would not be passing by. The feature triggers when a user walks in front of the device, but with this method the user may not even look, causing the device to play the animations and no-one will see them. When I developed this method I took this into consideration, the artefact can restart when motion is not detected for a while. Something that I gathered from this method of implementation is that it is hard to get users attention when they are doing something else. The artefact did not really alert users to the device being there, causing it to go by unnoticed. In the future, I would aim to implement sound so that users will look for where it is coming from.

Another method of evaluation I could do next would be to set up in an art gallery. This ensures that the device will get usage, as users will be stopping to look and observe. This would be a better case scenario for both face detection and motion detection. This is because with face detection, users will most definitely be close enough and easily detectable in a well lit gallery. Motion detection will also work better since things such as cars would not be a problem.

Ethical Procedures

As for the ethical procedures of my artefact, they were followed accordingly. The camera on the artefact did NOT record footage in any way. The feedback that I gathered was done by me observing what was happening as users passed by the device. The application also does not store footage internally.

Conclusion

To conclude, I achieved in creating my artefact differed to my aims and objectives, but accomplished what I defined using different methods. A few things changed in my development, the use of the Raspberry Pi and the usage of gesture detection. The reasons for these objectives changing was because they were slightly too ambitious, since the resources were not as developed as I would like. The current state of Android running on Raspberry Pi is in early stages of development, with no current official support from google, an adjustment had to be made. Since I already had been testing on an Android phone, I decided to use this as an alternative to the Raspberry Pi. Gesture detection having its limitations was a setback, the compromise of using face detection was made. I made use of face detection instead of gesture detection as it achieved the same goal but via different methods.

When recording my actor, I made a plan for all the animations that I needed. Also I had a backup actor in case my artefact went in a direction where actors could be swapped. I found it easy to edit and splice the footage i gathered. There was not much else to editing the video other than this, VideoPad had all of the features that i needed. I had no issues with recording the footage, the actor was very professional and did as I instructed.

Learning OpenCV, I had some basic knowledge of the library, since I had used it in a previous module. Once I began to read into it and play around with it, I realised how much more complex it was. The final Goal of OpenCV changed from what I wanted from it initially, Gesture detection. This was not really an issue as I made an adjustment and achieved the same thing, but with different methods (face detection).

Creating a physical deployment of the artefact had some problems when in development. I went through various iterations of how I was going to set up the device in an effective way. To begin with, I started with running Android on a Raspberry Pi. I ran into a few problems with this, the device could not recognise the USB camera this way, also there were various performance issues and pixels would tend to get displayed wrong. After this I attempted to make use of an Android phone to create the implementation instead. This went well but I had problems with displaying the artefact on a larger screen. I tried to use two different methods and both ran into different problems. The first was using the Google Chromecast (Google 2017), this worked fine at home but the issue raised was that the university internet required a security access type that the Chromecast did not support. The other method was to use a Slimport (Slimport, 2017) HDMI adapter. The problem that arose with this approach was that the phones that I tested with did not support this external display type.

Testing the implementation wasn't very difficult to do as debugging to Android is fairly intuitive to do. I made use of testing throughout development of the artefact, this ensured that any bugs that presented themselves were fixed before they became a problem. This also helped me to test different methods quickly and easily.

I had a few problems when deploying the artefact. Aforementioned, displaying on a bigger screen was an issue. The deployment scenario was close to what I initially wanted (set up in an office) but some issues were raised when it came to deploying somewhere else (at home, directed into the street). The biggest problem was when motion detection was used, anything that passed by would trigger the condition, as opposed to just a user activating it. Thankfully this would not be a problem in the office scenario, but the compromise had to be made. This caused some errors in my results, such as vehicles being counted as triggering the device. My method of evaluating my artefact made it easy to separate the data, as I observed as it ran.

Reflective Analysis

A problem that presented itself to me during the development process was that I had no previous knowledge of OpenCV, this hindered me partially when it came to developing the gesture detection for my artefact. In the end I decided to substitute it for face detection, since it could achieve the same goals. The problem being, Opencv does not work natively with Java/Android. A lot of the development process was figuring out how to get it set up and working with Java, and native C++ which then makes use of the OpenCV Java library.

Another problem that I ran into was using Android on the Raspberry Pi device. The main problem that I ran into was getting Android onto the Raspberry Pi in a way that worked well enough for my artefact to run smoothly. Getting this to work was frustrating as it required finding an Android image that worked as well as figuring out how to get it onto the Raspberry Pi. A problem that presented after this was the mini LCD screen was not displaying correctly. this was not much of a problem since I was making use of a monitor in the final implementation.

