TABLET PC BASED WEB APPLICATIONS IN THE REHABILITATION OF STROKE PATIENTS

MOHAMMAD USMAN ARSHAD

MSc Advanced Computer Science, the University of Strathclyde

Supervisor: Dr John Levine, Senior Lecturer

Word Count: 26105

DECLARATION

This dissertation is submitted in part fulfilment of the requirements for the degree of MSc of the University of Strathclyde.

I declare that this dissertation embodies the results of my own work and that it has been composed by myself. Following normal academic conventions, I have made due acknowledgement to the work of others.

I declare that I have sought, and received, ethics approval via the Departmental Ethics Committee as appropriate to my research.

I give permission to the University of Strathclyde, Department of Computer and Information Sciences, to provide copies of the dissertation, at cost, to those who may in the future request a copy of the dissertation for private study or research.

I give permission to the University of Strathclyde, Department of Computer and Information Sciences, to place a copy of the dissertation in a publicly available archive.

Yes [✓] No [ ]

I declare that the word count for this dissertation (excluding title page, declaration, abstract, acknowledgements, table of contents, list of illustrations, references and appendices is 26,106 words.

I confirm that I wish this to be assessed as a Type 3 Dissertation.

Signature:

Date:

ABSTRACT

This study investigated to what extent HTML5 and JavaScript could be used in the implementation of tablet/mobile based games which are platform independent, browser independent, and are engineered towards the rehabilitation of stroke victims. Firstly, project aims and minimum requirements were set. The methodological structure of this project, which was required to satisfy these aims and requirements was then outlined. Followed by this, a background study into video games and tablet PCs in the context of healthcare was carried out. From this study it was realised that the potential of tablet PCs and bespoke video game development within the healthcare system has yet to be fully realised.

Current methods of using video games within the healthcare system have been achieved through off-the-peg methods which may aid in rehabilitation, but only in a general manner. It was realised that there was a gap in research in regards to bespoke video games developed specifically for the rehabilitation of patients. Furthermore, it was discovered that the current usage of tablet PCs within the healthcare system was stagnant and the benefits of tablet PC-healthcare integration could yet be reaped. Through the aims and objectives set, this research worked towards bridging this gap between the usages of tablet PCs and bespoke video games in the healthcare system. Next, the capabilities and limitations of HTML5, JavaScript, and the Phaser game framework in regards to this project were highlighted. From this, it was concluded that theoretically the development languages and tools were adequate enough to achieve the outlined aims and objectives.

The next stage of this project was the design of the prototypes to be created. These designs were justified by the research carried out in the previous stage. This investigation was satisfied through the creation of four prototypes which acted as empirical evidence of how the aforementioned topics of interest were achieved. Two of the four prototypes were aimed at the stimulation of damaged cognitive abilities, more specifically visual and verbal memory. The other two prototypes were aimed at the stimulation of damaged physical abilities, more specifically, fine motor skills. The implementations were carried out successfully and could then be assessed in the evaluation stage.

The evaluation stage consisted of user trials, the researcher's evaluation, and professional feedback. From the user trials it was determined that platform independence, and browser independence, had been achieved. Furthermore, it was also concluded that the prototypes created were playable, reasonably robust, and seem to those who are not experts in the field or stroke patients that it is plausible that they can be used in the rehabilitation of stroke patients. This however can only be verified through extensive user trials and expert feedback. The researcher's evaluation was used to gauge whether or not from the researcher's perspective, the prototypes created met the minimum requirements and adhered to the designs outlined in the design stage. It was realised that prototypes did fully adhere to the designs set out for them and the project had met its minimum requirements. From the professional feedback, it was realised that although the prototypes created could potentially improve cognitive and physical abilities through stimulation, these improvements may not necessarily translate directly to improvements in a stroke patient's everyday activities. The most important significant knowledge gained from the profession evaluation was that, although tablet PC based video games from the professional's perspective, could potentially be used in the rehabilitation of stroke victims, their design is critical in doing so.

The prototypes created have given the researcher a method to gauge the reaction of professionals in the field and stroke patients. From this, requirements elicitation can now be sought as the researcher now has working prototypes to gauge reactions.

Acknowledgements

Without the support, patience, and assistance of the following people, this dissertation would not have been completed. I am eternally grateful.

Firstly, I would like to explicitly express my deepest appreciation to my supervisor, Dr John Levine, who has provided me with support and guidance during my project. Thanks to his weekly meetings and prompt responses to my e-mails, I have been clear of what I must do to be successful in this project throughout its undertaking. His patience and wisdom was instrumental in my undertaking of this project.

Secondly, I would like to thank Dr Mark Dunlop for his guidance offered throughout the undertaking of this project. I would like to thank him for the meetings that we had, as with every meeting I was more aware of what was required to be successful in this project.

In short, the wisdom and guidance offered by both Dr John Levine and Dr Mark Dunlop have been instrumental in the undertaking of this project.

Thirdly, I would like to thank the 5 participants and the professional who took time out to test the prototypes thereby assisting the completion of the dissertation.

Last but most certainly not least, I would like to thank my parents and family for their untiring support throughout my entire life.

Table of Contents

TABLE OF FIGURES

1 Chapter 1: Introduction

This chapter aims to introduce three components essential in the undertaking of this project. Firstly, the projects overall aims and objectives will be highlighted followed by the specification of the projects minimum requirements. How these minimum requirements could potentially be achieved will then be discussed. To end this chapter the project methodology will be defined, which acts as the projects structure.

1.1 Project Aims

The overall objective of this project is the creation of a tablet PC based software solution which will aid in the recovery and rehabilitation of stroke victims. This should be achieved through the medium that is video games and in a manner that is platform and browser independent. One of the main areas of interest to be investigated in this research is to what extent HTML5 and JavaScript can be used in the implementation of this software.

In order to determine whether or not these web languages are suitable for the implementation of this system the actual requirements of the system must be outlined.

1.2 Minimum Requirements

The minimum requirements for the software solution to be deemed successful are as follows:

1. Must be platform independent (Supporting at least both Android 4.1+ and iOS platforms).
2. Must be browser independent (Supporting at least Chrome and Mobile Safari).
3. Targeted at the rehabilitation of stroke victims.
    a. Stimulate damaged physical abilities
    b. Stimulate damaged cognitive abilities

1.2.1 Platform and Browser independence

The term 'platform independence' in the context of this project means an application should be developed which is supported by both Google's Android and Apple's iOS operating systems without bias. This meaning no special development specific to a certain operating system should occur. A software solution which can run on both Apple iOS and Android based devices without coding specifically for a certain platform is to be created.

A problem currently faced when developing applications for tablet and mobile devices is the fragmentation caused by different hardware manufacturers, operating systems, and browsers. According to Gartner (2014)[1] there are currently two primary shareholders in the Mobile/Tablet Operating System market; Google and Apple Inc. As of February 2014 Google's Android operating system has held approximately 78 percent of the overall Mobile/Tablet operating system market share. Apple's iOS operating system has held approximately 15 percent. There are other competitors in the market however, their market shares are so small that they are negligible. One market share holder that should be mentioned is Microsoft which holds a market share of approximately 3 percent. Microsoft operating system based devices will be discussed in later sections of this document. In regards to this project only support for Google's Android and Apples iOS operating systems will be taken into consideration.

Google's Android operating system has managed to gain significant traction within the market as a result of being open source. This allows different manufacturers to create their own hardware devices and then ship with the Android operating system. As the Android operating system is open source, tablet/mobile manufacturers can create their own versions of the Android operating system and deploy devices with it preinstalled. The popularity of the Android operating system can be attributed to the fact that it is open source.

Conversely, Apple's iOS is under a proprietary EULA and is exclusively deployed on Apple devices (Apple, 2014)[2]. This meaning, only Apple devices make use the Apple iOS software. A major difference between both operating systems is the development process when creating applications. Each operating system has its own storefront from which applications can be uploaded and downloaded. These applications are created in different manners and this is what limits current applications from being platform independent. An application created for an Android device cannot be deployed on an Apple device without completely revamping the applications code. This is because the programming languages and development tools used differ.

As stated by Android (2014)[3], Android based applications are primarily written in Java and the Android Software Development Kit (SDK) is used to compile the code into an installable application. Apple iOS based applications however are written in Objective-C. Apple iOS application development requires the iOS SDK, xCode (Apple's Integrated Development Environment) and a computer which runs OS X 10.8 or newer (Apple Inc, 2014)[4]. Although the general principles for Android and iOS application development remains the same (use of a platform specific language and a bespoke SDK), the detailed differences between the two set them apart significantly. In order to develop for each platform two separate pieces of code must be written, developed, and deployed using each platforms different programming languages and bespoke tools. Thus platform independence cannot be achieved through traditional native application development methods.

In order to create applications which are platform independent, web application can be created instead of native applications. These web applications would be developed in HTML5 and JavaScript, and would require the Tablet/Mobile browser to support these languages. They are inherently platform-agnostic on the basis that that the browsers on different platforms equally support all the required features. At this point the issue is no longer platform independence, but browser independence. Through creating a web application the concern is no longer whether or not the platform can support the application instead the problem is now creating an application which can run equally on the different platforms browsers.

Tablet and Mobile devices have the functionality to run web-browsers in the form of native applications. Both iOS and Android based devices come preinstalled with an operating system specific browser. A mobile version of the Safari browser was developed and deployed with the first generation iPhone in 2007 (Apple, 2007)[5]. This mobile version of Safari has been deployed on and is supported by all Tablet/Mobile based Apple devices. In order to develop web applications for the latest Tablet/Mobile Apple devices you only need to develop for the latest Safari mobile browser. Devices running older version of Android have the generic Android stock browser preinstalled. After the release of Android version 4 (ICS) the stock browser was replaced with Google Chrome (Smith, 2012)[6]. This meaning, the Google Chrome browser is preinstalled in all the latest and current Android devices.

The application storefronts on each platform can be used to download various other web-browsers i.e. Mozilla Firefox or Opera. This can be problematic when trying to provide support for as many browsers as possible as not all browsers support the same functionality. In order to overcome this certain restrictions on the web applications being developed for this project must be introduced. When creating this web application only compatibility for the default web-browsers of each platform will be supported. In this case this means only support for the latest versions of Apple Safari and

Google Chrome web-browsers will be targeted. This has to be done as developing to support all web-browsers on the market and older versions of these browsers will result in fragmentation. If features of the web application are not supported by a certain browser or browser version these features would have to be removed completely or solely from the incompatible browser. Removing these features completely limits the functionality of the final product. Removing the specific features from certain browsers or browser version creates an unequal experience across all the browsers.

For the aforementioned reasons only support for Safari and Chrome will be taken into account. In order to achieve platform independence a web application will be created with the HTML5 and JavaScript programming languages, instead of a native application. Furthermore, browser independence in the context of this project will mean the default Android and iOS browsers will be supported equally.

1.2.2 Targeted at the Rehabilitation of Stroke Patients

The web application to be developed in question is to focus on the rehabilitation of stroke patients. Strokes can potentially damage the body's physical, cognitive, and communication capabilities. Physical impairments include, damaged fine motor skills and visual perception. Cognitive impairments include, damaged cognitive thinking abilities, memory impairment, and Aphasia. Although there is no complete cure for the damage caused by strokes, the damage can be reduced through stimulation which improves upon the damaged area of effect (Stroke.org, 2014)[7].

As stroke patients can be affected by these impairments it is imperative that the software is designed in such a manner that is stroke patient friendly. The software's user interface and usability will have to be designed specifically for the use of stroke victims.

1.2.2.1 Stimulating Damaged Physical Abilities

The software to be developed is required to stimulate at least one damaged physical ability. It has been planned that this be a stroke patients fine motor skills. A person's fine motor skills allow them to make calculated movements with the use of their hands and fingers, this require precision both physically and mentally. Examples of such movements include handwriting and picking up a small object. According to StrokeAssociation (2014)[8], fine motor skills are made up of three components,

skeletal, neurological, and muscular functions. The functionality of one or more of these components can be disrupted after a stroke. In order to cope with these disruptions the human brain remodels itself to create more optimal neural connections. Doing so allows the body to adjust dynamically to disruptions. This happens throughout a human's life in accordance to its setting and to compensate for injuries (Hammond, 2002)[9]. This process is called Neuroplasticity and in the context of strokes it helps a stroke patient's brain to shape new neuronal pathways in place of those damaged by a stroke. As mentioned by Hammond (2002), physical stimulation stimulates the synapses of the nerve cells and thus enables them to create new neuronal connections.

From this, it is clear that damaged neurological functions can be improved upon with the use of physical stimulation. A working example of this is as follows, if a person tends to make more use of their muscular functions when picking up a small object and these functions are then damaged by a stroke. The brain can cope with this damage by employing more use of stronger functions over the damaged function. For example, making more use of skeletal features over muscular and vice versa. Neuroplasticity allows for this adaptability.

1.2.2.2 Stimulating Damaged Cognitive Abilities

The software to be developed is required to stimulate at least one damaged cognitive ability. It is intended that multiple cognitive abilities will be stimulated in order to improve upon them. The software should aim to help in the recovery of impaired memory abilities, object correlation abilities, and damage caused by Aphasia. This could potentially be achieved through exercising both visual and verbal memory. Furthermore, as with damaged physical abilities, neuroplasticity also plays a major role in the recovery of damaged cognitive abilities. Instead of macro changes taking place which are essentially changes of behaviour, micro changes within the brain occur. Micro changes such as the development and change of neuronal pathways in the brain. Neuronal pathways required for certain cognitive abilities which were damaged due to a stroke can be gradually remodelled through neuroplasticity.

1.3 Project Methodology

In order to achieve what is required of this project the method for its success must be outlined and followed. This will give structure and organisation to the undertaking of this project. The protocol for achieving success by the end of this project can be broken down into four phases which should be followed sequentially, Research and Preparation, Design, Implementation, and Evaluation. This methodology will loosely follow the waterfall development method. The same sequential approach to the different stages shall be taken and similar core activities shall be carried out.

1.3.1 Research and Preparation

The first phase of this process is Research and Preparation. This phase comprises of a thorough study into the background and current state of the topic. This is carried out to gain a relevant and comprehensive understanding of the subject. Furthermore, this phase acts as the precursor to the writing of the literature review and any other writing regarding the topic. It is nigh impossible to find specific information regarding the exact topic, instead, the components which make up the topic should be investigated. In order to gather enough information to move into the writing of the literature review and onwards, the components relevant to this topic are to be investigated. The investigation of these topics is structured in such a manner that the transition between moving from one topic to the other should feel natural.

This project aims to assess to what extent HTML5 and JavaScript tablet based video games can be used to aid in the rehabilitation of stroke victims. The most fundamental component of the topic is strokes and thus the first topic to be investigated is strokes. The effects of strokes are to be understood followed by an investigation of current rehabilitation methods. From this point the study should be steered towards experimental rehabilitation methods.

The next topic to be researched is video games. During this stage an understanding of video games and their current role within the healthcare system is to be gained. The potentially positive effects of video games in regards to health and how video games can be engineered to achieve these positive effects will also be investigated. It is planned that once an understanding of both strokes and video games has been achieved both of these topics can be compared in order to find overlaps between the two. Having this understanding of both video games and strokes will be instrumental in the design of the video games to be created as part of this project. Through researching the current usage of video

games within the healthcare system, current video game design can be observed. After determining which parts of the different video game designs fit the scope of this project and how well they complement this project, it will be possible to make educated decisions regarding the design of the games to be created. This part of the investigation will be most helpful during the design phase.

The next step in this process is the investigation of mobile devices (mobiles and tablets), their capabilities, and current attempts at using them within the healthcare system. This stage heavily ties into the two previous stages. Current methods of developing video games for these devices should also be looked at. In doing so, current attempts at creating video games with web based technologies will be highlighted. Next, the pros and cons of creating web based applications in place of native applications shall be investigated. This will naturally move the research towards the web based technologies used in the creation of web based video games. These technologies being, HTML5 and JavaScript. The capabilities and limitations of both of these web languages is to be investigated. To finalise on the research and preparation stage a deep yet broad understanding of HTML5 and JavaScript is to be gained.

1.3.2 Design

General design principles for modern video games shall be noted and taken into account for the design of the prototypes. This knowledge alongside the knowledge gained regarding the damage caused by strokes and current stroke rehabilitation methods, will be used to aid in the design of the games to be developed. Further information about the target demographic and the demographic of the victims of strokes should also be taken into account during the design stage. After taking these factors into account it is proposed that 2D renderings of the games to be developed are created. These renderings could then be analysed and scrutinised in order to determine both good and bad design choices. The information gathered during the research stage regarding the technologies being used will also be used to determine whether or not the proposed design shown in the renderings is possible with the tools being used. To summarise, sketches and 2D renderings of the games to be developed will be created, these should then go through an informal evaluation in order to determine flaws or shortcomings in their design.

1.3.3 Implementation

Implementation is the third phase in this process. The implementation will consist of the documentation of the development of prototypes. These prototypes will be based on the designs chosen during the design phase (phase two). The knowledge gained during the research and preparation phase will be tested in this stage. In order to complete the implementation to a good standard both the previous stages must be achieved first. The sequential structure that this process follows ensures that each phase is carried out to a high calibre, this is because each phase is dependent on the previous ones. The final deliverables at the end of this phase will be the prototypes, these prototypes will then be used to move into the next stage, the evaluation phase. It should be noted that it proposed that an incremental development approach be taken in order to complete the implementation stage.

1.3.4 Evaluation

The final stage in this process is the evaluation stage. In order to be successful in the completion of the evaluation stage the prototypes created in the previous stage have to be analysed and assessed in order to determine whether or not the questions raised as part of this project are answered. This evaluation shall consist of three stages, user evaluation, researcher evaluation, and professional evaluation. The user evaluation shall consist of user tests and questionnaires in order to gain actual user feedback regarding their experience with the prototypes. The researcher's evaluation shall be carried out by comparing the prototypes to the minimum requirements which were previously outlined. The professional evaluation shall consist of a user test carried out by a professional in the health sector followed by an interview. At the end of this stage it should be known to what extent JavaScript and HTML5 tablet based games can be used to aid in the rehabilitation of stroke victims. At this point future work and potential improvements to this work will be discussed. Conclusions will be drawn and this project will be finalised.

1.4 Summary

In this chapter, the project aims have been outlined and the minimum requirements which this project must meet to be deemed successful have been set. Furthermore, discussion regarding the nature of web applications and how they can used be used to satisfy the requirement of platform independence has been discussed. Additionally, how browser independence and the rehabilitation of stroke victims could potentially be achieved has also been outlined. Finally, the project methodology was set which is to act as the structure for the rest of this report. This structure is made up of four main components, background study, design, implementation, and evaluation.

2 Chapter 2: Literature Review

This chapter is made up of two main sections. The first section shall discuss video games within the healthcare system and in the context of strokes. Followed by the discussion of tablet PCs and their current usage within the healthcare system. Through highlighting previous work done in this fields, it is anticipated that the researcher is able to show the readers the gap in research which they aim to fill.

Furthermore, the second section shall explore the current capabilities of the HTML5 and JavaScript web programming languages. The features and capabilities of the Phaser framework will also be discussed. This is essential background study as from this, a theoretic estimation of whether or not the web languages are suitable in the creation of an application which is targeted at the rehabilitation of stroke patients, can be made.

