2012

ANDROID TABLETS IN THE REHABILITATION
OF CHILDREN WITH HEMIPLEGIA

DAVID SIMKIN (09107100)
NEWCASTLE UNIVERSITY
COMPUTING SCIENCE
SUPERVISOR: DR GRAHAM MORGAN

WORD COUNT: 12,953

11TH MAY 2012

[ANDROID TABLETS IN THE REHABILITATION OF CHILDREN WITH HEMIPLEGIA]

This dissertation researches the efficacy of using Android Tablets in the rehabilitation of children with hemiplegia, utilising gesture recognition and touchless interfaces in the rehabilitative process in order to provide an effective and enjoyable set of exercises for affected children to utilise. A summary of the current research within this area is outlined, and the development of a general purpose Android tablet storybook engine is documented. The author's hypothesis is tested by examining the performance of gesture recognition algorithms running on an Android tablet. The results and subsequent analysis are presented within this dissertation.

DECLARATION

"I DECLARE THAT THIS DISSERTATION REPRESENTS MY OWN WORK, EXCEPT WHERE OTHERWISE STATED"

Signed: _____

Dated: _____

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr Graham Morgan and my friends and family.

TABLE OF CONTENTS

TABLE OF CONTENTS

Abstract ..... **Cover**

Declaration **1**

Acknowledgments **2**

Chapter 1: Introduction **6**

Introduction 6

Dissertation Structure 6

Overview and Motivation 6

Hypothesis and objectives 7

Plan 7

Summary 7

Chapter 2: Prior Research and Background Information **8**

Introduction 8

Hemiplegia 9

Gesture Recognition 10

Interactive Storybooks 11

Summary 12

Chapter 3: System Design and Implementation **14**

Introduction 14

Requirements 14

System Architecture 15

Software Development Model 15

Tools and Technologies Used 16

Java 16

Android and the Android SDK 16

OpenCV 16

JSON 17

Current Generation Android Tablets 18

User Interface 18

Android Fragments 19

Implementation of Gesture Recognition 20
Overview 20
Varieties of Gestures Implemented 21
Algorithms and Techniques Used 21
Generic and Reusable Storybook Engine 22
Book List 22
File Format 24
Page Loader and Parser 26
Page Inflator 27
Gesture Layer 28
UI Layer 29
Summary 30
Chapter 4: Testing and Results **32**
Introduction 32
System Analysis 32
Environment Testing 32
Visual Quality of the System 32
Performance 34
Gesture Fidelity 36
Results 37
Final System Performance 37
Results Analysis 37
Summary 38
Chapter 5: Evaluation and Conclusion **40**
Introduction 40
Conclusion 40
Satisfaction of Hypothesis and Objectives 40
Satisfaction of the Hypothesis 42
Evaluation 43
Time Plan 43
What has been Learnt 43
Aspects of the Project that Went Well 44
Aspects of the Project that Could be Improved 44
Further Work 44

Summary 45

Bibliography **46**

Table of Figures **48**

Appendices **50**

CHAPTER 1: INTRODUCTION

INTRODUCTION

This chapter introduces the motivation upon which this project is based, outlined by discussing the author's hypothesis, and how this resulted in the project aims and objectives and the structure of this dissertation. The requirements of the system will also be discussed based upon completing the objectives.

DISSERTATION STRUCTURE

This dissertation is split into five chapters, introduction, prior research and background information, system design and implementation, testing and results, and the evaluation and conclusion. Each chapter begins with an introduction and ends in a summary.

The introduction gives some background to the project and attempts to set out the motivation for this dissertation and the implementation plan.

The chapter on prior research and background information explains key terms used throughout this dissertation and explores past research to impart an understanding of why this research is relevant and adds understanding in the field.

In the system design and implementation chapter the design decisions for the final system are explained at a high level and the techniques and technologies chosen are justified. The requirements for the system are also detailed.