When I was formulating my ideas, I thought of a few different ways that I could have implemented the artefact. The first way that I could have done it was make a Jump scare type application, rather than have the actor wave to the user, it could jump at the screen and play a sound in order to make them jump. This could have been used in a haunted house type fairground ride, since it would only jump if a user walks past and looks at it. Another application, which if I was to continue development, that I could have implemented was the ability to change the moving portrait video clips. My reason for this was that if I deployed the artefact online, then users could change the actor to themselves or a friend. This was because users would not really make use of someone they do not know as a portrait. Another direction that I thought about was the use of a multi-screen setup. Rather than the actor stay on a single screen, when a user walks past, the actor would then instead follow them to another screen. Although this is a good idea, I feel that cost could have been an issue, since multiple devices and screens would have to be set up. Also they would need to be synchronised to each other to ensure that they all react at the right time. Another direction that I thought about was making the moving portrait into a virtual pet, taking inspiration from previous works(Sniff, Good Boy Sammy). What I thought would have been interesting was a pet that grows and evolves as you interact and play with it, sort of like a tamagotchi. Although this was an interesting path to take I was not too sure on how the user could interact, if I had a longer time frame, I could have figured out different interaction methods such as voice. Something else that I thought was interesting would be to give it more of a practical use. The device could be set up in a building such as a university or school. The user could then type in a lecturer/teacher name and their portrait will appear, with contact information, and what room they may be in next. This would have taken a lot more time and resources to do, since I would have to have gotten footage of each lecturer, as well as their timetables and contact information. Another problem with this would be how updates would be handled, I thought about having the lecturer update it personally, by sending an email to the device saying what room they are in. A problem with this is that lecturers can be busy so they might not have the time to update it, also people can be forgetful. This could then cause a student to go looking for someone in a location that they have not been to in a while. Also a problem with it is, what if the lecturer updates the portrait, just after someone has used it, the user would then be looking in the wrong place from the get go.

Another feature that I wanted to implement was gesture detection based on finger count. The reason for not implementing this was that the feature only really worked when the hand was up close. This was something that I had to reconsider, since the camera was to be set up a good distance away from the user. I considered doing body gesture detection, but this would have limited what I could use the user's body for, since I based my ideas around a single hand, and body gestures are not as obvious. I thought that body gestures were very limiting for what I intended to achieve, so I decided against it. I I opted for face detection because, although difficult to do at first, it was a very

good way of implementing the feature. this is because because the scenario that I pictured, which was set up on the side of a hallway, enabled for the feature to be used effectively. For example the artefact wouldnt always see a full face for a duration of time, since the user would be passing by. The application made use of this by counting for a few seconds, if there is a face, then responding if they are still there. Although this works how I wanted it to, I feel as though I could have done it better if I had not used Android so that I then could make use of the Xbox Kinect. The Kinect has much better ways of detecting gestures than a standard camera. The reason for picking Android at first was because I had become quite familiar with it, however if I was to do the moving portrait idea again, I would either code with Java and C++ or even just make the entire thing in a windows form application, using just C++ on its own. This is because I had used C++ in the past, but the knowledge was not as fresh in my mind, so I was worried about it taking some time to get back into understanding the language again.

Video editing was the hardest part of creating the artefact, I have had barely any experience with editing in the past. I found it hard to find a free editor that worked for me, therefore I started using VideoPad. This software was fairly easy to use and I got what I wanted out of it. A problem came when the free trial expired and I did not really want to pay just to use it once or twice. Instead of paying I made use of a free trial of Adobe Premier. Adobe Premier was a lot harder for me to use compared to VideoPad but it achieved the same result as with videopad. furthermore another problem that I had with Adobe Premier is that it would constantly crash, within the first 5 minutes of using it, it crashed twice for 2 different reasons.