2.1 Video Games and Their Relation to Healthcare

The video game industry is one which has seen massive adoption, evolution, and growth within the 21st century, even more so than every other sector in the entertainment industry. Video games have changed the very dynamics of how we interact with technology by acting as a catalyst in the advancement of computer based technologies (ESA, 2013)[10]. Video games and their effects on society have become a sensitive topic within the media and they are constantly being demeaned for their arguable ill-effects. Connolly (2012)[11] has mentioned that much of the early research carried out regarding video games has focused on negative issues, these issues being greatly sensationalised in the media. The impact that gaming has made on society has triggered scholarly interest in the matter and thus in recent years, as expressed by Martin (2012)[12], many studies have been carried out which aim to bring to light the positive effects of video games. Video games are continually being tested in order to provide empirical evidence of their benefits.

Such a benefit, which has been empirically proven by Green and Bavelier (2003)[13], is that video games can enhance a player's visual perception amongst other visual sensory improvements. Video games by nature incorporate the use of multiple senses at once (sight, sound, fine motor skills and cognitive skills) and because of this they can invoke intense stimulation to them. The stimulation of these senses could possibly be engineered to achieve specific benefits such as improved fine motor skills. It is clear

from the vast amount of research carried out in this field that video games do indeed have many positive points and can benefit the players. As stated by Primack and Carroll (2012, p638)[14] there is potential for video games to improve health outcomes, particularly in the areas of psychological and physical therapy. Video games are also easy to adopt and engage, this results in them being able to impact large populations which has previously been a difficult task according Read and Shortell (2011)[15].

The use of video games as an aid in the healthcare sector is a logical relationship that should be built because of how each sector compliments each other. The sensory, motor, and cognitive improvements which video games can potentially offer, synergise with the treatment of patients within the healthcare sector. As articulated by Sawyer (2008)[16], one of the most significant sectors witnessing the impact of games is the healthcare sector.

2.2 Video games and their use in the context of strokes

It is empirically proven that the healthcare sector can indeed reap the benefits of playing video games when treating patients (Green and Bavelier, 2003). These games can then be tailored to gain specific benefits such as improved vision for a patient suffering from poor sight. However, as previously mentioned, video games invoke the use of multiple senses at once and because of this they could potentially be used to aid in the recovery of multiple conditions. Strokes are destructive and their victims almost always suffer from more than one side effect of the stroke. These side effects being, physical, cognitive, and emotional damage (National Stroke Association, 2014)[17]. One of the physical after effects of strokes is impaired visual perception and as previously mentioned there have already been studies carried out which demonstrate that video games can be used in order to improve visual perception.

As discovered by Lee (2013)[18] when experimenting with video games and their ability to improve fine motor skills in stroke victims, there was no difference between the group exposed to traditional methods of rehabilitation and the group exposed to video game based rehabilitation methods. This meaning, video games are more than capable of providing adequate rehabilitation. The recovery of the stroke victims that used video games in their rehabilitation programs was equally as good as the recovery of those exposed to the standard health service rehabilitation programs. The author further notes that during the experiment the subjects which used video games seen improved motor function and better performance in everyday activities. This research further highlights the potential usage of video games within the health sector.

Another issues faced by stroke victims is the relearning of what would seem like trivial functions to the average person, functions such as knowing the relationships of certain items I.e. a cup goes on top of a saucer and not the other way around. Video games could potentially be used to aid in the relearning of these basic functions lost due to strokes. De Wit-Zuurendonk and Oei (2011)[19] have both discussed the advantages of making use of video games in a learning environment, they have done so by looking at how adults and children learn, and based on this information they analysed how games comfortably fit these methods of learning. The authors make the point that ‘adults prefer independent learning’, and later mention that gaming promotes a dynamic method of learning which is done independently. This answers the question of whether or not video games can be used as tool for learning amongst adults. Furthermore, video games are seen as enjoyable and trendy amongst children and young adults and for this reason they are easily adoptable and can possibly aid in learning where traditional methods fail, simply due to the fact that they are appealing to younger generations.

It is clear that the multi-stimulant element of video games makes them a streamline candidate for use in the recovery of stroke victims which are affected by multiple conditions. It is also clear that video games offer the same if not a better recovery than traditional rehabilitation methods.

2.3 Tablet Computers

Advancements in the realm of computer architecture in the past decade have greatly impacted the modern world and the devices we interact with every day. Advancements which have allowed for the production of smaller CPU chips and general computer components have led us to where we are now in the 21st century, in a world where computing technology has become an integral part of everyday life. The development and adoption of mobile computing has allowed for this to happen. The reduction in CPU sizes over the years has led to less power consumption and less heat generation, as a result of this, compact mobile computing became a possibility.

A device which has been at the forefront of the mobile computing sector has been the tablet computer. As described by PCMag (2014) [20] a tablet computer is "a general purpose computer contained in a single panel". Its most common form of input is a touchscreen display and generally tablet computers have various hardware features inbuilt to the device. Features such as, cameras, accelerometers, gyroscopes, microphones, and various other hardware additions. Tablet computers were first conceptualised in 1940 (Hager and Burka, n.d., p2-3)[21] and have developed over many years of iterations into the Tablet computers that we see today. They have seen explosive growth both commercially and technologically in the last decade. This growth has been especially concentrated from 2010 onwards due to the introduction of the IPad. Prior to the IPad there had been multiple attempts at introducing tablets into the commercial world however, these attempts at mainstream popularity were short-lived. Twice before the coming of the IPad, Tablet computers had seen rapid adoption but on both occasions a crash in the Tablet market had set them back again (Walker, 2011)[22]. This unstable market can be attributed to factors such as, lack of suitable hardware, high prices, high power consumption, and various other factors.

The IPad was successful because of multiple reason however as stated by Walker (2011, p15) the success of the IPad can be accredited primarily to the "availability of appropriate hardware" and "the rise of social media". Technological advancements concerning CPU architecture (smaller chips) led to a more viable mobile computer in terms of power consumption and heat generation. Simply put, Tablet computers were not meant to succeed as a consumer product until now, this is because the technology required for a streamline user experience did not exist.

2.4 Tablet Computers in the Healthcare System

As previously mentioned, tablet PC have seen incredible growth in mainstream usage. However, their usage within the healthcare system has been limited and there is still potential uses of the Tablet PC within the healthcare system to be explored. An example of when tablet PCs have been used within the healthcare system is the usage of the HP tablet PC which was designed to better a healthcare professional's everyday activities (HP, 2014)[23]. Activities such as, patient record retrieval, review, and collaboration. Another significant activity improved upon is the collecting and integrating of patient data. As discussed by HP (2014), Tablet PC integration into a healthcare professionals everyday duties results in reduced medical errors, increased efficiency, and shorter patient visits. With HP's attempt at Tablet PC-healthcare integration it is evident that Tablet PCs can be used in the betterment of quality of life for healthcare professionals. This has been discovered by a range organisations who have worked towards Tablet PC-healthcare integration. As discussed in the Motion Computing white paper (2006)[24], tablet PCs benefit clinicians by saving time, ensuring they always have the most up to date information, and reducing errors by removing the human element in the aggregation of patient data. It is evident that tablet PCs can and have been used to the advantage of healthcare professionals.

The next step for tablet PCs within the healthcare system would be the usage of them in the betterment of patient's treatment, instead of solely healthcare professional's everyday activities. As previously discussed, video games have already shown that they could potentially be used as an aid in the recovery of patients. The usage of video games mentioned in section 2.2 were achieved through off-the-peg methods of rehabilitation. This meaning, the video games used to stimulated damaged abilities happened to aid in a patient's recovery by default without any bespoke development. The proposition offered by this project is the development of a bespoke piece of software which is designed specifically for the rehabilitation of stroke victims. This software should be designed in such a manner that it can be deployed on a range of tablet/mobile devices without additional development.

The rationale behind the deployment of such an application on tablet PC/mobile platforms can be attributed to multiple factors. Firstly, tablet PCs and mobile devices are easily accessible, have low manufacturing costs, and are already heavily integrated into society. Secondly, the potential usages of tablet PCs within the healthcare system are yet to be fully explored and for this reason, there is potential growth in this field. In other words, it is an untapped market. Furthermore, the reasons behind the development of a bespoke software solution instead of using a generic solution are also plentiful. The primary reason being, a bespoke software solution allows for the software to be tailored specifically towards certain patients. This is critical in fine tuning success in the recovery of a patient.

For example, a generic software solution may aid in the recovery of damaged physical abilities overall however, a bespoke software solution is capable of being tweaked to address a patients especially weak fine motor skills. This would allow for more precise targeting of certain damaged abilities. Furthermore, bespoke software solutions could be extended to allow for the collection, distribution, and storing of patient information. In summary, a bespoke software solution allows for rehabilitation specific to a patient to be achieved. Furthermore, bespoke software solutions allow for carers to stay involved with this new form of rehabilitation as patient progress and results could potentially be forwarded to carers when collected.

This research aims to bridge the gap between the usage of Tablet PCs and video games within the healthcare system. The benefits of Tablet PCs have already been outlined just as the usage of video games in the rehabilitation of patients has. These two elements could potentially come together through the development of a bespoke video game developed for tablet/mobile platforms, this would bring together the benefits of both elements.

2.5 HTML5 and JavaScript: Capabilities

This project aims to explore to what extent JavaScript and HTML5 can be used in the creation of a platform/ browser independent game which is aimed at the rehabilitation of stroke victims. In order to achieve this the relevant features and concepts of these languages will be singled out and discussed. The importance of each feature in achieving this goal shall be highlighted. Furthermore, how these will be used in order to achieve the end goal shall also be outlined.

2.5.1 HTML5

HTML (Hypertext Mark-up Language) is a mark-up language which is currently the standard in web development. Raggett, Le Hors and Jacobs (1998)[25] have described HTML as a publishing language which is used in the creation of all websites. It is universally understood and is one of the primary components in the World Wide Web as it gives websites the ability to structure and present content. HTML was created by Tim-Berners-Lee during his time working at the European Organization for Nuclear Research (CERN). As the web gained popularity and grew, so too did HTML. HTML has been incrementally improved upon with new releases throughout the years, each improvement adding new features and support. The most current and up to date version of HTML is HTML5.

HTML5's development started in 2006 when the World Wide Web Consortium (W3C) showed interest in joining the development of HTML5. At this point the W3C development team joined the Web Hypertext Application Technology Working Group (WHATWG) development team in the creation of a HTML5 specification. The two teams worked together until 2011 when it was realised that the aspirations that each team had for the future of HTML5 were different. WHATWG planned to continuously maintain the HTML5 specification and continually add new features, creating a living standard. While the WC3 team planned to work towards a final specification instead of a continually evolving specification. This resulted in both teams splitting up and from there on out each team worked towards their own aspirations for HTML5. The WC3 actively work towards the convergence of their HTML5 specification and the WHATWG specification. Naturally as the end goals of the two groups are different, the WC3 only seek convergence within the bounds of their goals (Berjon et al., (2014)[26]; Whatwg.org, (2014)[27]).

There is a wide range of difference between HTML5 and other antiquated versions of HTML however, the features of most relevance to this project are the new application programming interfaces (API). APIs relevant to this project will be analysed to determine how useful they are to the project.

2.5.1.1 Device Motion

Modern mobile devices are deployed with various sensors inbuilt to the device. A sensor commonly found in almost all modern devices is the accelerometer. HTML5 allows developers to access these hardware readings through the use of device motion events. The device motion event returns the readings of the accelerometer hardware at constant intervals. These readings hold information relating to the acceleration of the device on the x, y, and z axes and are measured in meters per second squared (HTML5 Rocks, 2011)[28]. These readings allow developers to determine the orientation of the device. In the context of this project, these readings could potentially be used as a means of stimulating a user's fine motor skills. As the orientation of the device can be determined, this orientation could possibly be used as control input for one of the games being developed. A user could control a player sprite based on the orientation of the device. Moving a player controller sprite through a maze with the use of the device's orientation could potentially aid in the rehabilitation of stroke victims by stimulating their fine motor skills. The x, y, and z axes relative to the positioning of a mobile device is shown in Figure 1.

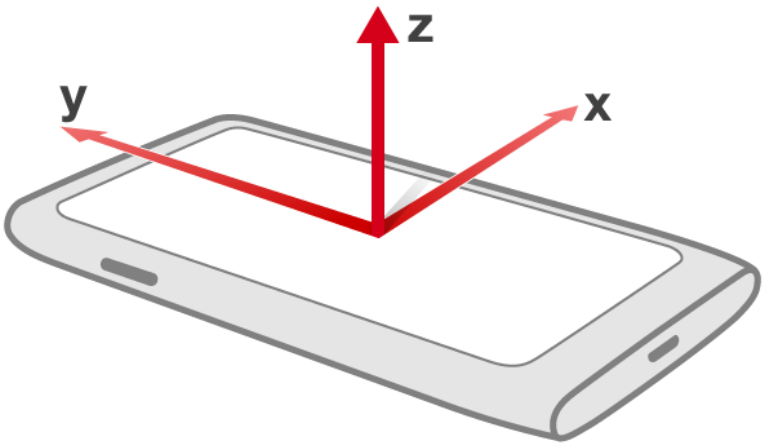

Figure 1: X, Y, and Z axes relative to a mobile device. Source: https://dev.opera.com/articles/w3c-device-orientation-usage/device-axes.png

2.5.1.2 Device Orientation

Another sensor which is deployed with most modern mobile devices is the Gyroscope. Gyroscopes allow for the angle of orientation of these devices to be calculated. These readings are returned through an event in HTML5 called device orientation. The rotation data returned includes the orientation of the device from side to side (gamma, x axis), from front to back (beta, y axis), and with the use of the devices compass the direction that the device is currently facing (alpha, z axis) (HTML5 Rocks, 2011). The alpha, beta, and gamma values returned represent the angle at which the device is being held, as a result of this the output is in degrees. Device Orientation could be used for the same purposes as device motion as they both have the ability to calculate the devices current orientation. Either one of the two could potentially be used in place of the other in the event that the device does not support one. For example, in the case that the device does not have a gyroscope, the device motion event could be used in its place. Functionality wise both the device orientation event and the device motion event can be used to achieve the same results (devices orientation). If in the case a device does not support device motion but does support device orientation, the device orientation results must be equalised in order to provide an equal experience across all devices. As the results from both events are different the controls of the games using the results may differ. In order to overcome this the results returned from both events should be equalised.

2.5.1.3 Canvas

Canvas is a HTML5 element which allows users to render graphics within the constraints of the canvas's dimensions. In the creation of the canvas element height and width dimensions are set, these dimensions act as a container for graphics. Graphics can then be drawn within this container with the use of a scripting language, most commonly this would be JavaScript (W3schools.com, 2014)[29]. The Canvas element is supported by almost all current mobile browsers, this includes, iOS Safari, Android Browser, Blackberry Browser, Opera Mobile, and Chrome for Android. The relevant versions of these browsers that are supported are shown in Figure 2.

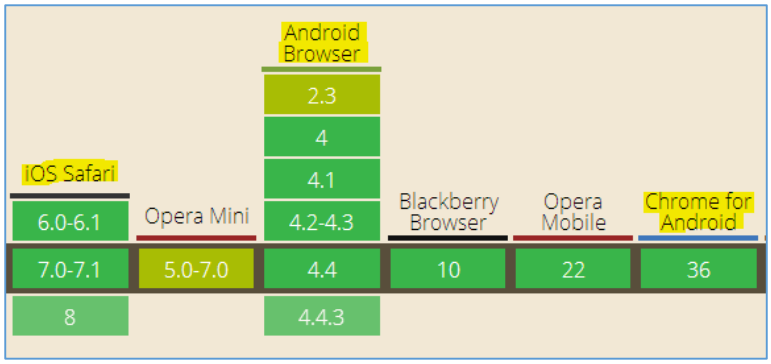

Figure 2: Mobile Browser Canvas support list. Source: Caniuse.com, (2014)[30]. The colour green signifies total support for this feature in these browser versions. The lime colour signifies partial support. Green indicates full support while lime represents partial support.

HTML5's Canvas is an essential tool in the creation of HTML5 and JavaScript based video games. As it allows for graphics to be rendered to a webpage it is an indispensable tool to be used in this project. HTML5's Canvas element is the most fundamental component in the creation of HTML5 and JavaScript based video games.

2.5.1.4 Offline Web Applications

A new feature introduced in HTML5 is the ability to run web applications while not connected to a network. In other words, the ability to run web applications while offline. This is achieved by setting up a manifest which specifies what data to store locally when required. It has already been possible to store data locally and thus achieve certain offline functionalities however, the offline web application features of HTML5 improve upon this significantly. Offline web applications comprise of two components, application caching, and offline storage. As articulated by Mahemoff (2010)[31], application caching involves saving an application's backend and frontend while offline storage involves saving data which has been explicitly created by a user or saving resources relevant to that user.

Users have been reaping the benefits of browser cache since its conception. The fundamentals of both browser and application cache are essentially the same however, the differences that do exist between the two set them apart significantly. In the context of the games to be developed as part of this project, browser cache could be used to store the files and documents required to run the web application (rehabilitation game). Through doing so, future launches of the game are essentially instantaneous, application cache can also achieve these results. However, this is where the similarities

between application and browser cache end. Browser cache is volatile and susceptible to being overwritten, conversely, application cache has special storing priorities on a user's local hard drive. This makes it far less volatile than browser cache. Furthermore, with application cache there is much more control over which files are saved and which files aren't when compared to the standard browser cache. Making use of application cache will allow for games to load quickly and thus improves the user's experience of the web application (Mahemoff, 2010; Pilgrim, 2014[32]).

As previously stated, application cache allows for the applications core logic and assets to be stored while offline storage allows for the storage of data relevant to a user. In the context of the games to be developed as part of this project, offline storage could potentially be used to store user information such as high scores and current progress etc. Although ideally this information would be stored on the cloud through uploading the relevant data, offline storage allows for users to continue progressing within the game world without being restricted to requiring a connection to a network. At the point of uploading the user's data to the cloud, copies of this data could be stored locally with the use of HTML5s offline storage. In the case that a user does not have a connection to the server that is hosting the web application, they would still have access to the application through HTML5's application cache. Furthermore, they would also have access to their current progress through offline storage. Both of these HTML5 features come together to provide a seamless offline web application experience. HTML5s offline web application features could greatly improve a user's experience of the web applications to be developed (Kesteren and Hickson, 2014[33]; Mahemoff, 2010).

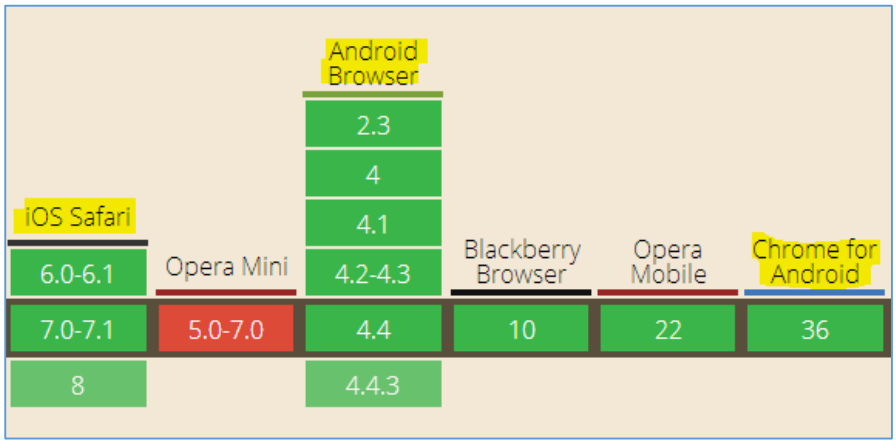

Figure 3 HTML5 Offline Web Applications Browser Support. Source: Caniuse.com, (2014)[34].)[35]. The colour green signifies total support for this feature in these browser versions. The red colour signifies lack of support. Green indicates full support while red indicates no support.