Testing and results sets out tests and their results that were designed in order to evaluate system correctness and to satisfy the hypothesis.

Finally the evaluation and conclusion attempt to use the results outlined in the previous chapter to satisfy the hypothesis, explain areas where the project could have been improved and where future work can be carried out.

OVERVIEW AND MOTIVATION

Android, Google's flagship mobile operating system (OS) has taken the world by storm; with over 200 million handsets running a variant of Android it is now the planet's most popular smartphone OS (Nielsen, 2011) and it continues to grow into other distribution avenues, including TV set-top boxes and now 'tablets', touch screen, large-format devices for use around the home. However these new and powerful devices are mainly used for the consumption of content and often not for any true practical purpose. (comScore, 2010) My experience developing and deploying applications for Android piqued my interest in this project. I have the knowledge and the skills to create applications which leverage the strengths of the platform and can hopefully help in a real-world application, in this project's case: the rehabilitation of children with neurological disorders such as hemiplegia.

Current methods used in the treatment of hemiplegia involve long periods of exercises such as stretches, clenches and other specially designed actions which help affected children by increasing confidence and building muscle. However these exercises often lack context for the children, they can be boring and require a parent's supervision and active support to be effective. (Schaechter, 2004) With digital devices such as the iPad and Android tablets through the use of their cameras, powerful processors and flexible frameworks it should be possible to create applications which can help parents and their affected children perform their exercises in a more engaging and fun manner, and hopefully increase the rate at which the children's motor functions improve.

However, tablet technologies are in their infancy and processing-heavy applications such as those involving image manipulation and gesture recognition are relatively untested but research shows they hold great promise on the platform. The aim of this project will be to investigate and build a prototype to test the capability of current generation Android devices with the processor-heavy operations involved in image manipulation, and to evaluate the benefits of such devices in the rehabilitation of children with hemiplegia.

This provides the motivation for the project, both on a personal and academic level; there is little research into using such mass market and mobile devices in the rehabilitation process, and it would be immensely satisfying to explore the work done by others in order to create a piece of software which can be used by healthcare professionals to directly improve the quality of life of an affected child.

HYPOTHESIS AND OBJECTIVES

Hypothesis: Android based tablet devices are suitable and effective at the rehabilitation of children with neurological disorders (specifically hemiplegia)

Objectives:

- Investigate current techniques involved in the rehabilitation of children with hemiplegia
- Identify areas in which traditional efforts are lacking or incapable of reaching
- Build an Android Tablet application involving gestures, colours and leveraging the advantages of a digital medium to help the children with their exercises while being enjoyable.
- Compare the traditional approach with the interactive Android based system and evaluate strengths and weaknesses.

PLAN

Figure 1 outlines the expected time plan for the project.

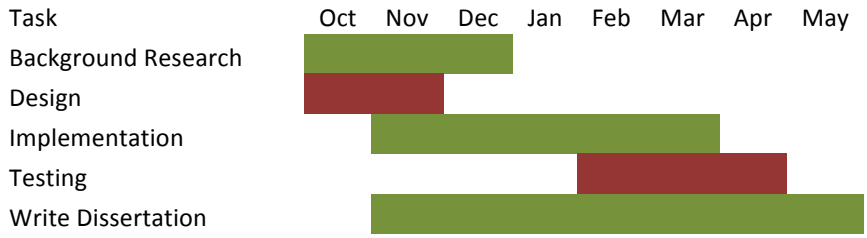


Figure 1: Project timescale and work order Gantt Chart.

SUMMARY

This chapter has set out the structure of this dissertation and has explained the motivation behind the project. It has also identified the issues that are present in developing an application which incorporates gesture recognition:

- Tablet technologies are in their infancy, with low resolution cameras, relatively weak processors and early software platforms to work on.
- Gesture recognition and image processing libraries have limited support for Android devices.
- Hemiplegic patients are only capable in certain cases of small movements, which can be very difficult to detect with a camera.