Another set of problems was deploying the artefact in a real environment, initially I was to set the device up in an office pointing into a hallway, I had many problems setting up this way. To begin with I wanted to run Android on a Raspberry Pi and make use of a usb camera for the face/motion detection, in the end this did not work. The main reasons being that the Pi did not recognise the usb camera, the other problem was that the Pi did not run the Android OS very well. After this I opted for running the application on an Android phone, since this is what worked already, and then connecting it to a Chromecast wirelessly. This worked at home, but Chromecast requires an internet connection to work. This was something that was hard to set up on the university campus since it requires security methods that the Chromecast does not support. After this did not work I tried to use a SlimPort adapter. This would work the same way as the Chromecast, only directly connected to the device itself, with no internet. I later found out that the phone that I planned to use, did not support this kind of display output. This entire process was frustrating as it took multiple attempts to try and get the physical build to work. This also restricted me when it came to evaluating what I produced since I had to have perfect conditions for the set up. I could have created a hotspot network that allowed just connection to the Chromecast and Android phone. This would mean learning how to set up a private network, then connecting to it. Due to time constraints, did not have time to do this. This is one of the main reasons why I had to change my method of deployment.

In the future, if I was to create the moving portrait again, rather than going with what I am familiar with, I might prefer going with what would work best for development. If I had taken some time to learn how Java works on its own, I could have done without using Android studio, this would then mean that my physical build would still be possible with the Raspberry Pi, resulting in a much cleaner setup. Something that I would like to do differently in the future would be that I would change the idea to a more practical usage, such as finding lecturer offices, and contact information. This is something that I may develop in the future and make use of it. This is because, although I did not realise at the time, first year students can struggle to find lecturer offices, since a lot of them are in hard to find areas. Also it can help the students put a name to a face, since Students would have mostly communicated with lecturers by email, they could search the name in the artefact and retrieve their moving portrait. The gesture detection was very ambitious for just starting out using OpenCV, a lot of the materials was hard to get running and difficult to figure out how it works. This is something I would consider changing, removing OpenCV altogether and just focusing on the idea of a

moving portrait. I feel as though the artefact isn't as good as it could be, if I had more knowledge in the subject area and the materials that I used then maybe it could have ended up how I wanted it.

My next goal with this idea would be to implement some kind of attention grabber. One of the main problems presented with my artefact was that it was hard to get someone's attention, sound was something that I could implement. This way the user would be notified of the artefact being there, as they would look to see where the sound came from. Something else that I would like to change is the orientation of the video, so that it is presented in portrait rather than landscape. This is because with a moving portrait, you would expect it to be portrait. Something that I have thought about improving is the physical implementation of the device, finding a period looking picture frame to house the artefact in could be something I pursue. Also I may put a oil painting filter on the new footage that I record, to help give it that feel of being a painting, since my inspiration was of the moving paintings from Harry Potter.

If I were to revisit the idea, with the knowledge that I have gained I would try an go in another direction with it. I would employ a more practical use for it, such as finding lecturers, their rooms and contact information. I plan to keep Android as it would be very useful for this since it would be easy to deploy, maybe make use of skills that I have picked up in cross platform and develop for ios also. Developing this idea makes it more widely available to users, since it would be easier to get running, with no additional software such as OpenCV.

Overall, I enjoyed working on this artefact. I hit a few snags when developing around certain aspects and this most definitely a useful learning experience. The skills that I picked up working with OpenCV along with the processes involved with developing an independent work will carry on over to other things in the future.

References

Azipurua, Héctor. (2017). h3ct0r/hand_finger_recognition_android. [online] GitHub. Available at: https://github.com/h3ct0r/hand_finger_recognition_android [Accessed 13 Feb. 2017].

Allaboutagile.com. (2017). 10 Good Reasons To Do Agile Development | All About Agile. [online] Available at: http://www.allaboutagile.com/10-good-reasons-to-do-agile-development/ [Accessed 16 Apr. 2017].

Android studio. (2017). *Android Studio*. [online] Available at: https://developer.android.com/studio/index.html [Accessed 16 Oct. 2016].

Arvind Rongala, I. (2017). What is White Box Software Testing: Advantages and Disadvantages. [online] Invensis Blog. Available at: https://www.invensis.net/blog/it/white-box-software-testing-advantages-disadvantages/ [Accessed 26 Apr. 2017].

Arvind Rongala, I. (2017). *What is Black Box Testing: Advantages and Disadvantages*. [online] Invensis Blog. Available at: https://www.invensis.net/blog/it/black-box-testing-advantages-disadvantages/ [Accessed 26 Apr. 2017].

Bmva.org. (2017). *What is computer vision?*. [online] Available at: http://www.bmva.org/visionoverview [Accessed 26 Apr. 2017].

Bradski, G. and Kaehler, A. (2011). *Learning OpenCV*. Beijing [u.a.]: O'Reilly.

Bruce, J. (2017). *Flow Diagram*. [image]<flowdiagram.jpg> [Accessed 26 Apr. 2017].