As shown in Figure 3, HTML5's Offline Web Application features are supported by most mobile browsers. More importantly however, they are supported by the mobile browsers relevant to this project, Chrome, Android Browser, and iOS Safari.

2.5.1.5 Drag and Drop

Drag and Drop is a feature which as the name suggests, allows users to select a virtual object and drag it across there screen to the desired location. This functionality has previously been achieved by older versions of HTML5 through external libraries. HTML5 allows for this function to be used without the need for external libraries as it is now officially part of the HTML5 standard.

The uses of the Drag and Drop feature are plentiful especially in the context of this project. This feature could potentially be used as a method of triggering stimulation in a user's fine motor skills. Users could be prompted to drag an object over a specific path in order to exercise their fine motor skills. The accuracy required to make these movements could be gradually improved by introducing different pathways which get more complex based on the users current skill. The Drag and Drop feature has potential use in the games to be developed as part of this project.

Unfortunately the HTML5 Drag and Drop feature is not supported by a majority of mobile browsers. Although this is true, the feature itself could still possibly be implemented on mobile platforms with the use of third party libraries.

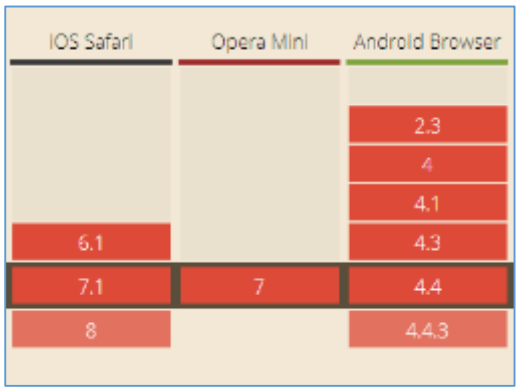

Figure 4: HTML5's Drag and Drop Feature Mobile Support. Source: Caniuse.com, (2014)[36]. The red colour signifies lack of support for this feature in these browser versions. Red indicates lack of support for these versions.

The mobile browser support for HTML5s Drag and Drop feature has been illustrated in Figure 4.

2.5.2 JavaScript

JavaScript is a web based programming language which is used by a majority of modern websites and is supported by essentially all modern web browsers (Flanagan, 2011)[37]. As stated by Flanagan (2011), modern game consoles, tablets, desktops, and smartphones include JavaScript interpreters and hence JavaScript is the most ubiquitous programming language to date. JavaScript allows web developers to have control over the behaviour of their web sites and for this reason, JavaScript skills are essential for every serious web developer. It is a high-level language and is dynamically typed, this makes it easier to understand and easier to work with. The JavaScript syntax is heavily based off of the Java programming languages syntax (Eich and Mckinney, 1996)[38].

The syntax of the JavaScript programming language is as previously mentioned heavily borrowed from the Java programming language. As a result of this, the core logic of the language comprises of the same building blocks as many other high level programming languages. This meaning almost all of the standard conditional statements, expressions, and constructs will be familiar to every developer. Conditional statements such as the 'if-then-else' statements act as the building blocks of applications. The design and syntax of the language makes it incredibly flexible and allows for the development of complete web applications.

In order to evaluate the effectiveness of this language in the creation of a mobile video game which is aimed at the rehabilitation of stroke victims, potentially helpful features of the language shall be analysed. This is to gauge their usefulness in this project.

2.5.2.1 Gyro.js

Gyro.js is a JavaScript library which aggregates the current methods of reading gyroscope and accelerometer data and combines this into a single object. This object can then be interacted with by the developer in order to read specific results and determine the browsers support for orientation/acceleration features (Gallacher, 2011)[39]. Gyro.js allows for dynamic interchanging of the data being read based on the features a browser/device supports. For example, Gyro.js can be tweaked to read acceleration data by default. However, in the case that the device does not have an accelerometer, the gyroscope can be used instead to read orientation data. Using this library allows for further platform and browser independence by removing the dependency of having a specific

sensor. With the use of this library, devices which only have either an accelerometer sensor or a gyroscope sensor will be supported without bias.

2.5.2.2 Game Engines; Software Frameworks.

It has already been established that JavaScript can be used as a complete programming language for the development of web applications, however, the development of video games is vastly different from standard web applications. Video game development without a game framework is a complex process which would require a developer to have a deep understanding of aspects such as 2D/3D graphics rendering, game world physics, artificial intelligence, networking, sound engines, and a plethora of other specialised fields. If each developer were to master these fields and create every game without the reuse of code, the current video game market would not exist. The pace of development and rate of delivery would be far too slow. Game Engines are frameworks which have been designed specifically for the development of video games. These engines provide core functionalities which allow the developer to focus on the development of the game itself, instead of focusing on setting up the pre-requisites of a video game. Game engines handle complex tasks such as the aforementioned functionalities so that the developer does not have to. Developers in essence make games through the reuse of code which has been written by the creators of the game engine. In order to be successful in the creation of web based video games a game engine which supports this development must be chosen. More specifically a game engine which allows for the development of video games within the JavaScript programming language is to be selected. It is possible to create a video game strictly through a combination of HTML5, CSS, and JavaScript however, this is unwise as it is complicates and slows down the development process. Furthermore, the reuse of code written by experts in the field of game engines increases efficiency and correctness.

2.5.2.3 *Phaser*

Phaser is a JavaScript based game framework which supports the development of desktop and mobile HTML5 video games. It was created by Richard Davey as a side project whilst working on the development of the Kiwi game engine. After leaving the development team of the Kiwi game engine, Richard Davey continued his work on the Phaser game engine. The inspiration for the creation of Phaser came from the frustration of not being able to implement what Richard thought to be excellent

additions to the Kiwi game engine due to the management's decisions (Davey, 2014)[40]. Although there is no explicit mention of the exact date Phaser was conceptualised, based on the public releases of the source code it can be said the first official release was in April of 2013 (GitHub, 2014)[41]. The latest release of Phaser is version 2.0.6 which was released in July 2014. As the release dates may suggest, Phaser is a relatively new game framework which is continually growing.

Phaser provides users with 2D Canvas/WebGL rendering, game physics, animations, sprites, various input systems, and a wide array of other features. Phaser has been selected over other JavaScript game frameworks not only because of its vast feature set but more importantly, because of its support for tablet/mobile browsers. Phaser was designed specifically for the development of video games that operate on mobile browsers. It is supported by all modern browsers that support the canvas tag however, mobile browsers are the main focus of the framework. Mobile browser support for the Android platform starts at versions 2.x and above. Support for the Apple iOS platform starts at iOS5 Mobile Safari and above (Phaser.io, 2014)[42]. Phaser makes use of the pixi.js rendering engine for all its rendering purposes. Pixi.js is a fast lightweight 2D library which was created to work across various devices. It achieves its fast speed by taking advantage of hardware acceleration (Graves, 2014)[43].

The Phaser framework is well documented, this allows for easy development and provides users with a sound understanding of the framework without using any third party sources for information. There are currently over 320 example implementations of the various features that Phaser supports on the official Phaser website. These are invaluable when evaluating to what extent this framework can help in the development of a HTML5/JavaScript based video game that aims to aid in the rehabilitation of stroke victims.

To summarise, Phaser has been chosen because of its explicit support for mobile game development, its performance on mobile devices, and its extensive feature set.

2.5.2.3.1 Graphics Rendering

Phaser allows for the rendering of 2D graphics more specifically, this rendering is done within HTML5's Canvas element. WebGL can be used in place of Canvas to achieve similar results. As previously mentioned, HTML5's Canvas element acts as a container for the graphics, Phaser in this scenario is used to render graphics within this container. It is important to note that Phaser provides the capability of automatically switching the renderer being used (Canvas or WebGL) to the most optimal one based on the browsers support. This provides a continuous experience for a user without

interruptions caused by lack of support from a browser. Phaser makes use the Pixi.js rendering engine to achieve its 2D graphics rendering. Pixi.js is a multi-platform rendering engine which is claimed to be the fastest engine for 2D graphics rendering in its field (PixiJS, 2013)[44]. Pixi.js allows for multi-touch interactivity on mobile and tablet devices. This is encouraging as multi-touch features could potentially be used in the rehabilitation of stroke victims. (Davey, 2014)[45]

2.5.2.3.2 Audio Capabilities

Phaser makes use of the Web Audio API and in doing so provides users with the ability to create audio sprites, stream audio, control volumes, and a wide array of other features available as part of the Web Audio API. As discussed by Rogers (2012)[46], Audio on the web had been underdeveloped up until the development of the Web Audio API. Prior to the Web Audio API, audio on the web was delivered through third party plugins such as Adobe Flash player. HTML5 introduced the Audio tag which allowed for basic audio playback via streaming from the site hosting the required files. This release is monumental as it significantly changes the dynamic of how audio can be delivered on the web. Although this is true, HTML5's Audio tag is severely limited to basic usage as it only offers single channel audio with no complex audio management and mixing features. These features are however available through the Web Audio API. In the context of video games, this element of complex audio management and mixing with multiple audio channels is required in order to provide a quality video game experience. Currently not all devices support the Web Audio API (primarily Android devices), in order to overcome this Phaser allows for the new Web Audio API to be used or the old Legacy Audio. These can both be used interchangeably at a developer's request, Phaser can also dynamically change between the two methods based on a devices capabilities (Rogers, 2012; Davey, 2014).

Through the use of the Web Audio API, Phaser is able to provide a rich audio experience which is suitable for video games. This will allow for a more immersive video game experience and consequently should be taken full advantage of in the creation of the game to be developed as part of this project.

2.5.2.3.3 Input Capabilities

Phaser supports all the commonly used input controls methods such as, mouse, touch, keyboard, and pointer controls. Similarly to Phaser's Audio and Graphics capabilities, controls can be interchanged automatically dependent on a devices capabilities without disrupting the flow of the game. The input capabilities of Phaser are very important as they are what allow a user to interact with the game world. A comfortable control scheme will allow for a more streamline gaming experience. Simplicity in the control scheme is required to achieve this comfortable experience. Stroke victims suffering from neurological damage may have difficulty overcoming the complexity of a difficult control scheme. Furthermore, if the control scheme is more mentally demanding than the game itself, results cannot be hoped to be achieved from the game itself. The user would simply be too distracted while focusing on the controls rather than focusing on their rehabilitation. One scenario where complex control schemes should be advocated would be when the control scheme is the driving factor for the user's rehabilitation. In the context of this project certain input control schemes can be used to exercise a stroke victim's fine motor skills (Davey, 2014).

2.5.2.3.4 Physics Engine

Physics engines are required in order to achieve collision between sprites, collision detection, and a wide array of other physics based functions. The game to be created as part of this project heavily rely on the use of a physics engine in order to achieve the desired results. Fortunately, Phaser supports physics based features through the use of three separate physics engines. The physics engine most relevant to this project is the Arcade physics engines which is an AABB physics engine. This makes it incredibly lightweight in comparison to other physics engines. As a result of this, it is the ideal solution for achieving game physics on low powered devices. Complex physics engines can be burden on mobile devices as the processing power required to carry out complex physics based tasks can be detrimental to the devices battery life. Furthermore, the need for such a complex physics system which provide accurate physics is unnecessary in the context of mobile web games, as long the in-game physics are perceptually accurate.

As articulated by Davey (2014), the basic Arcade physics engine which Phaser provides allows developers to apply gravity and velocity to sprites, these sprites can then be tested for collisions and separation. More complex physics tasks can be achieved through the use of the Ninja physics engine

or the p2.js physics engine. The complex physics capabilities offered by this engine are unneeded in the context of this project.

2.6 Summary

HTML5 and JavaScript

Through the undertaking of this background study the potential capabilities of HTML5, JavaScript, and the Phaser game framework have been realised. From a theoretical standpoint the languages and framework appear to support the capabilities required for the creation of a Tablet PC based platform and browser independent video game which is aimed at the rehabilitation of stroke patients. Theoretically, based on the information gathered in this section, from a technological standpoint the objectives of this project can be achieved. This however, will only be verified after the implementation stage.

3 Chapter 3: Design

This section of the report covers the design of the software solution. The design aims to model the proposed software solution in such a manner that it meets the previously outlined requirements. Design decisions will be made based on information gathered in the research and preparation stage, this design will then be used in the implementation stage for the creation of prototypes.

3.1 Ideal Application Outcome

Firstly an overview of an ideal final outcome will be outlined. This will allow for an evaluation to be made between the projected ideal outcome and the actual final outcome.

A stroke patient is advised by a rehabilitation specialist to make use of the game suite. The patient can then sign up for the service as a patient of the rehabilitation specialist. This will allow the carer to monitor and track a patient's progress. The user will be prompted to take an initial evaluation test which will consist of a short run of each type of game available. For example, a short run of games which can be used to evaluate fine motor, memory, and text to symbol recognition abilities. This initial evaluation run will allow a user and their carer to properly evaluate at what level the user is currently at in regards to their strengths and weaknesses with both their physical and cognitive abilities.

In the case that a user's evaluation has determined that a certain function is especially lacking, the software would softly prompt the user to play a game which aims to stimulate this function. There would be a score metric for users and carers to gauge how well a user is progressing. These scores will be periodically relayed back to the carer, at this point the carer could potentially make suggestions as to what the user should work on improving next. In the case that a carer realises that a patient is not employing the use of the software consistently (through checking time played in the logs that are relayed back to them), they could advise the patient to use the software more consistently. This aspect of relaying information back to the nurse creates a connection between both parties, this connection allows the nurse to gain a deeper understanding of the patient's recovery.

3.1.1 Suite of Games

Instead of developing a single web-application which stimulates multiple damaged functions, it is proposed that a suite of applications be developed. Creating this separation between the stimulation of different functions would allow users to focus on the rehabilitation of a specific function. For example, a user's physical functions may be significantly more damaged than their cognitive functions. Allowing this user to focus entirely on physical rehabilitation will allow them to benefit more than they would when being distracted by additional stimulation of other functions, when they don't require it. Splitting the stimulation of different functions into different application could potentially aid in keeping users interested. Instead of playing the same game over and over and eventually losing interest, the user will be able to switch between games and thus stay interested for longer. This is important as the longer a stroke patient works on their rehabilitation the more successful their rehabilitation would be.

3.2 Achieving Platform and Browser Independence

Platform independence cannot be achieved through the development of native applications for either the iOS or Android platform. This is clear from the fact that the development methods of both platforms differ. In order to create a software solution which is platform independent, web applications will be developed in place of native applications. These web-applications will be developed with the use of the HTML5 mark-up language and the JavaScript programming language. Instead of running the application directly on top of the operating system as a normal native application would, web-applications run from within a browser application. In other words, the web-application is launched from the web-browser which is turn in launched from the operating system.

Platform independence is assured under the basis that a platform has a native web-browser application which supports the aforementioned languages and all the required features. In order to be successful in the deploying of these web-applications on the iOS and Android platforms, the platforms must first implicitly support them through their native web-browser applications. Fortunately the default Android and iOS web-browsers which are preinstalled on the operating systems both support these languages, such is the case with all serious internet browsers.

Apple iOS's default browser is a mobile version of Safari and Android's default browser for devices running Android version 4 or above is Google Chrome. This has been highlighted as the design of the

web-applications to be developed must fit the constraints of the browsers which are to be supported. Specific versions of the Safari and Chrome browsers must be selected and developed for as different versions support different features i.e. an older version of a browser may not support a newer feature where a new version would. The versions to be supported in this project are version 33+ of Google Chrome and Mobile Safari version 5+.

3.3 Targeting Stroke Patients

The target user base for the video game applications to be developed are stroke patients. Knowing this the design of the software must be created to fit this constraint. Factors which must be taken into account to achieve this include the applications user interface and usability.

3.3.1 User Interface

The user interface of the software should be designed in such a manner that it can used by those affected by the typical stroke side effects. Common side effects include, visual perception and stroke induced tremors. According to Stroke.org.uk (2014)[47] the chance of having a stroke is increased as a person ages. Furthermore, approximately 75 percent of strokes that occur are in people older than 65 years of age. These statistics allow for further decisions concerning the user interface design to be made. In order to properly design this software's user interface there must be an appreciation for the fact that the user base will be suffering certain impairments and will predominantly be of an older persuasion.

The user interface should be clearly labelled and simplistic. A simple layout will allow for those of all ages to make use of it, a layout which is cluttered and not properly labelled may lead to certain users becoming lost when trying to navigate through the software. Having a layout which is too complex and cluttered may put users off from using the application, not only because it would be uneasy on the eyes but also because some may genuinely struggle to navigate through the software as a result of neurological damage caused by a stroke. Having too much text displaying at once could potentially confuse those affected by neurological damage or visual impairments. Furthermore, factors such as button size must also be taken into account. Small buttons could be difficult to see or differentiate from the background for those who have damaged visual perception. Small buttons could also be

problematic for those with damaged fine motor skills as they may have trouble interacting with them. The precision required to click small buttons can be damaged by a stroke and thus larger buttons to interact with are required. As discussed by Tiwar, Astheimer and File (2004)[48] older users can have difficulty in interacting with multiple components at the same time i.e. multi-touch buttons. Anything which requires complicated manoeuvers should not be present as part of the user interface.

3.3.2 Cognitive Stimulation

It has been established that through physical stimulation the brain is directly affected by processes such as Neuroplasticity. This knowledge allows us to come to the conclusion that cognitive stimulation is not only activated through cognitive based activities but also through physical activities. This information should be made use of when designing the suite of games in order to fully take advantage of processes such as Neuroplasticity. In order to fully activate cognitive stimulation, some form of physical stimulation should be employed in tandem with cognitive stimulation methods.

One of the cognitive abilities damaged by strokes is a victim's memory. Memory can be broken down into two sub categories; verbal memory and visual memory. Verbal memory is the memory of things related to words such as the names of different objects. Visual memory on the other hand is the memory of how well a person remembers things that they have actually seen, for example, faces and objects (Novitzke, J, 2008)[49]. Cognitive stimulation should be achieved through the exercise of visual and verbal memory.

3.3.2.1 Memory Drag

Memory Drag is one of the games in the suite which aims to stimulate cognitive abilities. The basic outline of the game is as follows. A user will be presented with a screen which displays eight objects in total. Four of these objects will be aligned to the left side of the screen in a column and the other four will be aligned to the right side of the screen in a column. Objects on the left side will have some correlation/link with the objects on the right side of the screen and vice versa. For example, a teacup may appear on the left side and a teaspoon on the right side. Each object will be placed in a random row in the column on their side. This meaning, a teacup may appear at row one in the left column however, the teaspoon in the right column can appear in any of the four possible rows. This provides an element of randomisation which creates a fresh experience with every run of the game. Once a user has created a correlation between two objects they can drag an object from the left column onto an object onto the right column in order to solidify their decision. If they have made the right correlation between the objects the user will be rewarded with a point.

This game exercises a user's cognitive abilities by allowing them to create mental links between objects. These correlations may seem obvious to someone unaffected by a stroke however, damage to the brain caused by a stroke can potentially damage a user's visual memory and this damages their ability to recognise certain objects. A stroke patient may recognise something like a teacup however, they may not recognise a teaspoon. Furthermore, where a normal person would create the link between the two objects a stroke patient may not as their damaged visual memory prevents them from doing so. They may recognise the objects themselves but not what they are used for. This simple exercise allows a user to stimulate their cognitive abilities through exercising their visual memory.

The fundamental basics of the game have been illustrated in figure 5.