CHAPTER 2: PRIOR RESEARCH AND BACKGROUND INFORMATION

INTRODUCTION

This chapter introduces the concepts and prior research that are related to this project. hemiplegia is defined and the importance and methods of treatments is examined, and gesture recognition is explained and several techniques are analysed in order to determine which approach is most suitable for the project's implementation.

HEMIPLEGIA

Hemiplegia is a neurological disorder in which an individual, following some sort of trauma, such as a stroke or illness, has total paralysis of the arm, leg, and trunk on one half of their body (HemiHelp). Hemiplegia literally means "half struck," and individuals suffering from the disorder undergo extensive rehabilitation in order to improve motor function in the hopes of them leading a full life. (Bach-Y-Rita, 1990)

Rehabilitation for hemiplegia has been standard practise in medicine for over fifty years and has proved to be extremely effective in improving quality of life. One study shows that, in adults, only 8% of patients are able to dress themselves independently and 73% of patients require rehabilitation before they can be discharged following a stroke. However, following intensive rehabilitative exercises and physiotherapy, 65% of patients were able to return to work. The study also showed a correlation between the success of rehabilitation and age, so younger individuals can be expected to reap greater benefits. (Millard, 1966)

Hemiplegic patients who go without rehabilitative treatment will often end up developing a contracture of the shoulder or wrist. This is when the muscles in the joint begin to permanently contract, limiting movement range and causing a lot of pain to patients (Snels, Dekker, van der Lee, Lankhorst, Beckerman, & Bouter, 2002). It has been hypothesised that by exercising the affected limb the patient will be able to increase joint mobility and avoid a contracture developing. Contractures have also been linked to a resistance in passive movements by patients, which would decrease the patient's quality of life significantly because fewer day to day tasks would be able to be carried out unsupervised. (de Jong, Nieuwboer, & Aufdemkampe, 2006)

Conventional rehabilitation of hemiplegic patients involves a wide range of approaches designed to target all areas of recovery for the individual. These include neurofacilitation techniques, task-specific training, and task-oriented training. Neurofaciliation techniques involve inhibiting areas of stronger movement, encouraging the individual to use their affected limb and not their normal limb; a therapist may hold the normal limb, for example, forcing the child to use their weaker limb to do some task. Task specific training is doing exercises which directly mimic a task or encompass the movements of some specific task, such as doing up buttons or brushing teeth. Task-oriented training involves general training such as grasping, movement on a treadmill and so on, in order to improve the basic actions that are used in many different tasks. (Schaechter, 2004)

While performing gestures has been shown to help with the physical effects of hemiplegia, approximately 35% of patients who are diagnosed with hemiplegia are often diagnosed with additional diagnoses such as perceptual problems, learning difficulties and other behavioural problems (Frampton, Yude, & Goodman, 1998). As the child's brain is young and developing it is possible for other areas of the brain to take over tasks which would normally be carried out by now damaged areas of the brain. By providing mental stimulation through bright colours, interactivity and basic problem solving skills it is possible that an interactive storybook

application may help in more than just the use of gestures, the mental stimulation may help in the rehabilitation of the other learning or perceptual diagnoses.

GESTURE RECOGNITION

Gesture recognition falls under the umbrella topics of Image Processing and Computer Vision, and because much research has gone into those areas and is on-going there are a large series of accepted algorithms, approaches and processes that can be used in a variety of tasks As a significant amount of groundwork and implementation in the field was being duplicated by various research teams, Intel developed the OpenCV (Open Source Computer Vision Library) in order to prevent researchers from having to re-tread established ground. OpenCV was released under the BSD open source license and is now maintained by Willow Garage. While being originally written in C, there have been a variety of ports to different languages and systems, including Python and Java, and now Android. It is that variant of OpenCV which has been the topic of much previous research into the viability of the Android platform to run image processing and gesture recognition tasks. (Willow Garage, 2011)