Bruce, J. (2017). *Moving portrait button*. [image]<movingPortButton.jpg> [Accessed 26 Apr. 2017].

Bruce, J. (2017). *Motion detection*. [image]<motionDetection.jpg> [Accessed 26 Apr. 2017].

Bruce, J. (2017). *Face detection*. [image]<faceDetection.jpg> [Accessed 26 Apr. 2017].

Bruce, J. (2017). *Deployment*. [image]<deployment.jpg> [Accessed 26 Apr. 2017].

Costello, B., Muller, L., Amitani, S. and Edmonds, E. (2005). Understanding the Experience of Interactive Art: Iamascope in Beta_space. [online] Available at: https://opus.lib.uts.edu.au/bitstream/10453/11518/1/2006014049.pdf [Accessed 5 Dec. 2016].

"Dtdzung". YouTube. N.p., 2017. https://www.youtube.com/channel/UCvRTnofGf_Svg2Q-94KUm8g. 1 mar.. 2017.

Disk Imager, (2016). Win32 Disk Imager. [online] SourceForge. Available at: https://sourceforge.net/projects/win32diskimager/ [Accessed 1 Dec. 2016].

Gehring, J. (2017). appsthatmatter/android-motion-detection. [online] GitHub. Available at: https://github.com/appsthatmatter/android-motion-detection [Accessed 8 Feb. 2017].

George, J. (2012) Sniff - obviousjim. http://jamesgeorge.org/Sniff (Accessed: 21 October 2016).

GlobalTeckz. (2017). Characteristics of Agile Methodology in Software Development - GlobalTeckz. [online] http://blogs.globalteckz.com/characteristics-of-agile-methodology-in-software-development/ [Accessed 1 Apr. 2017].

Google. (2017). For TV - Chromecast - Google. [online] Available at: https://www.google.com/intl/en_uk/chromecast/tv/explore/?gclid=CjwKEAjw5_vHBRCBtt2NqqCDjiESJABD5rCJqT4z_wKG2FxXTrJR6Cb7hKfj04BVSLKxjnB7jK58YhoC7jbw_wcB&gclsrc=aw.ds&dclid=CJ7

isL-fwNMCFZIx0wodO3YBIA [Accessed 25 Apr. 2017].

Guru99.com. (2017). [online] Available at: http://www.guru99.com/black-box-testing.html [Accessed 26 Apr. 2017].

Harry Potter: Prisoner of Azkaban.. (2004). [film] Directed by A. Cuarón. United Kingdom: Warner Bros.

ISTQB(2017). What is Agile model – advantages, disadvantages and when to use it?. [online] Istqbexamcertification.com. Available at: http://istqbexamcertification.com/what-is-agile-model-advantages-disadvantages-and-when-to-use-it/ [Accessed 16 Apr. 2017].

Janarthanan .S. (2017). Beta_space. [online] Available at: http://research.it.uts.edu.au/creative/betaspace/moreInfo.html [Accessed 17 Mar. 2017].

Perrone, A. (2016) Good boy, Sammy - angela @ ITP. http://www.angelaitp.com/goodboysammy/ (Accessed: 21 October 2016).

Qin, Y. (2017). eaglesky/HandGestureApp. [online] GitHub. Available at: https://github.com/eaglesky/HandGestureApp [Accessed 13 Feb. 2017].

Raspberry Pi Foundation. (2017). Raspberry Pi - Teach, Learn, and Make with Raspberry Pi. [online] Available at: https://www.raspberrypi.org/ [Accessed 1 Oct. 2016].

Research.it.uts.edu.au. (2017). Beta_space. [online] Available at: http://research.it.uts.edu.au/creative/betaspace/moreInfo.html [Accessed 17 Mar. 2017].

Samsung uk. (2017). Samsung Galaxy S7 and S7 edge - The Official Samsung Galaxy Site. [online] http://www.samsung.com/uk/smartphones/galaxy-s7/overview [Accessed 18 Feb. 2017].

Sdcard.org. (2016). SD Card Formatter - SD Association. [online] Available at: https://www.sdcard.org/downloads/formatter_4/ [Accessed 1 Dec. 2016].

Slimport. (2017). *SlimPort - Connect mobile devices to displays*. [online] Available at: http://gb.slimportconnect.com/ [Accessed 26 Apr. 2017].

Wetherell, J. (2017). phishman3579/android-motion-detection. [online] GitHub. Available at: https://github.com/phishman3579/android-motion-detection [Accessed 8 Feb. 2017].

.