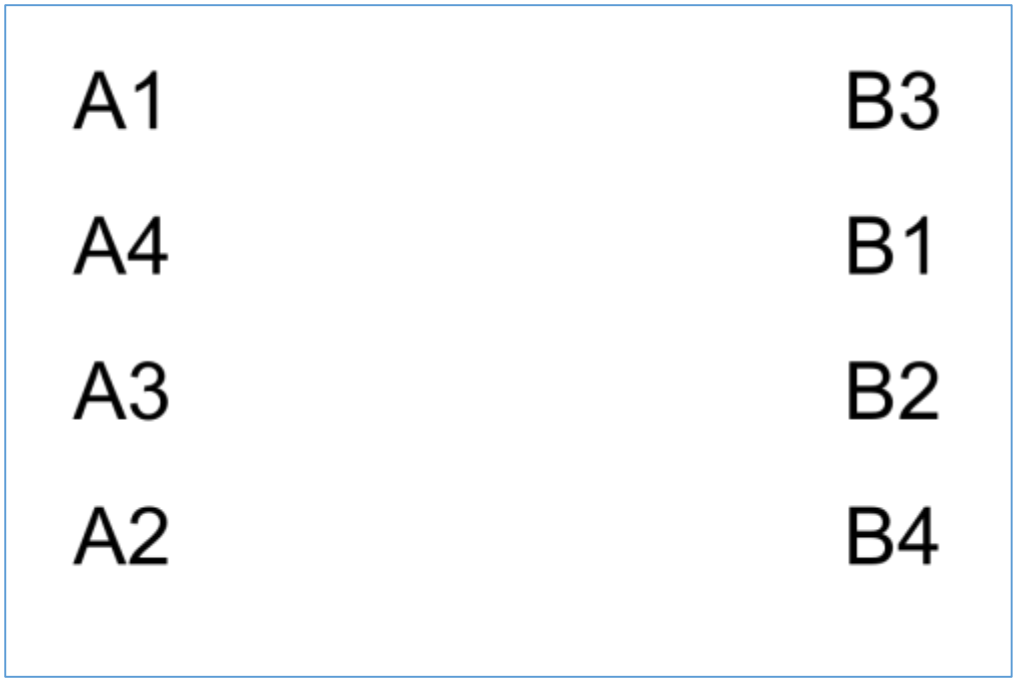

Figure 5: Memory Drag Concept

Each element in Figure 5 represents a real world object. The left column (column A) stores objects which correlate with those in the right column (column B). An example of this would be, A1 and B1 are both objects which correlate with one another i.e. a teacup and a teaspoon. The user will be prompted to match the objects by dragging an object from one column onto the matching object in the other column. For example, dragging object A1 onto object B1 or object A2 onto object B2.

As previously mentioned making the correlation mentally between two objects allows a user to exercise their visual memory and thus cognitive stimulation is achieved. Furthermore, actually clicking an object and then dragging it to its intended location allows for fine motor skills to be activated implicitly. The task of using your finger to pinpoint a specific object and then drag it across the screen is not too strenuous yet it still stimulates a user's fine motor skills. This action is so subtle and simple that a user will not feel overburdened when carrying it out. This additional physical stimulation activates Neuroplasticity in the brain and thus further improves cognitive abilities (Hammond, 2002).

This game can also be reengineered in order to focus on the rehabilitation of verbal memory over visual memory. This can be achieved by making use of the names of objects (text) in place of actual images of the objects. For example, the left column could have A1 set to the text 'teacup' and the right column could have B1 set to a picture of a teacup. Making use of both text and images will allow users to exercise both their verbal and visual memory.

3.3.2.2 Memory Pair

Memory pair is another game in the suite of games which aims to stimulate cognitive abilities. This game is fundamentally the same as Memory Drag however, the differences between it and Memory Drag, allows it to stimulate the same cognitive abilities whilst also being able to stimulate deeper fine motor skill abilities.

The general premise of the same is the same as Memory Drag. A user is shown two columns of objects and the must make the correlation between object. Instead of simply dragging an object over to the other in order to finalise their decision, the user must select the two objects which they feel correlate with one another, this is done by clicking on them. If the correct objects are chosen, a randomly generated curvy line will appear between the two chosen objects. At this stage in the game the user must follow the line with their finger from one column to the other. The user must follow the line accurately at the risk of losing of points when they do not do so. On completion of following the line users will be awarded points and they can then move onto the next pair of objects. This concept has been illustrated in figure 6.

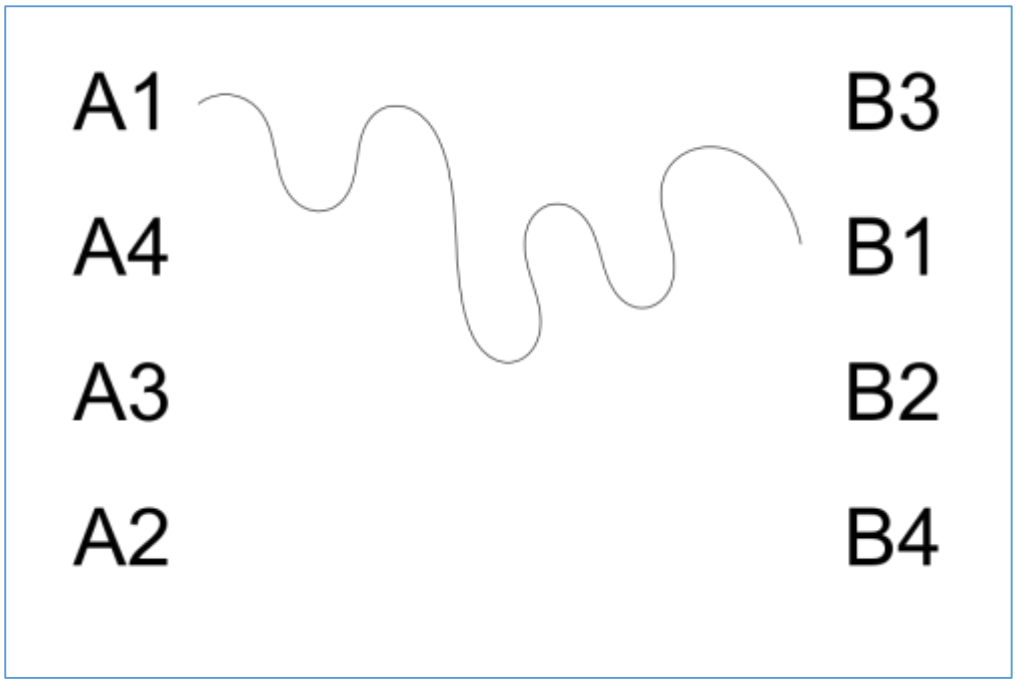

Figure 6: Memory Pair Concept

Similarly to the Memory Drag game, this game also stimulates cognitive abilities through exercising a user's verbal and visual memory. More importantly however, the dragging action to be carried out in this exercise is significantly different from the one carried out in Memory Drag. Instead of making use of very subtle fine motor skills, this game aims to provoke a user's physical abilities much more than Memory Drag. A great deal of precision is required in the actions to be carried out by the user. Following the line with your finger while under the pressure of losing points by moving your finger off the line, adds the additional cognitive strain of focusing on your physical movements. This multi-stimulant factor of the game allows another dimension to be added on top of the basic object correlation game. A user should be prompted to play one of the two games dependent on how strong their cognitive and physical abilities are. Making use of both would be optimal, a gradual improvement made through playing the first game should allow a user to progress onto the second game and further develop their abilities.

3.3.3 Physical Stimulation

After having a stroke a victim can be affected by multiple impairments, damage to a person's fine motor skills is an especially common impairment. For this reason the primary physical ability which shall be focused on for rehabilitation is fine motor skills. Fine motor skills can be exercised through almost all basic hand and finger movements. Furthermore, another physical ability which can be damaged is a person's visual perception. Simply looking at the display of a device could potentially count at visual perception stimulation. Although this may be the case, in order to provide proper rehabilitation and improvement to the user it is proposed that more complex tasks which require varying levels of hand-eye coordination be carried out.

In order to provide the best possible rehabilitation to both of the aforementioned physical abilities the natural synergy between visual perception and fine motor skills should be taken advantage of. Hand-eye coordination is a skill which makes use of both of these abilities, what a user perceives with their eyes provides them with enough information to make useful physical movements.

3.3.3.1 Dropper

Dropper is a game which aims to stimulate a user's physical abilities, more specifically, visual perception, fine motor skills, and physical reaction time. The general concept of the game is as follows. A user will take control of a player sprite which to start with will continually fall down the screen. The game starts when a 'pipe' is displayed, from this point it is the users goal to score as many points as possible through keeping the player sprite on the screen. This meaning, if the player sprite falls through the bottom of the screen (off screen), the game will end. Similarly, if the sprite is pushed through the top of the screen the game will end and the user will stop scoring points. Pipes are randomly generated and continuously move upwards in an attempt to push the user out of the screen and thus ending the game. The user combats this by navigating the sprite through the gaps in the pipes. The controlling of the sprite is to be achieved with the orientation of the device. If the user wishes to move the sprite to the right they must rotate the device to the right, if they wish to move the sprite to the left they must rotate the device to the left. This concept has been illustrated in figure 7.

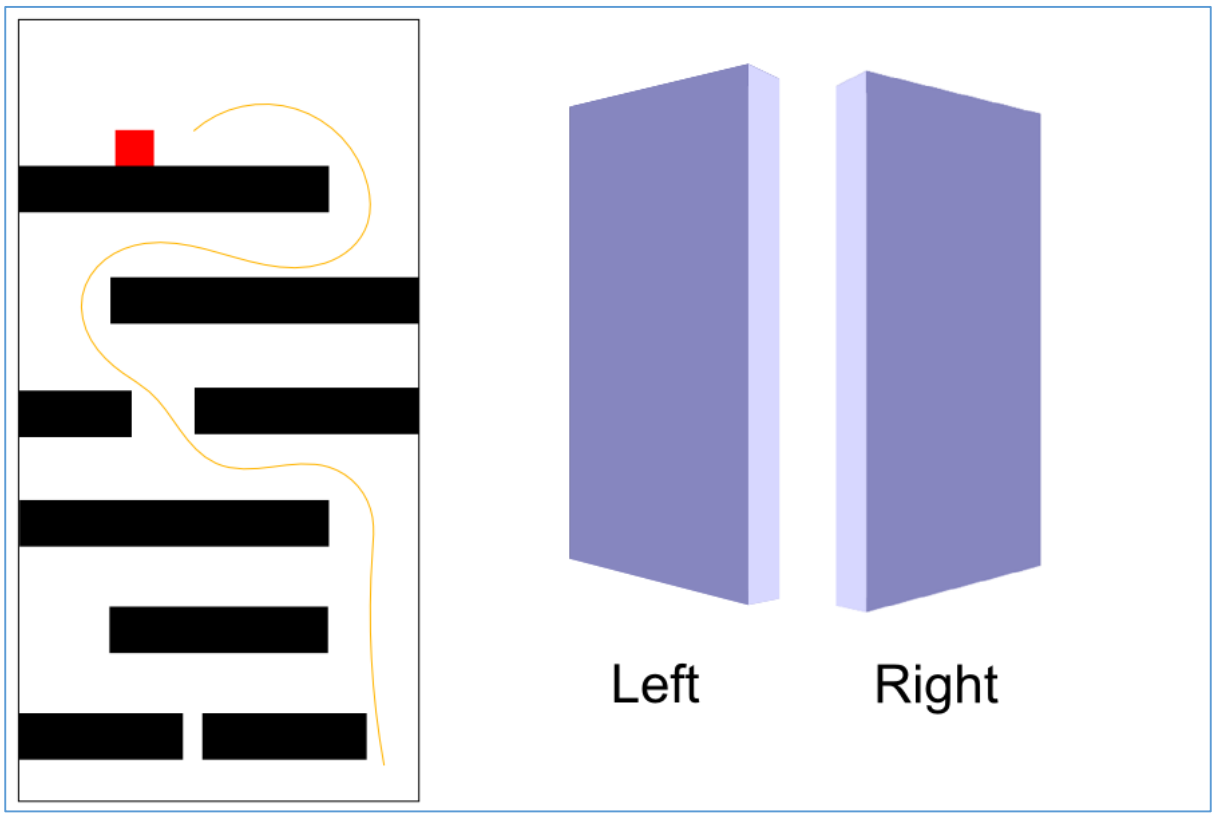

Figure 7: Dropper Concept

The red square in figure 7 represents the user controlled sprite, the black rectangles represent the pipes which will continually move upwards. The orange line simply demonstrates the movements the user would have to carry out with their device. As shown by the 3D models of a mobile device (see figure 7), when the device is orientated towards the left side, the sprite should move to the left. When the device is orientated to the right side the sprite should move to the right. As demonstrated by the orange line in figure 7, the movements required of the user can be either subtle or sharp based on the pathway generated by the pipes. The sharpness of the movements could potentially start very low and then gradually increase as the game goes on. This sharpness would be controlled by changing the distance between gaps of the pipes. Changing the sharpness could give users a sense of progression and at the same time this will be excellent for gradually improving a user's fine motor skills. Furthermore, the speed at which the pipes rise could be slowly increase as the game goes on, once again adding a sense of progression and gradually improving a user's fine motor skills as they get better at the game.

The controlling of the sprite through the devices orientation requires precise coordinated hand-eye movement. Based on what the user sees they will have to make a decision on which way to move the sprite and then perform the required action. Physical stimulation is achieved through these actions. An array of impairments can be potentially exercised through using this game. Impairments such as, sluggish reflexes, damaged fine motor skills, disrupted hand-eye coordination, and impaired organisation skills. The gradual increase in difficulty of the game should progressively improve upon these damaged impairments.

3.3.3.2 Roller

Roller is another game designed to stimulate physical abilities, it builds on the same ideas as Dropper. The outline of the game is as follows. The user is presented with a level which contains randomly placed star shaped sprites. The user must take control of a ball sprite and manoeuvre it through the level in order to collect all the star sprites. The user controls the ball sprite with the use of device orientation as in Dropper. The primary difference between this game and Dropper is the increase of difficulty but reduction in pressure. As the user is required to make use of all angles of orientation of the device they are required to use the full range of motion of their hand. Instead of simply moving left and right as in the Dropper game they are required to move left, right, up, down, and diagonal. This requirement to move in all of these directions adds another layer of difficulty when controlling the sprite. When a stroke patient's fine motor skills are damaged it can be difficult for them to make use of the full range of motion in their hand. Dropper is forgiving and allows users to make up for their lack of range of motion by using gross motor skills instead of fine motor skills. This meaning, a user could potentially orient the device left or right with the use of their arm instead of their hands and fingers, thus bypassing the need for fine motor skills. This is detrimental to patient as they won't be exercising their fine motor skills and instead of trying to rehabilitate these abilities they have bypassed their use. This was permissible in Dropper as there was the added pressure of the pipes continually moving upwards and the quick decisions the user was required to make.

The same cannot be done for Roller as the up, down, and diagonal movements required to be made by the user are much more difficult to make with the use of gross motor skills. Thus the user is forced to make use of their fine motor skills. For someone who has damaged fine motor skills it can be difficult to make use of the full range of motion of their hand without fully concentrating on the task. In order to allow for the user to fully focus on the orientation of the device and the movement of their hand, the elements of the Dropper game which added a sense of pressure to the user have been removed. There is no way for the user to lose in this game, they can only progress. This concept has been illustrated in Figure 8.

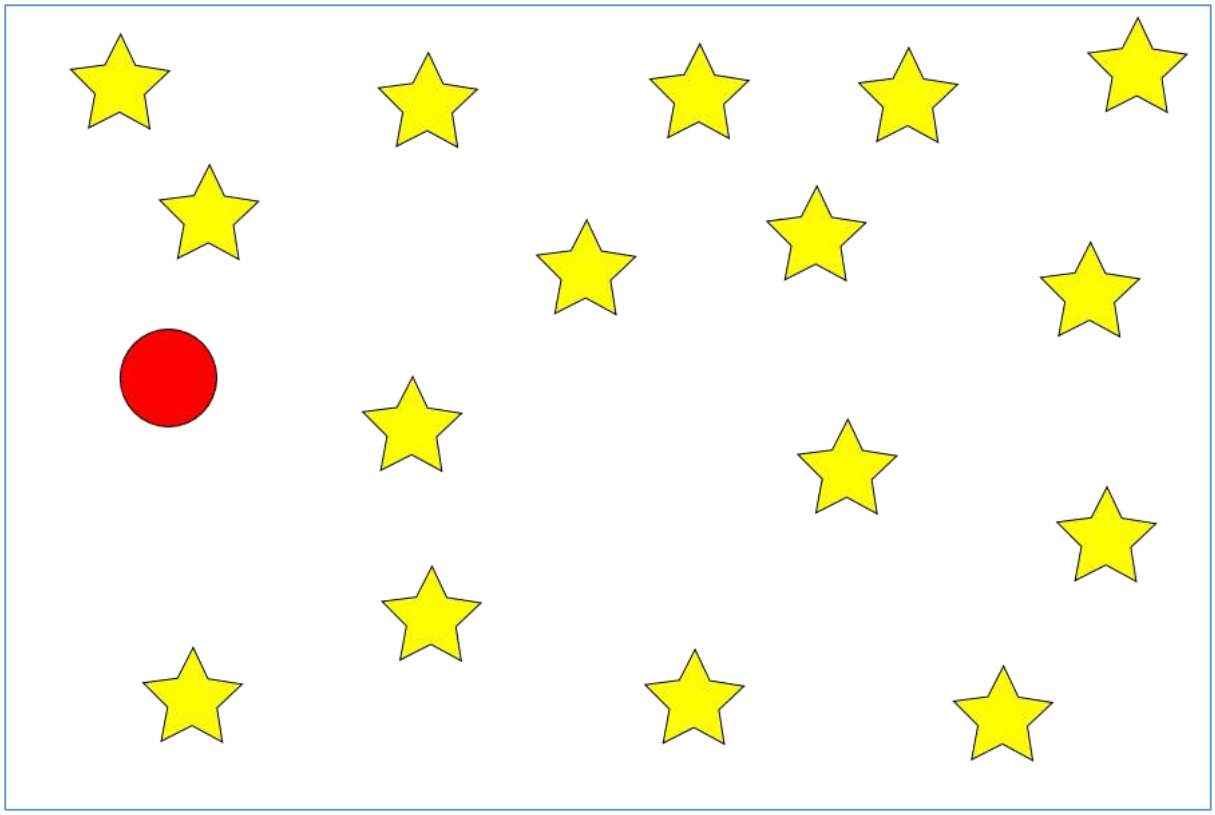

Figure 8: Roller Concept

The red circle in Figure 8 represents the player controlled sprite and the yellow stars represent the sprites which have to be collected by the user in order to score points.

3.3.3.3 Tilt Maze

Tilt Maze is a continuation of Roller which aims so increase the difficulty for the player while adding additional gameplay mechanics. The screen will be populated with various different sprites which will be randomly placed. For example, different types of vegetables will populate the screen. Amongst these sprites there will be one that is distinctly different from the rest. For example, a fruit instead of a vegetable. The user will control the player sprite with the use of the device orientation, the user's goal is to navigate past the randomly placed sprites (vegetables) in order to collect the sprite which is distinctly different from the rest (fruit). The randomly placed sprites can potentially be made immovable thus making a roughly generated maze for the player to navigate through. Touching the randomly placed sprites shouldn't affect a user's score but should cause both sprite to collide and thus disrupting the user's navigation, this will make the player's journey to the distinctly different sprite more challenging. This concept has been illustrated in Figure 9.