There exists a large body of research into gesture recognition (Manresa, Varona, Mas, & Perales, 2000), a small amount into the viability of Android as a platform for those gestures (using OpenCV), and no published research into using Android Tablets with OpenCV. OpenCV was found to be a viable base to use for the recognition of a wide variety of gestures, but that Android phones struggle to meet the processing requirements and ran at such a low frame rate (around 0.16 frames per second) that effective gesture recognition was limited (Cooper), because many algorithms for gesture recognition examine the difference between frames, and if the processing rate is so slow large jumps in movement begin to appear the accuracy begins to break down (Yu, 2010). That research was carried out several years ago however, and current generation Android phones and tablets now have dual core processors with clock speeds greater than 1Ghz, a major step up from the single core 600-800MHz processors that the application of much prior research is based on.

Previous research has pointed to a variety of different techniques for the isolation of gestures, each with different complexities and associated processing costs and different granularities in the variety and detail of gestures that can be detected. For a device with a relatively slow processor, such as a mobile device, it is therefore necessary to balance the rather fine level of gesture detection needed to recognize the small movements that some hemiplegia sufferers are capable of with the processing power needed to run these in real time. One method explored in previous research is tracking the pixels which change between each frame to within some threshold. This has the advantage of being able to potentially pick up small movements, but in terms of determining what is a finger and what is an arm this approach is ineffective. It also suffers from background interference as any movement of the device's camera would render gesture tracking impossible with this method. (Denman, Chandran, & Sridharan, 2007)

A second technique that has been researched is colour tracking, isolating patches of colour between two different shades and using them as known tracking points. For example a child with a red sticker on one finger and a green sticker on another finger could perform a variety of gestures with a high degree of fidelity. The downside of this approach is that it is more computationally intensive and for each colour tracked requires a separate scan of the image so only one or two colours could feasibly be tracked by a mobile device, however it should provide a good base algorithm for the tracking of different gestures. (Munoz-Salinas, Aguirre, & Garcıa-Silvente, 2007)