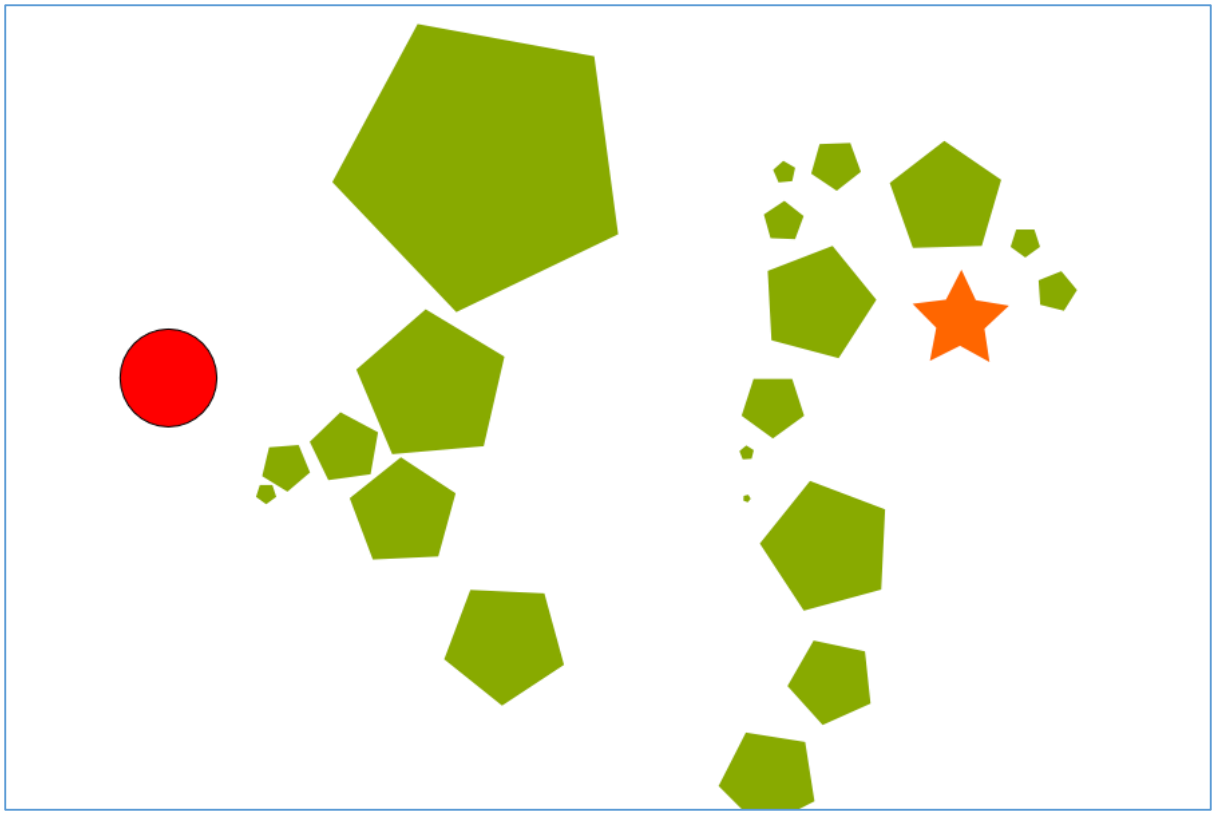

Figure 9: Tilt Maze Concept

The red circle in Figure 9 represents the user controller sprite. The green pentagons represent the randomly generated sprites. These sprites will be immovable, this meaning, the user controlled sprite will collide with them on contact. These sprites create a rough path/maze for the user to navigate through. This element of random generation provides the user with a fresh experience with every play through of the game. The orange star represents the distinctly different sprite which is to be collected by the user in order to score points.

4 Chapter 4: Implementation

This chapter documents the implementation of the previously outlined prototypes. This implementation is to adhere to the designs outlined in the design stage. Instead of discussing every line of code written, only relevant code and the code required to emulate these prototypes has been discussed.

4.1 Memory Drag

The first prototypes to be developed is the Memory Drag game. Firstly, an HTML5 canvas had to be generated, this canvas is the most fundamental component in the creation of HTML5 and JavaScript based video games. Rather than use the traditional HTML5 methods of creating a canvas element, the Phaser framework methods were used. Phaser allows users to automatically generate a HTML5 canvas element to which graphics can be rendered through the creation of a Phaser game instance. In order to create a working Phaser game instance there is certain code to be written which acts as the boilerplate code for all Phaser games.

In order to create the template which all the prototypes could use as their starting point, two files were first create. These being, the index HTML file, which acts as the frontend with which the user will interact with and the JavaScript file, which holds the backend game logic from which the Phaser framework is made use of. The HTML file will be referred to as index.html and the JavaScript file will be referred to as main.js for the remainder of this section. It should be noted that the Phaser framework file which contains the source code for the Phaser framework must be present in the same directory as files from which an instance of the Phaser object is to be created.

The code written in main.js which is required to create an instance of the Phaser object is shown in Figure 10. Firstly an instance of the Phaser game is assigned to a local variable. A new instance of a Phaser game is called within the code and the dimensions required are set (800 x 600 pixels), next the researcher specified how graphics should be rendered, in this instance, this was set to ‘AUTO’ which automatically choses how graphics should be rendered. By default WebGL is used to render graphics to however, if the device being used supports HTML5’s canvas element, canvas will be used instead. ‘AUTO’ is set as it saves from dealing with compatibility issues with devices which do not support HTML5’s canvas element. The final parameters of this code references functions which are essential

in operating Phaser, these being, preload, create, and update (Davey, 2013[50]). A HTML5 element is then to be created in the HTML5 index file which shall contain the canvas generated by Phaser. In the case of the games created as part of this project the canvas was held within a HTML5 div element.

```
var game = new Phaser.Game(800, 600, Phaser.AUTO, 'game-div', {
preload: preload, create: create, update: update });
```

Figure 10: Phaser object creation code.

The aforementioned functions were then created as they are mandatory in the creation of any Phaser based game, these functions being, preload, create, and update. These functions are executed at specific times during the operation of the game (Davey, 2013)[51]. The preload function is first function to be executed, this function is responsible for the loading of game assets such as images and sounds. The create function is executed directly after the preload function and is responsible for the creation of the assets which were previously loaded. The third reserved Phaser function used, is the update function. This function is called before the rendering of every frame, this typically means it is called every 1/60th of a second or in other words 60 times a second. Usually this function is used for things such as collision detection and other actions which require polling. The next step for the developer at this point was to populate these functions in order to create some semblance of a game.

Firstly the preload function had to be populated. The assets to be loaded were specified through the code shown in Figure 11.

```
function preload() {

    game.load.image('teaspoon', 'teaspoon.png');
    game.load.image('teacup', 'teacup.png');
    game.load.image('shoes', 'shoes.png');
    game.load.image('socks', 'sock.png');
    game.load.image('television', 'television.png');
    game.load.image('remote', 'remote.png');
    game.load.image('hat', 'hat.png');
    game.load.image('head', 'head.png');

};
```

Figure 11: Reserved Phaser function used to load assets.

As demonstrated in Figure 11, within the preload function each object to appear in the game screen has been loaded through the use of the previously created game instance and the correct calling of methods. Eight objects (sprites) were loaded, each sprite represents a real world object and each sprite has one other sprite with which it shares a relationship with i.e. teaspoons and teacups correlate with one another. In total there are four matching pairs. The design of this game requires four sprites to be positioned on the left side of the screen and the other four sprites which share relationships with those on the left will be placed on the right side of the screen. This has been illustrated in Figure 5.

Before the sprites were rendered and positioned onto the canvas, the logic for the game code had to be created. This meaning, the sprites had to be created in such a manner that allowed for the game to function as designed. Firstly the eight sprites were split into two categories, one category to hold the sprites that were to be align on the left side and one category to hold the sprites that were to be aligned on the right side. Identification numbers had to be assigned to each sprite so that the relationships between the matching sprites could be recognised through the code.

Secondly the potential positions for each sprite were mapped to two different arrays, one for the left column and the other for the right column. These arrays each contained four potential positions across the y axis in which the sprites could potentially be placed. In order to fulfil the design requirements each sprite on each side must be placed on at least one of the four specified locations, this would result in a full column on either side. The contents of these arrays were then rearranged randomly so that when the sprites were created they would be randomly placed. The rearranging of these arrays was carried out with the use of a Phaser framework method named 'Shuffle', this is demonstrated in Figure 12. This method simply rearranges arrays into random order.

```
ys1Array = Phaser.Math.shuffleArray(yArray);
ys2Array = Phaser.Math.shuffleArray(y2Array);
```

Figure 12: Phaser Shuffle method for rearranging arrays

The code required for the placement and rendering of the sprites was to be carried out in the create method. This method is used to set the initial scene of the game. Two arrays were created in the create method which held the names of the sprites which were previously loaded, one array held the objects which were to be positioned on the left side of the screen and the other held the objects to be positioned on the right side of the screen. At this point the possible positions, and sprites that can be

placed in these positions were configured correctly. In order to tie the two together the actual creation of the sprites commenced. In order to create the sprites and render them onto the screen the method shown in Figure 12 was used.

```
    for (i = 0; i <= 3; i ++) {

        var spriteName = "sprite" + i;
        spriteName = game.add.sprite(50, ys1Array[i],
leftSideSprites[i]);
        spriteName.id = i;
```

Figure 13: Sprite selection, creation, positioning, and Identification assignment.

This Phaser framework code allows for the creation of sprites through specifying their position on the x and y axis followed by the name of a sprite which was previously loaded. 'ys1Array' in this code is the array which holds the four possible positions that the sprites can be placed in. As the 'for' statement only loops four times and there are only four values in the position array, every position can and is used only once, so there are no overlapping sprites. The x axis is statically set to 50 pixels as the four sprites being generated in this piece of code were only to be aligned to the left side of the screen, there is no need to change this value. Furthermore, the last parameter specified in the creation of a sprite is the name of sprite which you wish to create, this sprite must have been previously loaded in the preload method.

As previously mentioned, arrays were created which were to hold the previously loaded sprites. These preloaded sprites are used in the creation and rendering of sprites. Once again as the 'for' loop only loops four times and there are only four possible sprites which can be added (from each array), every sprite is added properly without any duplicates. The final action carried out in this process was the assignment of the identification numbers. This was simply carried out through the use of the Phaser ID method.

The sprites to be aligned to the right side of the screen were done so through the same method however, the x axis value was set differently to allow them to be aligned to the right. Naturally the array which held the right columns sprites was used instead of the left column sprite array. In the completion of this stage each sprite was correctly placed in their columns, they also had the correct identification numbers attached to them in order to allow for relationships between sprites in the left column and sprites in the right column to be identified. This has been illustrated in Figure 5, the letters

in each column of Figure 5 represent the objects and the objects they match, the numbers represent the identification numbers which were assigned to each object.

At this point only the scene had been set, next user control had to be configured in order to allow for the sprites to be dragged across the screen onto the matching sprites. In order to achieve this, a physics engine must first be enabled on the sprites which are to be moved. This is achieved by simply using the enable physics method from the Phaser framework. This method takes two parameters, the sprite to enable physics on and the type of physics engine to use. In this case the Arcade physics engine was used as there was no need for more complex physics operations. The performance offered by this physics engine is excellent for use on tablets and other mobile devices.

After physics have been enabled on both sets of sprites (left and right column), drag functions can then be applied to the left column. There is no need to grant the sprites on the right side the ability to be dragged as the games design dictates the sprites in the left column are to be matched with the sprites on the right column, rather than the other way around. In order to enable user input on the sprites, this must be specified with the use of another Phaser function, this being the input enabled function. Next the actual drag function were enabled, this was again done through the use of a Phaser specific method this method being, the enable drag method. At this point each sprite placed in the left column could be dragged where pleased. In order to determine whether or not the sprite being dragged has been placed on top of a matching sprite events listeners and an overlap check had to be implemented. The code used to enable input, dragging, and the listeners is shown in Figure 14. There are two event listeners which are implemented, one is for when the dragging is initiated and the other for when the dragging stops.

```
spriteName.inputEnabled = true;
spriteName.input.enableDrag();

spriteName.events.onDragStart.add(startDrag, this);
spriteName.events.onDragStop.add(stopDrag, this);
```

Figure 14: Enabling input, dragging, and listeners through the Phaser framework.

The ‘startDrag’ method returns the sprite that has been clicked (currently being dragged), this sprite can then be used to identify the matching sprite in the opposite column. The identification numbers assigned to each sprite were strategically mapped so that each pair of objects shared the same identification number. For example, if teacups ID was 1, teaspoons ID was also 1. Through using the

information gathered from the clicked sprite the matching sprite could be determined. Both of these sprites were then stored in global variables so that they may be accessed later.

The 'stopDrag' method makes use of the aforementioned variables which store the two matching sprites. This is then used to perform an overlap check between the two sprites. This overlap check is achieved through the use of a function which is part of the physics engine used by Phaser. This function can take a total of five parameter however, in the context of this game only 3 were required. Firstly the two sprites which are to be checked for an overlap are specified. The global variables storing these sprites were used here. The last parameter specified, was the name of the overlap handler method. This method is executed whenever there is an overlap between the two specified sprites. The overlap check is performed purposefully when the sprite has stopped moving, this is because the act of no longer moving the sprite and purposefully placing it somewhere else on the canvas acts as solidification of a decision. The check is then carried out to determine whether or not there has been an overlap between the sprite being dragged and its matching pair.

```
function startDrag(clickedSprite) {

    sprite1 = leftSpriteArray[clickedSprite.id];
    sprite2 = rightSpriteArray[clickedSprite.id];

}

function stopDrag(clickedSprite) {

    game.physics.arcade.overlap(sprite1, sprite2, overlapHandler,
null, this);

}
```

Figure 15: startDrag and stopDrag methods. Checking for overlaps.

The overlap handler is used to remove the matching sprites from the playing field and increase the player's score in the case that they have correctly placed the sprite. The removal of the sprites is achieved through the use of the 'kill' function which is part of the Phaser framework. In order to display the score, text had to be rendered to the screen. This was achieved by using an inbuilt text Phaser function which adds text after specifying, the position, style, and text to be displayed. This code was placed into the create function. The rendered text holding the score was then updated whenever there was a change of score within the overlap handler. After matching all four pairs the user is congratulated through the text rendered onscreen.

4.2 Memory Pair

Memory Pair is the second game designed to stimulate damaged cognitive abilities. This game builds on the subtle stimulation of fine motor skills from the previous game by adding elements of mental focus and requiring more controlled user input. The fundamentals of this game are almost the exact same as Memory Drag, the user is to match objects which share a relationship in order to improve upon visual and verbal memory. Where this game differs from Memory Drag is in the how the matching is done. Instead of simply dragging an object to its matching pair, the user must select the two objects that match. In doing so a randomly generated curvy line is to be created between the two objects. The user must then follow this newly create line accurately with their finger. This builds upon the subtle fine motor skills stimulation offered in Memory Drag however, there is an additional layer of complexity.

The implementation of this game is very much the same as Memory Drag as the fundamental concepts between them are the same. The exact same steps that were taken in the implementation of Memory Drag were carried out in Memory Pair. This was done up until the point of enabling user input as the user input elements of this game are carried out differently from the previous one. Once the basic template of the game has been created it closely resembles the illustration shown in Figure 5.

As implemented in Memory Drag, input had to be enabled on the sprites during their creation followed by the attachment of an event listener specific to the type of input that is desired. In the case of this game the user is required to click (touch) two sprites in order to make a selection. Using the generic ‘click’ event does not allow for touch events to be detected on tablets. The click event in Phaser is only relevant on desktops and laptops. In order to allow for touch events and click events to register on the sprites the ‘on input down’ event listener was added to each sprite. This allows for input down events regardless of their source to be registered. This meaning, desktop mouse, laptop pointers, and tablet touch interfaces all register, this grants further cross platform abilities. Figure 16 shows the code which was used to enable input and attach an event listener to the sprites.

```
    for (i = 0; i <= 3; i ++) {

        sprite = game.add.sprite(50, ys1Array[i],
leftSideSprites[i]);
        console.log(leftSideSprites[i]);
        sprite.id = i;

        // Enabling input of each sprite
        sprite.inputEnabled = true;

        // Add the callback to the input down event
        sprite.events.onInputDown.add(listener, this);
        }
```

Figure 16: Enabling input and attaching the 'onInputDown' event listener.

Within the code to enable the 'onInputDown' event listener a listener method is specified. This method is then called whenever an 'onInputDown' event is triggered.

The listener method had to be coded in such a manner that a detection could be made when clicking on a sprite, this detection would check to see whether or not the first object in a pair was selected or whether or not the second object in a pair was select. From this information, whether or not the user had made the correct decision could be calculated. The code used in the event listener method can be seen in Figure 17.

```
function listener(clickedSprite) {

    if (spriteId === clickedSprite.id) {
        line1 = new Phaser.Line(xSprite + 25, ySprite + 25,
clickedSprite.x + 25, clickedSprite.y + 25);
        console.log("Match");

        game.debug.geom(line1);
    }

    else {
        spriteId = clickedSprite.id;
        xSprite = clickedSprite.x;
        ySprite = clickedSprite.y;
    }
}
```

Figure 17: Event listener used to check for matches and generate a line between the two sprites.

Firstly an 'if' statement is used to check whether or not the local variable 'spriteId' (which holds a sprites ID) matches the clicked sprites ID. If there is not a match between the identification numbers, the clicked sprites ID is assigned to the global variable 'spriteId'. Furthermore, the x and y axis values

of the clicked sprite are stored in two more global variables. This process can be seen in the 'else' statement in Figure 17. Assigning these values to the global variables allows us to save the information of the first clicked sprite, this information can then be used in the 'if' statements condition to check if there is a match. In the case that there is a match, a new Phaser line object is created. This line object takes four coordinates in its creation, the first two act as the starting point for the line to be generated and the second two points act as the ending point. The first two coordinates given are the x and y axis values of the first sprite that was clicked, the second two coordinates are the x and y axis values of the second object clicked.

To finalise the creation of the line, the debug geometry method is used which takes the previously created line as a parameter. After the completion of this, the user could now select sprites and the code was adequate enough to determine whether or not there was a match between the two. The 'if' statements condition is passed if the first clicked sprites ID matches the second clicked sprites ID. As all pairs of objects which share a connection share the same identification number (as designed previously), it is very easy to determine whether or not there is a match between sprites. It should be noted that at this point with the use of the debug geometry function the line between the sprites has only been created but not rendered. In order to finalise the rendering of the line the Phaser reserved 'render' method must be used. The render method is used to carry out additional rendering out with the initialisation of the game done in the create method.

```
function render() {

    game.debug.geom(line1);

}
```

Figure 18: Phaser reserved render method.

As these lines being generated are created out with the create method, the render method must be used. This method works similarly to the update method as it is executed sixty times a second however, it is specifically used for rendering graphics and post-processing effects instead of handling polling events. Within this method the same code to initialise the creation of the line must be written, this allows for the line to render to the screen. This can be seen in Figure 18.

4.3 Dropper

Dropper was the first prototype in this project to be created which was targeted at the rehabilitation of physical abilities, more specifically, the rehabilitation of damaged fine motor skills. This was to be achieved by controlling the player sprite through the devices orientation. This would require them to make use of their fine motor skills in order to advance in the game. As with the two previous prototypes, the first step in the development of this game is the level creation and game logic creation. In order to meet the demands of the previously outlined design, the scene must be set and the games code must be engineered in such a way that it allows for all the features in the design stage to be implemented.

As with all Phaser games, firstly the boilerplate code had to be initialized. The Phaser object which generates the canvas on which graphics can be render was first created and the dimensions of the canvas were specified. The three required Phaser methods were then also added. Next, the reserved preload method had to be populated. In this method the preferred background colour was specified with the use of the 'backgroundColor' Phaser specific method. Next, the two sprites which were to be used throughout the game were loaded. These sprites being, the player sprite and the pipe sprite. The pipe sprite is essential in the creation of the level and acts as the main obstacle for the player through the game. Once the background colour had been set and the required sprites were loaded, the creation of the game itself could begin. The code used in the preloading method is shown in Figure 19.

```
function preload() []

    game.stage.backgroundColor = '#71c5cf';

    game.load.image('player', 'assets/player.png');

    game.load.image('pipe', 'assets/pipe.png');
};
```

Figure 19: Preload method used to load sprites and set background colour.

The next steps in this process were, the initialisation of the physics system, adding the player sprite, enabling physics on the newly added sprite, and applying gravity to the sprite. The code required to do this all had to be written in the Phaser reserved 'create' method. Unlike the previous prototypes that were developed, this game requires a physics system to be applied to the entire game. This is

because unlike the previous prototypes, physics based calculations affects every rendered element in this game. In order to allow for collisions between sprites and for detections to be made when these collisions happen, the entire instance of the game must have a physics system running. It should be noted that the primary difference between how sprites interact with each other in this prototype in comparison to Memory Pair and Memory Drag, is that they collide instead of simply overlapping.

In order to allow for such collisions and collision detections to be made, the 'Arcade' physics system was applied to the entire instance of the game. This was done through specifying the desired physics system in the Phaser 'startSystem' method. The player sprite loaded in the 'preload' method was then rendered to the screen with standard Phaser add sprite method as demonstrated in the implementation of the previous prototypes (see Figure 20).

```
function create() {

    // Fuction called after 'preload' to setup the game
    game.physics.startSystem(Phaser.Physics.ARCADE);

    // Display the player sprite on the screen
    player = game.add.sprite(100, 245, 'player');

    game.physics.enable(player, Phaser.Physics.ARCADE)

    // Add gravity to the player sprite to make it fall
    player.body.gravity.y = 1000;
```

Figure 20: Code required to apply a physics system to the game instance, render preloaded sprites, enable physics on player sprite, and apply gravity to the sprite.

Although the physics system was now attached to the entire game instance, the physics system had to be reapplied onto the player sprite separately in order to allow for the manipulation of the physics which affect the sprite. In other words, instead of manipulating the physics of the entire game instance just to effects of physics on the sprite, the sprite could be manipulated separately without affecting the rest of the game. This meaning the sprite would comply with the general physics rules that the game instance is controlled by unless overridden by the developer. An example of overriding the game instances physics can be seen when applying gravity effects to the sprite. The code required for this can be seen in Figure 20. Gravity on the y axis was applied to the player sprite so that it would have the effect of falling.

Next, multiple copies of the pipe sprite which was previously loaded were to be added to the game. These are required as they act as the main obstacle for the player in the game and are essential in the

games level creation. Instead of using the create sprite method multiple times, the 'createMultiple' Phaser method was used. This method allows for the creation of the same sprite multiple times, in order to do so however, a sprite group must first be made in which the sprites will be stored. After creating the sprite group with the Phaser 'group' feature, the 'createMultiple' method was used to add twenty copies of the previously loaded 'pipe' sprite to the group.

Unlike the standard sprite creation method, no coordinates for the newly created sprites are specified. The sprites are still created, however, they are not visible to the user as they have not been placed anywhere. The previously created group acts as the storage for the sprites and these sprites can then be accessed and rendered onto the display when required. Followed by this, the bodies of all the pipe sprites held in the pipes group were enabled as to allow for collisions and collision detections with the player sprite to be made. Instead of enabling the body of each sprite separately, the 'enableBody' method was simply applied to the group of sprites, thus enabling the body for each sprite within the group. The 'Arcade' physics system was then applied to each sprite in the group in a similar fashion. Through completion of this stage, the researcher had created twenty copies of the pipe sprite which were stored in the pipes group. These sprites had been given the capabilities for collisions, collision detection, and physics based manipulations separate from the physics affecting the rest of the game. The code required to implement all of this can be seen in Figure 21.

```
pipes = game.add.group();
pipes.createMultiple(20, 'pipe');
pipes.enableBody = true;

game.physics.enable(pipes, Phaser.Physics.ARCADE)
```

Figure 21: Code required to create a group, add multiple sprites, and enable body/physics on group.

Creating multiple sprites and storing them in a group results in the sprites being in a 'dead' state. This meaning, they are not rendered onscreen and thus not using the same amount of space in memory that a sprite in the alive state would. Sprites which are not displayed onscreen are usually stored in this state as memory is a precious resource on mobile devices. Performance is critical on low powered mobile devices in order to provide a streamline user experience.

In order to achieve the creation of the rising platforms which will act as the obstacles that a user must navigate through, two methods were created. The first function is used to render and position a single pipe, the second function is used to render multiple pipes which come together to create the platform

which the user will collide with. In order to create a single pipe a method was created named 'addOnePipe', which takes x and y axis coordinates as its parameters. Within this function firstly a local variable named pipe is used to store the first dead pipe from the previously created group of sprites. This is done through the use of the 'getFirstDead' Phaser method. Once the local variable pipe is set to store the first dead pipe in the group, it is checked to see if the pipe returns null. The pipe would return null in the case that there are no more dead pipes in the group to be assigned to the local variable. If there are no more dead pipes to assign to the local variable in the group, a new pipe sprite is created with the standard sprite creation method in Phaser. This method takes the x and y coordinates previously passed to the 'addOnePipe' method as its parameters. Once this sprite is created it is added to the previously created pipes group and automatically goes into a 'dead' state.

In the case that the local variable 'pipe' does not return null it is given the aforementioned x and y axis coordinates passed in as parameters from the parent method, with these parameters it positions the sprite stored in the local variable at the specified coordinates. This is achieved through the use of the 'reset' Phaser method which is used to reposition sprites. Next velocity on the y axis is added to the newly positioned sprite, this results in the pipe sprite rising from its initial position until it leaves the bounds of the canvas and is no longer visible to the user.

At this point a pipe sprite is rendered onto the screen and constantly moves upwards on the y axis. Once moving out of the bounds of the canvas the rendered pipe is no longer visible, however, once it has left the canvas area visible to the user it is still in memory. If each sprite was to continuously move out of the visible canvas area but still remain in memory, the mobile device would eventually run out of memory. As performance is critical in order to deliver a streamline user experience and memory is a scarce resource in mobile devices, it is essential that these rogue sprites be dealt with once they have carried out their purpose. In order to deal with this problem, the sprites have to be put back into the 'dead' state after they are no longer visible to users.

This was achieved with the use of two Phaser methods, 'outOfBoundsKill' and 'checkWorldBounds'. When 'outOfBoundsKill' is set to true and a sprite is marked as out of the world's bounds (not visible to a user) by the 'checkWorldBounds' method, the sprite is killed and no longer uses space in memory. The code used to carry these actions out can be seen in Figure 22.

```
pipe.outOfBoundsKill = true;
pipe.checkWorldBounds = true;
pipe.body.immovable = true;
```

Figure 22: Code required to remove the 'pipe' sprite when it leaves the world's bounds, and making the sprite immovable.

As shown in Figure 22, additional code to make the 'pipe' sprite immovable was also added. This was required as without it whenever the player sprite collided with the pipe sprite it would be knocked out of place. This meaning, the pipe sprite may have been placed correctly but it did not act as an obstacle for the player sprite as it would be pushed out of the path of the player sprite on collision. In order to resolve this, the 'pipe' sprite was set to immovable. After doing this whenever the player sprite collided with the 'pipe' sprite, the 'pipe' sprite would remain in its position.

The 'addOnePipe' method could now be used to create a row of pipes which would act as platforms for the player sprite to navigate through. In order to achieve this a new method named 'addRowOfPipes' was created. The code used in this method is shown in Figure 23.

```
function addRowOfPipes() {

    var hole = Math.floor(Math.random()*5)+1;

    for (var i = 0; i < 8; i++)
        if (i != hole && i != hole +1)
            addOnePipe(i*60+10, 500);

    score ++;
};
```

Figure 23: Adding a row of pipes with the use of the 'addOnePipe' method.

As each pipe sprite is 50 pixels wide it is possible to fit 8 of them into the 400 pixel wide canvas. Creating eight pipe sprites at each of the possible eight positions creates a platform however, the platform is unpassable by the user as there is no gap in the platform from which they can pass through onto the lower platforms. In order to create this gap a random number between 0 and 6 is generated and assigned to the local variable 'hole'. This number is used to represent one of the possible 8 positions that a pipe sprite can be placed on. Pipes are then created at 6 of the 8 possible locations, the two exceptions being the positions represented by the randomly generated number and the randomly generated number plus one e.g. pipes were not generated at positions 4, and 5 in the case that the random number was 4.

A for loop which iterated 8 times was used to implement this. The 'addOnePipe' is used in every iteration of the loop except in the iteration number which matches the randomly generated number and the randomly generated number plus one. This results in a platform made of pipe sprites being generated every time the 'addRowOfPipes' method is called. Each platform has a gap of 100 pixels (two pipe sprites wide), this gap is always in a random place of the platform due to the use of a random

number earlier. The gap is large enough for the player to navigate through in order to move onto lower platforms.

Next the newly created platforms were to be added periodically to the game with the use of the 'addRowOfPipes' method. This was achieved through adding a loop in the create method which was coded to executed the 'addRowOfPipes' method every 750 milliseconds. The code used is shown in Figure 24.

```
// Timer and adds rows of pipes
timer = game.time.events.loop(750, addRowOfPipes, this);
```

Figure 24: Adding a Phaser timed loop which adds pipes.

The loop used, is part of the time and event features offered by Phaser, it is implemented in the create function and continuously loops for the specified time.

At this point the player sprites falls with the effects of gravity and the platforms (created with pipe sprites) continuously rise. These rising platforms force the player to move between the gaps in the platforms to the lower platforms as failing to do so would result in the game ending. This is because the player sprite would be crushed between the platform and the games world's upper boundary. This concept has been illustrated in Figure 7. The next steps in this process was to enable collision with the world's boundaries and to enable user input. Collision with the world's boundaries and the player sprite was enabled by settings the 'collideWorldBounds' player setting to true. User input was to be achieved with the use of the devices orientation. In order to do this, the Gyro.js library was used. This library can be used to automatically track device orientation sensor readings. Firstly the tracking was initialised with the use of the Gyro.js 'startTracking' method. This method assigns the current values of the accelerometer and gyroscope to global variables, these variables can then be used to read the values of the sensors. The code used to achieve this has been demonstrated in Figure 25.

```
var accx = 0, accy = 0, accz = 0;
var gyroa = 0, gyrob = 0, gyrog = 0;
var score;
var player;

gyro.startTracking(function(o) {
    gyro.getFeatures();
    accx = Math.round(o.x);
    accy = Math.round(o.y);
    accz = Math.round(o.z);
    gyroa = o.alpha.toFixed(2);
    gyrob = o.beta.toFixed(2);
    gyrog = o.gamma.toFixed(2);
});
```

Figure 25: Gyro.js startTracking method used to assign sensor data to global variables.

When a mobile device is tilted to the right the acceleration on the x axis moves from 0 to a positive number, in the case that the device is tilted to the left the acceleration on the x axis moves from 0 to a negative number. With this information the controls can be implemented in the Phaser update method. Within the update method an if statement is used to check what the current acceleration on the x axis is, based on this information the player sprite is given the respective velocity to move left or right. The code used is shown in Figure 26.

```
function update() {

    if (accx > (0.9))
    {
        player.body.velocity.x = -350
    }
    else if (accx < (-0.9))

    {
        player.body.velocity.x = 350
    }
```

Figure 26: Mapping the devices orientation to player sprite movement.

It should be noted that the values -0.9 and 0.9 are used to remove any noise, anything between these numbers is considered too small of a movement to be considered as intended by the user. This also deals with tremors that a stroke victim can often face which disables a stroke victim from holding the device steady.

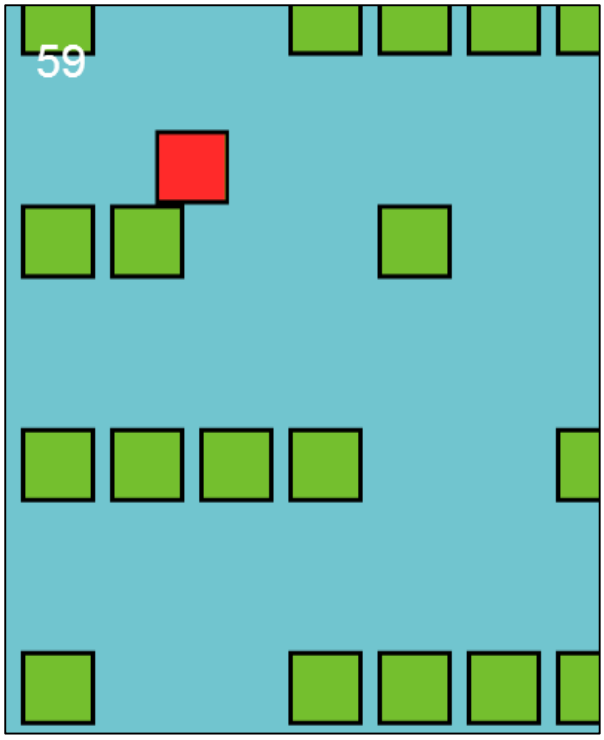

Figure 27: Complete Dropper prototype.

In order to finalise the creation of the game a score counter was added. This score counter is incremented with every new platform that is generated. The score counter is reset alongside the game itself in the case that the user collides with the top or bottom levels of the game world's boundaries. The finished prototype can be seen in Figure 27.

4.4 Roller

Roller is the second prototype created to stimulate a user's physical abilities, more specifically their fine motor skills. Roller's control scheme also makes use of the devices orientation however, instead of simply moving the device left and right on the x axis, this game requires a full range of motion on both the x and y axes. This prototype's level creation and game logic is very simple in comparison to Dropper as the level is static and does not require the generation of obstacles for the user.

As with previous prototypes the required boilerplate code was first initialised and the game instance was created. Next the required sprites were preloaded, these being the player sprite and the star sprite. The player sprite being the sprite controlled by the user and the star being the sprites which the user is to collect in order to increase their score. The player sprite was rendered onscreen and the games background colour was set using the same methods as in previous prototype implementations. Physics were also enabled on the entire game instance and on the player sprite. This was followed by the enabling of collision with the worlds bounds on the player sprite. This assured that the player sprite could only stay within the constraints of the canvas's borders. At this point there was a player sprite onscreen with no other sprites to interact with. Next the stars had to be rendered which would be collected by the user in order to score points. A group was created to hold the star sprites. This group was used to enable the Arcade physics system and the bodies of each sprite without typing out the same code multiple times. The body of the star sprites was enabled as to allow for collision detections to be made between them and the player sprite.

Next, multiple sprites were to be created, placed in random locations on the canvas, and added to the previously created group. The sprites were added to the group so that they could inherit the physics system and enabled body settings applied to the group. This was achieved through the use of a 'for' statement which iterated 25 times. This meaning, 25 star sprites were created in total. The code used to achieve the aforementioned implementations can be seen in Figure 28.

```
function create() {

    text = game.add.text(15, 15, String(score), style);
    game.physics.startSystem(Phaser.Physics.ARCADE);

    game.stage.backgroundColor = '#2d2d2d';

    //  This example will check Sprite vs. Group collision
    playerSprite = game.add.sprite(32, 200, 'player');

    game.physics.enable(playerSprite, Phaser.Physics.ARCADE);
    playerSprite.body.collideWorldBounds = true;

    starGroup = game.add.group();
    starGroup.enableBody = true;
    starGroup.physicsBodyType = Phaser.Physics.ARCADE;

    for (var i = 0; i < 25; i++) {
        var star = starGroup.create(game.rnd.integerInRange(100,
770), game.rnd.integerInRange(0, 570), 'stars',
game.rnd.integerInRange(0, 35));

    game.scale.fullScreenScaleMode = Phaser.ScaleManager.SHOW_ALL;
    game.input.onDown.add(function(){game.scale.startFullScreen();},
this);
    }

}
```

Figure 28: Create method used in the Roller prototype.

In order to enable user input the same methods used in the Dropper prototype creation were employed. This meaning, the Gyro.js library was used in the same fashion as previously, this can be seen in Figure 25. Within the update method gravity was applied to the player sprite based on the acceleration on the x and y axes (orientation). At this point the user now had control of the player sprite through the devices orientation however, on collision with star sprites nothing would happen as there was no method handling the collisions.

In order to deal with collisions within the update method the ‘collide’ Phaser method was used and the two sprites to collide were specified. The first being the player sprite and the second being the group holding the star sprites. The third parameter passed into this method was the name of function to be called in the event of a collision. The function ‘collisionHandler’ was created to handle collision events. Within this function the star sprites are killed, the score is incremented, and the score display is refreshed. As this point the user has control of the player sprite through the devices orientation and on collision with a star sprite the star sprite disappears and the score is incremented. The code used to control the player sprite is shown in Figure 29.

```
if (accx > (0.9)){
    playerSprite.body.gravity.x = accx * -200;
}
//right
else if (accx < (-0.9)){
    playerSprite.body.gravity.x = accx * -200;
}
//up
if (accy < (-0.9)){
    playerSprite.body.gravity.y = accy * 200;
}
else if (accy > 0.9){
    playerSprite.body.gravity.y = accy * 200;
}
```

Figure 29: Affecting the player sprites gravity based on the orientation readings.

4.5 Tilt Maze

Tilt Maze was the final game to be created as part of the suite of games. Unfortunately, the time constraints of the project restricted the researcher from carrying out this implementation. This was because the complexity involved with the pseudo-random maze generation was underestimated. Such a complex task could not have been achieved alongside the creation of four other prototypes. For this reason Tilt Maze's implementation was dropped from this project. This however, can be taken on in advancements of this research in the future.

4.6 Prototype Public Access

The four prototypes are publically available for testing at www.stickerdude.co.uk/game. Each prototype is accompanied by documentation regarding the games instructions and all the assets and sources used can also be viewed here.

5 Chapter 5: Evaluation

This chapter shall discuss the results gained from the five participants in the testing of the four prototypes. All four prototypes will be tested by each participant. The information gained as part of these results will then be discussed. Next, an evaluation from the researcher's perspective regarding the minimum requirements of the project is to be carried out. This evaluation shall be used to determine whether or not the project met the minimum requirements for it to be deemed successful. This will be followed by a professional evaluation which will allow the researcher to gather feedback concerning the prototypes from a professional perspective. Finally, the prototypes created in the implementation stage will be assessed to determine whether or not they met the previously outlined design requirements.

5.1 User Evaluation

A user evaluation was carried out by giving a number of participants access to the prototypes so that they may offer feedback on their designs and implementations. Furthermore, a questionnaire was also issued to each participant so that specific information of interest to the researcher could be gathered. Followed by the completion of the questionnaire the participants were given an opportunity to vocally provide feedback regarding the reasoning behind their answers in the questionnaires.

Before the questionnaires were handed out to the participants the participants were each asked four setup questions. These setup questions are as follows:

1. What age are you?
2. Which device are you using?
3. Which operating system are you using?
4. Which browser are you running?

The results of these setup questions can be seen in Table 1.