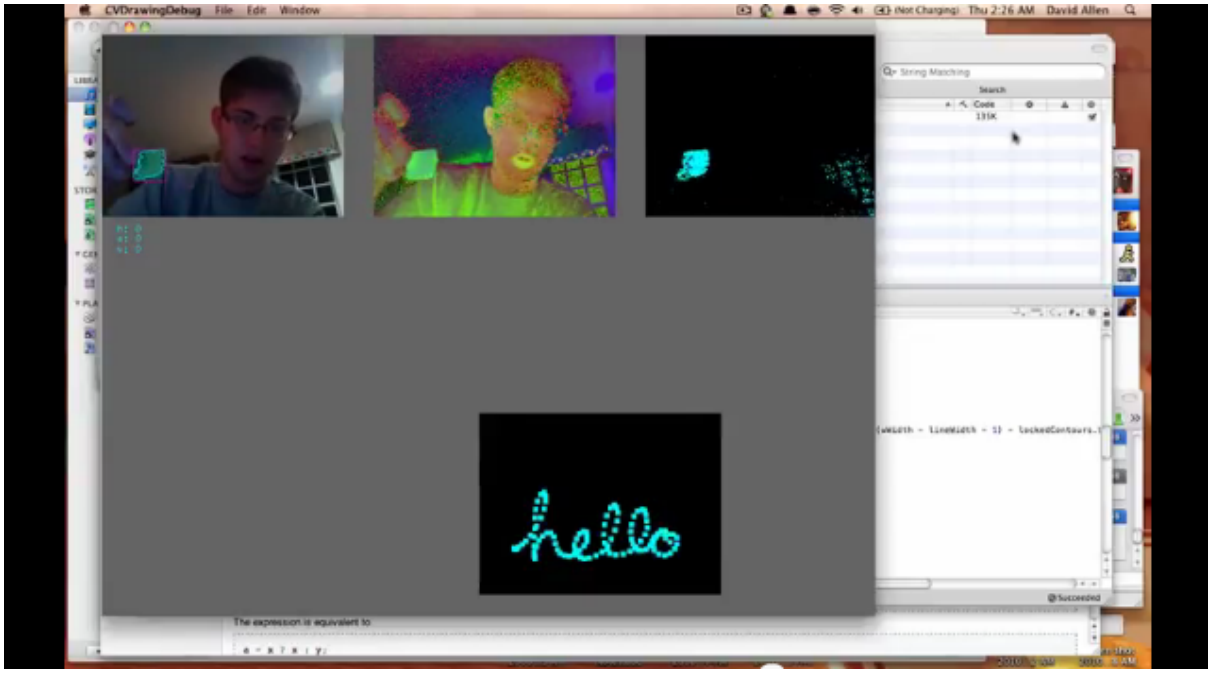


Figure 2: Using colour tracking to "write" a word by tracking the path of a piece of card. (davidalln, 2010)

A third approach is called Canny Edge Detection, developed by John F. Canny in 1986, and an example of which can be seen in Figure 3. This involves using a multi-stage algorithm to detect a wide variety of edges in images and the processed image can be segmented into "blobs" for the tracking of individual elements. This approach however, due to its multi-algorithm approach is very expensive and can only feasibly be used in an offline setting, taking over thirty seconds to complete one sweep on an Android phone. (Toytman & Thambidurai)

Figure 3: An example of Canny Edge detection on an Android device.

While there are few working implementations of gesture recognition and touch-free interfaces on Android devices, on other platforms they are a commercial reality. For example Microsoft's Kinect uses a colour camera in tandem with an infra-red emitter and sensor to build up 3D maps and detailed skeletal information about its

users, allowing for accurate movements and gestures to control their XBOX 360 platform. While this allows very accurate skeletal models and other gesture information there is no feasible way to use this technology on an Android device which uses a standard mobile camera sensor, and all information will have to be gleaned from processing standard flat images. There are also many research projects on-going which are looking into the capabilities of the Kinect in the rehabilitation of stroke victims and other disorders in which exercises and physiotherapy can be beneficial in improving the individual's quality of life. (Kajaani University of Applied Sciences, 2011)

Figure 4: Using a Kinect device with an XBOX 360 to navigate a user interface without any traditional physical input.

However like many implementations of this technology the XBOX 360 has a powerful processor and large amounts of RAM, unlike the mobile devices that will be explored through this research and as such there is a larger degree of processing which can be done on an XBOX device and still give real-time response speeds which cannot be attempted on a mobile phone or tablet.

INTERACTIVE STORYBOOKS

An interactive storybook is a story, usually aimed at children, which incorporates graphics, sounds and other elements such as quizzes or tasks to complete which advance the story. In the past they have usually been published on CD-ROM, but with the advent of unified software stores for major devices (Apple's App Store or the Android Market) there is a new avenue for the distribution of such stories.

The use of interactive storybooks has been shown to stimulate language skills in young children, (Mol, Bus, & de Jong, 2009) however other studies have shown that some students will simply play with the graphics or sounds of the storybook and will not actually read the book themselves (Lefever-Davis & Pearman, 2005). However there has been no previous research into using interactive storybooks in the rehabilitative process, and how feasible and effective it would be.

There are many storybook applications currently available for Android devices, however relatively few incorporate truly interactive elements aside from basic swipes to change the pages and buttons to click. The applications available also tend to each be self-contained stories, often based on a real-world children's book and not storybook platforms, which would provide greater flexibility in providing multiple stories to the user.

SUMMARY

While there are various different ways of approaching touchless input and gesture recognition, due to the hardware limitations of an Android tablet performance will have to be first consideration. Broadly colour tracking appears to be the best method of tracking, providing a good approximation of the location of fingers while not being too computationally expensive compared to other methods. It also appears that most applications on the market today for interactive storybooks are single-story applications, and not publishing platforms, and so the application built as a part of this project should provide simple access for a user or publisher (in a relatively simple file format) in order to create and share or sell their own stories.