The next stage was the user questionnaire. This questionnaire was filled out a total of four times by each of the five participants. A questionnaire was filled out for each prototype tested by the participants. Each participant tested all four prototypes which were previously implemented. The questions asked in this questionnaire are as follows:

1. On a scale of 1 to 10, how was your experience with this prototype?
2. Which abilities do you think were the main focus in this game, cognitive abilities, physical abilities or neither?
3. Out of the abilities used for this game which percentage were cognitive skills?
4. Out of the abilities used for this game which percentage were physical skills?
5. On a scale of 1 to 10, how good was the games performance on your device?

The average results for the questionnaire can be seen in its entirety in Appendix 1.

5.1.1 Results

The results gathered from the setup questions are shown in Table 1.

	Participant A	**Participant B**	**Participant C**	**Participant D**	**Participant E**
Age	21	23	24	26	52
Device	Google Nexus 7 (2013)	Google Nexus 5 (Mobile)	Samsung Galaxy Note 10.1	Toshiba Encore	Apple iPad 2
Operating System	Android 4.4.4	Android 4.4.4	Android 4.1	Windows 8.1	iOS 7.1.2
Browser	Google Chrome	Google Chrome	Google Chrome	Mozilla Firefox	Apple Safari

Table 1: User evaluation setup question results.

As shown in Table 1, the participants tested all of the prototypes on various different devices, browsers, and operating systems. This was essential in order to provide empirical evidence of platform and browser independence. It should be noted that the devices used by the participants were the participants own devices, this meaning, they were not supplied by the researcher.

	Memory Drag	**Memory Pair**	**Dropper**	**Roller**
On a scale of 1 to 10, how was your experience with this prototype?	10	10	8	10

Table 2: Questionnaire Average Results Table, Question 1.

As shown in Table 2, the general experience of using the prototype was excellent across all participants. Experience in the context of this questionnaire meaning whether or not the prototype functioned as intended. On a scale of 1 to 10 the experience of each game was a perfect 10 except in the case on Dropper. This can attributed to the fact that the participant using the Toshiba Encore Windows 8.1 tablet faced problems when running the application and thus decreased the total average by providing a lower mark out of 10 than the other participants. The problem faced by this participant was control based and resulted in them not being able to play the Dropper prototype as anticipated. Tilting the Toshiba Encore tablet left and right did not move the player sprite as anticipated, instead, the device had to be moved upwards and backwards in order to move the player sprite left and right. This problem could have been attributed to any of the following three factors, the device, operating system, and browser. In the case of this participant (participant D) neither the operating system nor browser used fell under the supported platforms and browsers in this project. Through the knowledge gained in the research stage is was clear that the problem component was the browser being used. As mentioned by MDN (2014)[52], Google Chrome and Mozilla Firefox handle angle based calculations differently, and for this reason, readings on some axes can be reversed. In order to ensure that the problem component was indeed the browser, the participant was asked to use the Google Chrome browser instead. Through doing so the prototype ran as expected.

In summary, each prototype met the constraints of platform and browser independence by running correctly on a range of platforms and browsers. In terms of platform independence the prototypes

have surpassed their minimum requirements by operating correctly on a platform other than the ones support was specifically developed for (Windows 8.1).The issue faced with reversed controls on the Mozilla Firefox browser could potentially be solved by adding some form of check before the game loads. This would check which browser is being used and based on the browser an appropriate control scheme should be set.

	Memory Drag	**Memory Pair**	**Dropper**	**Roller**
Which abilities do you think were the main focus in this game, cognitive abilities, physical abilities or neither?	Cognitive	Cognitive	Physical	Physical

Table 3: Questionnaire Average Results, Question 2.

The second question asked was used to gauge whether or not the users themselves were clear of which abilities were the main focus of the prototypes. This was useful information as it is imperative that the users themselves are aware of the abilities they are using so that they when progressing in the game they are aware of which abilities they are rehabilitating. As shown in Table 3, no participant felt that neither cognitive nor physical abilities were being stimulated. Every participant chose the abilities which matched the abilities the prototypes were supposed to target. This acts as evidence of the fact that with all prototypes users were aware of which abilities were targeted by the game.

	Memory Drag	Memory Pair	Dropper	Roller
Out of the abilities used for this game which percentage were cognitive skills?	70	60	50	30
Out of the abilities used for this game which percentage were physical skills?	30	40	50	70

Table 4: Questionnaire Average Results, Questions 3-4.

This average results of the third question asked in the questionnaire can be seen in Table 4. The third question asked was used to determine at what level a user's cognitive and physical abilities were being used. With this information the success of each prototype in the stimulation of these abilities could be gauged. The two games designed for cognitive stimulation received the expected results. This meaning a majority percentage of the abilities being used were classed as cognitive skills according to the participants. This was expected as Memory Pair and Memory Drag were both designed specifically for cognitive stimulation. A surprising result was found when asking the participants which percentage of cognitive skills were used in the games designed for physical stimulation. The Dropper prototype received an average of 50 percent for cognitive stimulation and 50 percent for physical stimulation.

This was unexpected as the game was designed specifically for physical stimulation, however, the participants felt that it invoked an equal amount of cognitive and physical stimulation. When questioned about the reasoning behind this, the general consensus amongst the participants was that the cognitive fast planning and the organisation skills required to be successful at the game were just as demanding as the physical fine motor skills required. Although some usage of cognitive abilities was expected it was not expected to be at this scale.

These results have allowed for a better understanding of how to provoke a more balanced stimulation across both cognitive and physical abilities. It has been realised from these results that cognitive stimulation is required in the rehabilitation of fine motor skills. Fine motor movements are complex and require concentration and control, this control does not only fall under physical abilities but also cognitive abilities. Mental focus is required to carry out the precise physical fine motor skill

movements. Roller was classed as using less cognitive stimulation than Dropper however, the levels of cognitive stimulation were still much higher than anticipated.

	Memory Drag	**Memory Pair**	**Dropper**	**Roller**
On a scale of 1 to 10, how good was the games performance on your device?	10	10	7	10

Table 5: Questionnaire Average Results, Question 5.

The final question asked as part of the questionnaire was used to determine how each prototype performed on the various devices. As shown in Table 5, three out of the four prototypes offered excellent performance with no interruptions. Dropper however as previously mentioned had disruptions with the control scheme. This once again has brought down the average lower than it would have been without this participant. From a technical perspective each prototype ran flawlessly in terms of frames per second and controlled system usage. However, in terms of control scheme performance participant D faced problems. These were resolved through changing the browser being used, however, the user had already faced the disruption at this point.

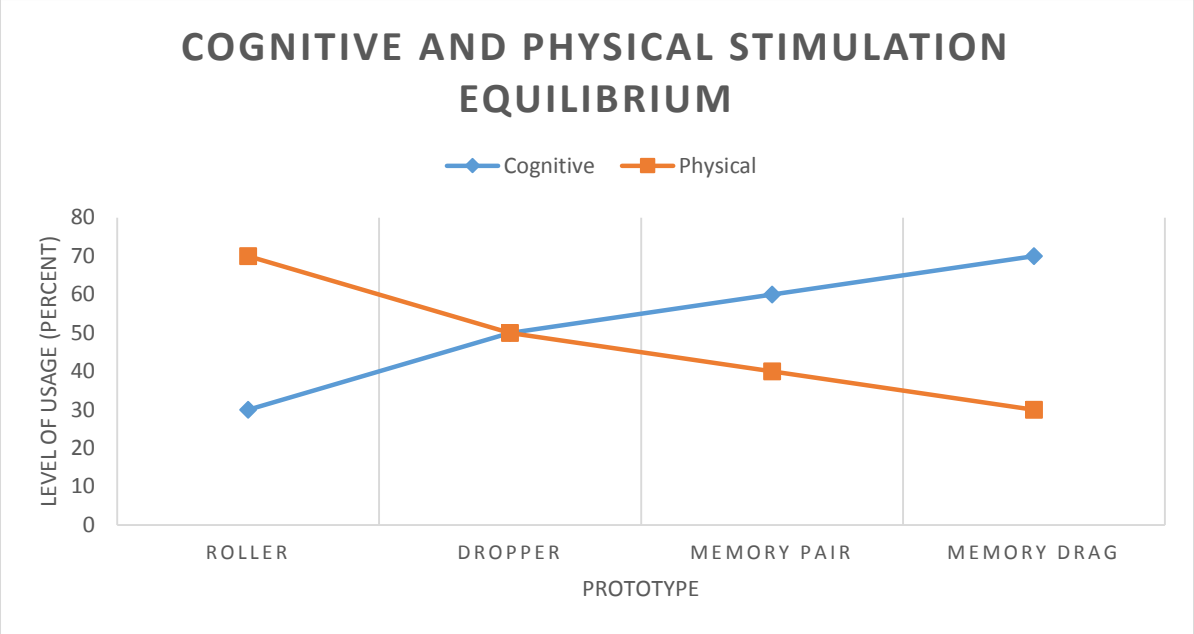

Figure 30: Graph depicting the results shown in Table 4.

As shown in Figure 30, out of the two physical stimulation based prototypes, Dropper invoked the most cognitive stimulation. This information is useful in the creation of future games where a balance between physical and cognitive stimulation is to be achieved. Limiting the users in controlling the player sprite by only giving them the ability to tilt the device left or right resulted in lower physical stimulation than Roller, where a full range of motion was used. This was expected however, the high level of cognitive stimulation achieved were not. This can be attributed to the quick movements and elements of planning involved in the Dropper game. These elements were required to be successful in the game and should be carried over to other games where an equal amount of stimulation of physical and cognitive abilities are to be achieved.

5.2 Researcher's evaluation

This section is dedicated to the evaluation of the prototypes from the researcher's perspective. Firstly whether or not the minimum requirements needed to be satisfied for this project to be classified as successful have been met, will be discussed. Followed by this, whether or not the prototypes fit the design requirements set shall be highlighted. How they have achieved these requirements or why they have failed to meet certain requirements shall be discussed.

5.2.1 Minimum Requirements

This stage of the evaluation shall discuss how each of the minimum requirements for this project to be deemed successful were met. These minimum requirements being:

The software solution must be:

1. Platform independent (Supporting at least both Android 4.1+ and iOS platforms).
2. Browser independent (Supporting at least Chrome and Mobile Safari).
3. Targeted at the rehabilitation of stroke victims.
    a. Stimulate damaged physical abilities
    b. Stimulate damaged cognitive abilities

As discussed in section 1.2.1, platform independence was implicitly achieved through the creation of web applications instead of native applications. The very nature of web applications allows for them to be deployed on any platform on the basis that the platforms browser supports all the features required of the web application (video game). Furthermore, the prototypes created were tested on multiple platforms (devices) with success. These devices are shown in Table 1. Three Android OS based devices from two different manufacturers were tested and provided excellent performance and functionality with no problems. Furthermore, a non-Android OS based device was also tested with success, certain problems were faced however these were not related to the platform itself.

The final device tested was an Apple iOS based device which performed equally as well as the Android OS based devices. This not only satisfies the minimum requirement of platform independence but surpassed the requirement by the ability to be deployed on non-Android and iOS devices. Web

applications are almost always platform-agnostic and are restricted by the browsers they are deployed on rather than the platforms those browsers are deployed on.

The second requirement; browser independence, was achieved through the use of development tools and features which were compatible with the target browsers (Google Chrome and Mobile Safari). The main development tool being, the Phaser game framework and the main features being, the features available through this framework, JavaScript, and HTML5. The compatibility of the features was confirmed through the use of official documentation and compatibility checkers. Generally amongst all major browsers compatibility is equal except in the case of features which are rarely used by the masses. As not all browsers support every HTML5, JavaScript, or Phaser feature it is inevitable that browser independence cannot be achieved on a scale which envelops support for all browsers on the market. However, through the setting of certain restrictions, controlled browser independence was achieved. This meaning, browser independence was achieved between the Google Chrome and Mobile Safari browsers. The user studies carried out provide empirical evidence of this.

The final requirement of the prototypes in order for them to be deemed successful was that they should be able to stimulate both cognitive and physical abilities for the rehabilitation of stroke victims. This was achieved through emulating traditional methods of stroke rehabilitation through the prototypes game mechanics. In the case of fine motor skills, the movements stroke victims are taught to make during certain rehabilitation exercises (Stroke-rehab, 2014)[53] were made by the players of the Dropper and Roller prototypes. This allowed for physical abilities to be stimulated and thus improved upon them. Similar methods were used to achieve cognitive stimulation. To summarise, both physical and cognitive stimulation were achieved for the purpose of stroke rehabilitation. This is evident from the adoption of traditional rehabilitation methods into the prototypes and the results concluded through the user evaluation.

5.2.2 Design Requirements

This stage of the evaluation shall consist of a comparison of the final implementations and their designs previously outlined in the design section. This shall allow the researcher to gauge how successful they were in the implementation of the prototypes.

5.2.2.1 Memory Drag

The implementation of Memory Drag was successful and managed to meet all the design requirements. In this implementation the researcher was also able to add additional features which were not outlined in the design of the game. The basic concept of placing eight sprites on the canvas, four in the left column and four in the right column, was achieved easily. This was well within the limitations of the web languages used and the framework employed. The ability to interact with these sprites through a touch interface on tablet or mobile device was also easily achieved as both the JavaScript programming language and the Phaser game framework supported the features required. The code written for this prototype allows for text sprites, image sprites, or both to be used with little change required. As outlined in the design, this interchangeability between image and text sprites will allow future iterations of the prototype to switch between targeting visual memory and verbal memory, or both simultaneously. All the required features to implement the correct design for this game were supported by the web languages and framework used. Furthermore, a scoring system was also added to the final prototype which was not outlined in the design stage. Through achieving this the Memory Drag prototype surpassed its original design. To summarise, through completing the outlined design objectives it was possible to trigger cognitive stimulation successfully with the Memory Drag prototype.

5.2.2.2 Memory Pair

The implementation of Memory Pair was not successful and failed to meet all of the design requirements, this can be attributed to limitations with the HTML5 canvas and the Phaser game framework. The underlying mechanics were similar to Memory Drag however, Memory Pair greatly differed when the control scheme and how points are scored are taken into account. Once again the placement of sprites as with Memory Drag, was well within the limitations of the Phaser game

framework. The selecting of sprites through touching them was easily achieved with the use of inbuilt features which supported touch interface with tablet and mobile devices.

Where the prototype failed to meet its design requirements was in the generation of a curvy line between the two selected sprites, and the detection of a player's finger on this line. The Phaser game framework does not currently at the time of the undertaking of this project, support the generation of geometrical objects which can be interacted with. In the final prototype for Memory Pair a line was generated between the two objects successfully however this line was not curvy as intended, and users could not interact with it. The generation of this line was achieved through the use of the geometric debug tools of Phaser and was added in place of the curvy line so that the game would at least be demonstrable as a proof of concept.

As the line was a Phaser geometry object it could not be interacted with as you could with a sprite. The first proposed solution to this problem was the creation of a curvy line sprite that would be generated instead of an actual line object. This however presented more problems than it solved as firstly the element of random generation can no longer exist as you would have to choose a sprite from a set of sprites instead of creating a new curvy line sprite every time. Furthermore, although it was first anticipated that detections regarding whether or not a user's finger was on the line could be made, this was not the case. It was possible to determine where the user's finger was on the canvas and then match these points against every possible position the curvy line could take. This however is an extremely convoluted method of achieving this feature especially when the fact that every point the line occupies on the canvas must also be known.

The second proposed solution was the use of the HTML5 'arcTo' method. This method is used to generate lines with curves onto the HTML5 canvas. The required algorithm to generate these curved lines could potentially be used alongside this method in order to achieve the desired results. However, this still does not mean that the generated line can be interacted with by the user, especially not in the case where this generated line would only overlap with whatever is generated by the Phaser framework and is not actual part of Phaser elements rendered onto the canvas.

Finalising this prototype by implementing one of two proposed solutions would have led to disruptions in the project timeline and thus it was left with the generation of a simple Phaser geometry line in order to provide evidence of the concept. With some effort the original design is still quite possible to achieve however, not as first anticipated. To summarise, the prototype created was still able to stimulate cognitive abilities even though it did not complete the design requirements.

5.2.2.3 Dropper

The Dropper implementation met all of the outlined design requirement as projected. There were three main elements required from the Phaser framework to meet the design requirements. These being, sprite rendering, sprite physics, and collision detection. Once the two main sprites being used were preloaded and physics systems were attached, the rest of the games development was carried out with the use of the JavaScript programming language. The functions required to create platforms out of the Phaser sprite objects were all able to be created because of the functionality offered by JavaScript. To summarise, the Dropper prototype was able to stimulate physical fine motor skills successfully and managed to fully adhere to designs set.

5.2.2.4 Roller

The Roller prototype was implemented successfully and met the outlined design requirements. It was able to successfully stimulate physical fine motor skills with a full range of motion. Furthermore, just as the rest of the prototypes it allowed for platform independence through the fact that it was a web application. Browser independence was achieved through the usage of framework and language features which were supported by the target browsers. Unsupported features were not used in development of any of the prototypes. To summarise, Roller satisfied the outlined design requirements and allowed for the stimulation of damaged physical abilities.

5.3 Professional Feedback

As part of the evaluation, professional feedback concerning the prototypes and the project as a whole was sought. This was made up of an informal discussion before and after the healthcare professional had tested the prototypes. The professional in question was, is a Clinical Neurophysiologist in the NHS at the Royal Hallamshire Hospital based in Sheffield, UK. As articulated by Nhscareers.nhs.uk (2014)[54], Neurophysiologists are specialist practitioners who deeply investigate the nervous system in order to diagnose and monitor neurological disorders such as, strokes, muscle dysfunction, and other neurological disorders. This makes the professional a strong candidate for offering feedback on this project.

Firstly the professional was asked how they felt about tablet PCs and their current role within the healthcare system. The professional felt that an increase in adoption of tablet PCs within the healthcare system was essential as they themselves felt it was an untapped market with many potential uses. They went on to say, tablet PCs have slowly made an appearance amongst clinicians however, the technology currently used to aid clinicians in their everyday work is primitive and has much room for improvement. The professional then went on to discuss the fact that due to monetary constraints the NHS rarely makes improvements on the tablet PC technology which is currently in use. This meaning, generally, the tablet PCs currently used by clinicians at the Royal Hallamshire Hospital in Sheffield, UK, are antiquated and can be improved upon immensely. Before testing the four prototypes created, the professional went on to make a final remark. This being, tablet PCs already seamlessly integrate into a clinicians everyday activities, research towards the integration of these devices into a patient's everyday rehabilitation activities does show potential, purely based of how well they have integrated into a clinicians everyday activities.

At this point the professional was asked to test each of the four prototypes. The professional made use of a HTC One M8 device running Android version 4.4.4, in the testing of the prototypes. The browser used by the professional was Google Chrome. It should be noted that the device used was the clinicians own device and was not supplied by the researcher. The first two prototypes tested were the prototypes aimed at the stimulation of cognitive abilities, these being, Memory Pair and Memory Drag. The clinician felt that their judgement alone would not be enough to deem these prototypes successful in the rehabilitation of stroke victims. This is because, without clinical trials and extensive testing their judgement could not be verified. They could only offer a professional opinion of whether or not through their expertise they felt the prototypes could be used in the rehabilitation of stroke victims.

The neurophysiologist felt that the specific targeting of visual and verbal memory was not the best approach in the rehabilitation of stroke induced aphasia. This was because, as stated by the clinician, although there are loose definitions of the types of aphasia, there is far too much variation in symptoms amongst those affected by aphasia in order to target specific ones. This point has been affirmed by Kolb and Whishaw (2003)[55]. As stated by the professional, in order to target stroke induced aphasia, more general symptoms must be targeted. In the case of aphasia, the most common damage caused is the disruption of the comprehension and expression of language. A suggestion was then made to design a prototype which incorporate elements of speech recognition and typing. The reasoning behind this being that targeting these two elements would be the most general method of targeting aphasia. Through a general approach multiple types of aphasia can be rehabilitated instead of individual symptoms.