CHAPTER 3: SYSTEM DESIGN AND IMPLEMENTATION

INTRODUCTION

This chapter outlines and explains the design decisions made and the process followed in order to develop the final system. Based on the research carried out in Chapter One a series of design decisions were made and this chapter will explain the relative advantages and disadvantages and the reasoning behind their use. This chapter will also discuss the implementation of the system.

REQUIREMENTS

Based on the hypothesis and aims outlined in Chapter 2 a series of requirements have been formulated for the system and they are outlined below.

1. The system must support a variety of gestures.
    1.1. Gesture recognition must be real-time and responsive.
    1.2. Gestures must reflect a number of different clinical exercises.
    1.3. Gestures must be accurate and have a low number of false positives.
2. The system must have a generic engine which can support a variety of different stories and graphics.
    2.1. The system should load these stories from a file-format.
        2.1.1. Stories stored in this file format must be scanned for on the storage of the device and displayed in an easy-to-browse format.
    2.2. The system must scale to different tablet screen sizes.
    2.3. These stories must be interactive.
        2.3.1. The system must include a number of different generic pages from which interactivity can be derived, with a number of tasks and games for the user to play.

SYSTEM ARCHITECTURE

The system will have several major parts; the book file which is resident on the file system of the device, the page loader which reads the file, parses the file format and creates a “book” of “pages” which are then inflated on demand and the gesture and UI layer which form the interactive and visual components of the system. This is illustrated in Figure 5.

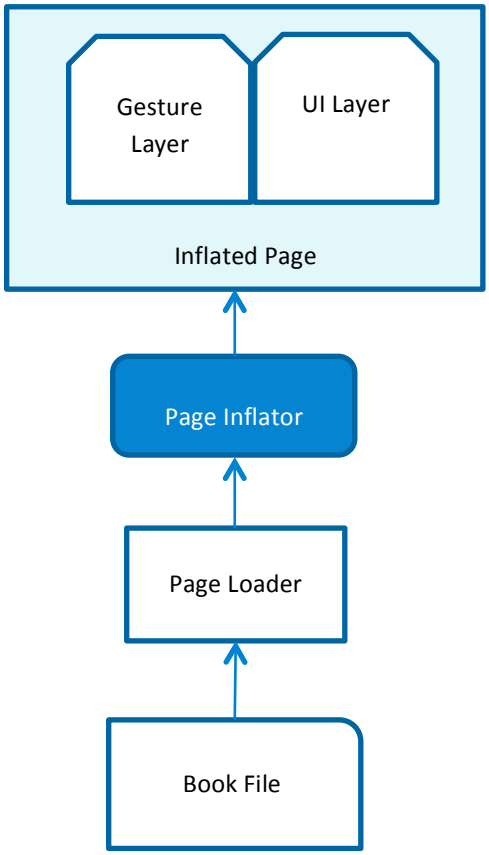


Figure 5: High level overview of the system

The different components of the system will be examined in more detail later in this chapter.

SOFTWARE DEVELOPMENT MODEL

The development model chosen for the development of this project is the agile approach. The agile approach promotes planning and an evolutionary approach to software development, and because the project as a whole has rapidly changing focuses and an experimental approach to development the agile method embodies a closer and more realistic process to other, more rigid approaches.

The development consisted however of four phases:

- System design
- Storybook engine development
- Integration with existing system
- Addition of more generic pages

There were considerable changes made to the overall design at all stages of development however as the project shifted and changed from feedback and external pressures, but the overall stages of development

progressed fairly linearly, starting with the design of the system and moving forward through development and integration.

TOOLS AND TECHNOLOGIES USED

JAVA

Java is a programming language originally developed at Sun Microsystems, and is now owned and maintained by Oracle. Java code, instead of being compiled down to machine code is compiled into Java byte code, which is intended to be run by a Java Virtual Machine (JVM) which is written specifically for the host hardware. This allows a "write once, run everywhere" approach to Java development as the JVM is designed to abstract away hardware and low level software differences between platforms.