After testing the physical ability based prototypes the clinician went on to discuss doubts they had regarding whether or not improvements made by stimulating these abilities would translate directly into improvements of everyday activities. The player's fine motor skills may have been improved upon however, it did not seem likely that through these specific games the stimulation achieved would directly translate over into useful functions. It should be noted however, that the neurophysiologist did still feel the capability to rehabilitate stroke victims existed, but the effectiveness of this rehabilitation was heavily dependent on the design of the games.

5.4 Summary

User Trials

The most significant knowledge gained as part of the user trials is the levels of stimulation offered by each game. It has been realised that focusing on individual abilities without invoking others is harder than anticipated. Knowing this, future work on these prototypes could move towards a more general rehabilitation approach instead of isolating certain abilities. This is because it would be best to take advantage of the synergy offered by the prototypes. Every prototype created to stimulate cognitive abilities will always invoke some form of physical stimulation purely because of the control scheme, every input method requires some form of physical exercise. For this reason it would be optimal to develop games in such a manner that they accept this extra stimulation and use it to their advantage instead of marking it off as unintended. During the development of the physical ability based games the scale of the level of usage of cognitive abilities was completely unknown. Due to these results it is now known how mentally straining it can be to make use of certain physical abilities. Once again instead of developing against this, efforts should be made to accept this.

Researchers Evaluation

Through carrying out the researcher evaluation, the success of this project in meeting the minimum requirements was outlined. Furthermore, whether or not the prototypes created adhered to the designs previously outlined was also discussed. In regards to meeting the minimum requirements, this project can be deemed successful. This is because, a tablet PC based software solution which is platform independent, browser independent, and targeted at the rehabilitation of stroke victims, has successfully been created. Whether or not this rehabilitation is useful within its intended environment is unknown without professional advice and clinical trials. This however, does not negate the research done in this project as the stimulation of physical and cognitive abilities has been achieved. From this stimulation it can be said from a theoretical standpoint, that stroke rehabilitation has been achieved in this project. Additionally, from a technological standpoint, it has been empirically proven with the languages and tools used that Tablet PC based video games which are platform independent, browser independent, and targeted at the rehabilitation of stroke victims can be achieved.

Professional Evaluation

Through carrying out the professional evaluation, the importance of the design of the video games was realised. As discussed by the clinician, the tentative potential of tablet PC based video games in the rehabilitation of stroke victims exists and can be improved upon. After testing the prototypes the professional was certain that through the right design choices, this potential could be unlocked and actual stroke patient rehabilitation would be in sight. In summary, the prototypes themselves may not be designed perfectly for the rehabilitation of stroke victims, however, the feedback offered by the professional has confirmed that through the right design choices, stroke rehabilitation is possible through tablet PC based video games.

6 Chapter 6: Conclusion

This study investigated to what extent HTML5 and JavaScript could be used in the implementation of tablet/mobile based games which are platform independent, browser independent, and are engineered towards the rehabilitation of stroke victims. This was achieved through outlining and accomplishing aims, objectives, and minimum requirements. These were met through following the project methodology. This methodology was made of multiple stages such as the, research, design, implementation, and implementation stages. Through the research stage the void in research which was to be filled was outlined. This being, the gap between the usage of tablet PCs and video games in the healthcare system. The rationale behind this was the potential benefits offered by both tablet PCs and bespoke software solutions. Benefits attached to tablet PCs included factors such as, accessibility, low manufacturing costs, and seamless integration into a patient's rehabilitation program. The creation of a bespoke video game allowed for games to be tailored to the needs of stroke patients. Furthermore, bespoke video games can potentially be tailored to collect and distribute important patient information to carers.

Next, the capabilities of HTML5, JavaScript, and the Phaser game framework were highlighted. The theoretical implications from this were that these web languages and tools were adequate in the creation of a software solution which is targeted at the rehabilitation of stroke victims. Based on the information gained during the research stage, the designs of the prototypes could be planned. Five prototypes were designed in total however, only four were implemented. This was because of time constraints and lack of appreciation of how comprehensive the undertaking of such a prototype would be. One of the designed prototypes could not be fully implemented due to limitations with the game framework being used. How these limitations could be overcome were later outlined in the researcher's evaluation.

After the design and implementation of the prototypes, the evaluation could begin. This evaluation was made of user trials, the researcher's evaluation, and professional feedback. From the user trials it was empirically proven that the prototypes developed were indeed platform and browser independent, as they had been tested on a range of platforms and browsers. The platform independence requirements was surpassed as in the user trials, a participant successfully tested the prototypes on a device which was not explicitly supported i.e. a non-Apple or Android device. Through carrying out the user trials it was concluded that the games are playable, robust and seemed to people who are not experts in the field or stroke patients to be plausible for stroke patient rehabilitation. This final point however cannot be verified without extensive professional feedback and clinical tests.

From the researcher's evaluation, it was firmly concluded that the prototypes created had met the outlined aims, objectives, and design requirements. From this it can be concluded, that tablet-based games of the type envisaged are technologically possible using HTML5, JavaScript, and Phaser. Although this was theorised in the research stage, it was yet to be empirically proven.

From the professional feedback, it can be tentatively concluded that the potential for stroke rehabilitation through tablet PC based video games exists, however, the design of the games is critical in doing this successfully. The designs used in this project were intended to test the limits of the web technologies and give us, as researchers, a working prototype which could be shown to people in order to gauge their reactions. The next step in this research should be, showing the designs to healthcare professionals and stroke patients in order to gauge their reaction. This is a common technique for requirements elicitation where something is created in an attempt to get the target end-users talking about what it is they actually require. This meaning, through creating these designs, healthcare professionals and stroke patients now have something to critique. From this critique, requirements can be gathered and a more effective design can be implemented.

6.1 Future Work

The prototypes created as part of this project act as evidence of the ability to create platform and browser independent video games which can be used in the rehabilitation of stroke victims. These prototypes are still in a primitive stage and have only been developed to the point of proof of concept rather than to the point of actual mainstream usage. This is one of the main points of future expansion, the prototypes created can be improved upon in every aspect.

6.2 User Interface

The user interfaces created for the prototypes was a placeholder user interface created only so the prototypes could be demonstrated. There is much room for advancement in regards to the user interfaces for the four games created. One of these improvements being better adherence to the design principles proposed in the design section of this document. The user interfaces could be further tweaked towards the target demographic with the aid of specialist advice. In the design and development of a user interface which is designed for mainstream usage, advice from stroke specialists and game designers should be sought for the best results.

6.3 Suite of Games

As discussed in section 3.1 which outlined the ideal application outcome, ideally a suite of games was to be developed. This suite of games would be presented to the user in the form of a single application and user scores across all games which are a part of this application should be stored centrally on this single application. The prototypes created were presented as four separate video games and there was no element of them coming together to form a single application. This is something which can be worked on in future work. Bringing these games together within an application which is able to manage user scores and make suggestions based on these scores as to what game a user should play is an extensive task. Additionally users should be able to navigate between different games within this single application. This could potentially be achieved through the usage of the Phaser frameworks state system. This system allows for the creation of menus and the changing of levels. With some development this system could potentially be tweaked to allow for the switching between games

instead of levels. This is potentially possible because each level is treated as a state just as each game is treated as its own state. Phaser provides the ability to switch between states and through using this it is potentially possible to allow for such features.

6.4 Gathering and Relaying Information Back to Carers

Another feature which could be worked on in the future in order to achieve the outlined ideal outcome, would be the implementation of a system which allows for user performance data to be relayed back to carers. This would allow for carers to gain an insight on the progress of their patients and would allow them to make decisions regarding a patient's rehabilitation based on their performances. This could be achieved through the introduction of PHP which is a server side scripting language. Functionalities such as the relaying of information to remote nodes can already be achieved through PHP however, the possibility of integrating these abilities with Phaser based video games is yet to be explored.

6.5 Ideal Application Outcome

There is much work yet to be done to achieve the final ideal application outcome. Accomplishing everything in this ideal outcome would have been extremely difficult to achieve within the time constraints of this project. For this reason it was expected for this projects work to continue past the project deadline. The researcher aims to continue his work on this project and one day finally accomplish the ideal application outcome.

7 Chapter 8: References

[1] Gartner.com, (2014). Gartner Says Annual Smartphone Sales Surpassed Sales of Feature Phones for the First Time in 2013. [online] Available at: http://www.gartner.com/newsroom/id/2665715 [Accessed 24 Jul. 2014].

[2] Apple, (2014). Apple IOS6 EULA. Apple [online] Available at: http://www.apple.com/legal/sla/docs/ios6.pdf [Accessed 25 Jul. 2014].

[3] Developer.android.com, (2014). Application Fundamentals | Android Developers. [online] Available at: http://developer.android.com/guide/components/fundamentals.html [Accessed 25 Jul. 2014].

[4] Developer.apple.com (2014). Start developing iOS Apps Today: Setup. [online] Available at: https://developer.apple.com/library/iOS/referencelibrary/GettingStarted/RoadMapiOS/index.html [Accessed 25 Jul. 2014].

[5] Apple.com (2014). Apple - Press Info - Apple Reinvents the Phone with iPhone. [online] Available at: http://www.apple.com/pr/library/2007/01/09Apple-Reinvents-the-Phone-with-iPhone.html [Accessed 26 Jul. 2014].

[6] Smith, M. (2014). Google Chrome browser arrives on Android (video). [online] Engadget. Available at: http://www.engadget.com/2012/02/07/google-chrome-browser-arrives-on-android-video/ [Accessed 26 Jul. 2014].

[7] Stroke.org, (2014). *Stroke Recovery - How do I Recover from a Stroke? - National Stroke Association.* [online] Available at: http://www.stroke.org/site/PageServer?pagename=recov [Accessed 27 Jul. 2014].

[8] Strokeassociation.org, (2014). Tips for Improving Fine Motor Skills. [online] Available at: http://www.strokeassociation.org/STROKEORG/LifeAfterStroke/RegainingIndependence/PhysicalChallenges/Tips-for-Improving-Fine-Motor-Skills_UCM_309776_Article.jsp [Accessed 28 Jul. 2014].

[9] Hammond (2002). Neuroplasticity. [online] Available at: http://web.stanford.edu/group/hopes/cgi-bin/wordpress/2010/06/neuroplasticity/ [Accessed 28 Jul. 2014].

[10] Entertainment Software Association. 2013. *ESSENTIAL FACTS ABOUT THE COMPUTER AND VIDEO GAME INDUSTRY*. [report] ESA, p. 2.

[11] Connolly, T. M., Boyle, E. A., Macarthur, E., Hainey, T. and Boyle, J. M. 2012. A systematic literature review of empirical evidence on computer games and serious games. *Computers \& Education*, 59 (2), pp. 661--686.

[12] Martin, J. 2012. Game On: The Challenges and Benefits of Video Games. *Bulletin of Science, Technology \& Society*, 32 (5), pp. 343--344.

[13] Green, C. S. and Bavelier, D. 2003. Action video game modifies visual selective attention. *Nature*, 423 (6939), pp. 534--537.

[14] Primack, B. A., Carroll, M. V., Mcnamara, M., Klem, M. L., King, B., Rich, M., Chan, C. W. and Nayak, S. 2012. Role of video games in improving health-related outcomes: a systematic review. *American journal of preventive medicine*, 42 (6), pp. 630--638.

[15] Read, J. L. and Shortell, S. M. 2011. Interactive games to promote behavior change in prevention and treatment. *Jama*, 305 (16), pp. 1704--1705.

[16] Sawyer, B. 2008. From cells to cell processors: the integration of health and video games. *Computer Graphics and Applications, IEEE*, 28 (6), pp. 83--85.

[17] Stroke.org. 2012. *Effects of Stroke - National Stroke Association*. [online] Available at: http://www.stroke.org/site/PageServer?pagename=effect [Accessed: 3 Apr 2014].

[18] Lee, G. 2013. Effects of Training Using Video Games on the Muscle Strength, Muscle Tone, and Activities of Daily Living of Chronic Stroke Patients. *Journal of Physical Therapy Science*, 25 (5), p. 595.

[19] De Wit-Zuurendonk, L. and Oei, S. 2011. Serious gaming in women's health care. *BJOG: An International Journal of Obstetrics \& Gynaecology*, 118 (s3), pp. 17--21.

[20] Pcmag.com, (2014). Tablet computer Definition from PC Magazine Encyclopaedia. [online] Available at: http://www.pcmag.com/encyclopedia/term/52520/tablet-computer [Accessed 6 Jul. 2014].

[21] Hager, H. and Burka, S. (n.d.). A historical overview of tablet computing, GUIs and hypertext. [online] Available at: http://www.uni-salzburg.at/fileadmin/multimedia/SRC/docs/teaching/SS11/SaI/seminararbeit_burku_hager.pdf

[22] Walker, G. (2011). Tablet Product and Market History.[online] Available at: http://www.walkermobile.com/Tablet_History.pdf

[23] HP (2014). [online] Available at: http://www.hp.com/sbso/solutions/healthcare/hp_tablet_whitepaper.pdf [Accessed 27 Aug. 2014].

[24] Motion Computing (2006). [online] Available at: http://www.motioncomputing.com/downloads/User_Docs/Accessories/WhitePaper_HealthCare.pdf [Accessed 28 Aug. 2014]

[25] Raggett, D., Le Hors, A. and Jacobs, I. (1998). HTML 4.1 Specification. [online] p.18. Available at: http://www.w3.org/TR/1998/REC-html40-19980424/html40.pdf [Accessed 9 Aug. 2014].

[26] Berjon, R., Faulkner, S., Leithead, T., Doyle Navara, E., O'Connor, E., Atkins, T., Pfeiffer, S., Weiss, Y., Pieters, S., Marquis, M., Hickson, I. and Cáceres, M. (2014). HTML 5.1 Nightly. [online] Available at: http://www.w3.org/html/wg/drafts/html/master/introduction.html#background [Accessed 9 Aug. 2014].

[27] http://www.whatwg.org/specs/web-apps/current-work/multipage/ [Accessed 9 Aug. 2014].

[28] This End Up: Using Device Orientation - HTML5 Rocks. [online] Available at: http://www.html5rocks.com/en/tutorials/device/orientation/ [Accessed 9 Aug. 2014].

[29] W3schools.com, (2014). HTML5 Canvas. [online] Available at: http://www.w3schools.com/html/html5_canvas.asp [Accessed 10 Aug. 2014].

[30] Can I use the HTML5 canvas element? [online] Available at: http://caniuse.com/canvas [Accessed 10 Aug. 2014].

[31] Mahemoff, M. (2010). "Offline": What does it mean and why should I care? - HTML5 Rocks. [online] HTML5 Rocks - A resource for open web HTML5 developers. Available at: http://www.html5rocks.com/en/tutorials/offline/whats-offline/ [Accessed 10 Aug. 2014].

[32] Pilgrim, M. (2014). Offline Web Applications - Dive Into HTML5. [online] Diveintohtml5.info. Available at: http://diveintohtml5.info/offline.html [Accessed 11 Aug. 2014].

[33] Kesteren, A. and Hickson, I. (2014). Offline Web Applications. [online] W3.org. Available at: http://www.w3.org/TR/offline-webapps/ [Accessed 11 Aug. 2014].

[34] Caniuse.com, (2014). Can I use offline web Applications? [online] Available at: http://caniuse.com/offline-apps [Accessed 11 Aug. 2014].

[35] Can I use the HTML5 canvas element? [online] Available at: http://caniuse.com/canvas [Accessed 10 Aug. 2014].

[36] Caniuse.com, (2014). Can I use... Support tables for HTML5, CSS3, etc. [online] Available at: http://caniuse.com/#search=drag [Accessed 11 Aug. 2014].

[37] Flanagan, D. (2011). JavaScript. 1st ed. Cambridge: O'Reilly.

[38] Eich, B. and Mckinney, R. (1996). JavaScript Language Specification. [online] Available at: http://hepunx.rl.ac.uk/~adye/jsspec11/jsrefspe.htm [Accessed 7 Aug. 2014].

[39] Gallacher, T. (2011). tomgco - Tom Gallacher. [online] Available at: http://tomg.co/gyrojs [Accessed 29 Aug. 2014].

[40] Davey, R. (2014). Kiwi.js vs Phaser. [online] HTML5 Game Devs. Available at: http://www.html5gamedevs.com/topic/4281-kiwijs-vs-phaser/ [Accessed 8 Aug. 2014].

[41] GitHub, (2014). photonstorm/phaser. [online] Available at: https://github.com/photonstorm/phaser/releases [Accessed 8 Aug. 2014].

[42] Phaser.io, (2014). Phaser - HTML5 game framework. [online] Available at: http://phaser.io/ [Accessed 8 Aug. 2014].

[43] Graves, M. (2014). GoodBoyDigital/pixi.js. [online] GitHub. Available at: https://github.com/GoodBoyDigital/pixi.js/ [Accessed 9 Aug. 2014].

[44] Pixi.js - 2D webGL renderer with canvas fallback. [online] Available at: http://www.pixijs.com/ [Accessed 11 Aug. 2014].

[45] Davey, R. (2014). How to Learn the Phaser HTML5 Game Engine - Tuts+ Game Development Article. [online] Game Development Tuts+. Available at: http://gamedevelopment.tutsplus.com/articles/how-to-learn-the-phaser-html5-game-engine--gamedev-13643 [Accessed 12 Aug. 2014].

[46] Rogers, C. (2012). Web Audio API. [online] Dvcs.w3.org. Available at: https://dvcs.w3.org/hg/audio/raw-file/tip/webaudio/specification.html [Accessed 11 Aug. 2014].

[47] Stroke.org.uk, (2014). [online] Available at: http://www.stroke.org.uk/sites/default/files/Stroke%20statistics.pdf [Accessed 2 Aug. 2014].

[48] Tiwari, L., Astheimer, P. and File, P. (2004). Considerations in designing games for older people. HCI and the Older Population, p.39.

[49] Novitzke, J. (2008). Privation of Memory: What can be done to help stroke patients remember?. Journal of Vascular and Interventional Neurology, [online] 1(4), p.122. Available at: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3317323/ [Accessed 2 Aug. 2014].

[50] Davey, R (2013). Tutorial: Making your first Phaser game. [online] Available at: http://www.photonstorm.com/phaser/tutorial-making-your-first-phaser-game [Accessed 14 Aug. 2014].

[51] Davey, R; HTML5 Game Devs Forum, (2014). Phaser function order, reserved names and special uses - Phaser. [online] Available at: http://www.html5gamedevs.com/topic/1372-phaser-function-order-reserved-names-and-special-uses/ [Accessed 14 Aug. 2014].

[52] Mozilla Developer Network, (2014). Detecting device orientation. [online] Available at: https://developer.mozilla.org/en-US/docs/Web/API/Detecting_device_orientation [Accessed 25 Aug. 2014].

[53] Stroke-rehab.com, (2014). Hand Exercises for Stroke Patients. [online] Available at: http://www.stroke-rehab.com/hand-exercises.html [Accessed 27 Aug. 2014].

[54] Nhscareers.nhs.uk, (2014). Neurophysiology - NHS Careers. [online] Available at: http://www.nhscareers.nhs.uk/explore-by-career/healthcare-science/careers-in-healthcare-science/careers-in-physiological-sciences/neurophysiology/ [Accessed 29 Aug. 2014].

[55] Kolb, B. and Whishaw, I. (2003). Fundamentals of human neuropsychology. 1st ed. New York, NY: Worth Publishers.