Java has been chosen to be the principal development language of this project because while Android devices have support for natively compiled languages (C, C++ etc), it is strongly recommended by the Android team to use the Java Android API in order to build applications with quick support for different screen sizes and hardware profiles, and full support for Android platform features (Google, 2012).

ANDROID AND THE ANDROID SDK

Android is a free, open source operating system (OS) for ARM based devices, usually mobile phones or tablets. Development of Android is done in-house by Google and each release of Android is accompanied by an API and SDK which exposes core features of the OS, such as the utilization of hardware features like the camera, microphone or notification LEDs, and other operating system specific features such as Intents, Views and Fragments which allow developers to create applications which can interact and interoperate.

In order to develop applications for Android, the SDK must be used to enable applications to run on the device, and while the principal language is Java for the development of Android applications, the structure of an Android application is significantly different and far more event driven than a typical Java application.

The API exposed through the SDK will provide all the lower level functions of the application, such as hardware acceleration, UI drawing, access to the camera and microphone and other operating system features. There are a number of different API versions, and the version picked for the development of this system is API level 11, which corresponds to Android 3.0 (Google, 2012), the first Android version to be compatible officially with tablets. All features used in this API level will be forward compatible, and the most important features have been provided backwards compatibility through the development of a compatibility library. (Google, 2011)

OPENCV

OpenCV is a C and later C++ library of programming functions for applications mainly in the field of computer vision and image manipulation. It includes various functions for the processing and manipulation of images and video, and was designed to prevent individuals from having to implement common algorithms and approaches themselves. OpenCV is released under a permissive BSD open-source license which allows its use in both commercial and non-commercial projects.

OpenCV for Android comes in two variants, the C++ version in a Java wrapper and a native Java version. The two different distributions have different advantages and disadvantages as shown in Table 1.

Distribution	Advantages	Disadvantages
OpenCV (with Java wrapper)	Speed.	Frequent native calls can be slower due to the cost of translation between Java and C++. (Google, 2012) Silently fails.
JavaCV (native Java version)	Stack traces and other debugging features. Better support for Android defaults and constructs.	Much slower.

Table 1: Table comparing advantages and disadvantages of OpenCV and JavaCV

Despite JavaCV appearing to have more advantages than disadvantages it was not chosen for the implementation of the final system, because speed and the ability to run complex algorithms in a real-time scenario was more important than the other factors, which could be worked around.

A test was carried out, with the same simple image-recognition program written in both JavaCV and OpenCV and the results in Table 2 below show the difference in performance.

Distribution	Frames-per-second
JavaCV	3.1
OpenCV	17.3

Table 2: Table showing the difference in performance between JavaCV and OpenCV

JSON

JSON (JavaScript Object Notation) is a lightweight text format which has been designed to be human readable. While the name mentions JavaScript JSON is independent from the Javascript and parsers are available in many languages, including Java.

Figure 6 shows an example JSON file which could be parsed to retrieve the data encoded in it.

```
{
	"foo":"bar",
	"example": {
		"example_text":"hello world"
		}
}
```

Figure 6: Some example JSON

JSON has support for "objects" and arrays, and parsers allow the traversal and mapping of objects to objects in whichever language the parser is written for. JSON formatted documents are also easy to read, understand and edit, and as a result is an excellent choice of data standard to represent the book structure and data.

There were two main choices for JSON parsers to be used on the project, the standard Java/Android parser contained in the `org.json` package, or Jackson, a third party Java JSON parser. Jackson was chosen because it provides a cleaner API and much faster performance across the board (Adamek, 2011), which is especially important on Android, as any slowdown in performance can cause system unresponsiveness.

CURRENT GENERATION ANDROID TABLETS

Because Android is an open source operating system there are a variety of different Android tablets available on the market and there is a lot of innovation and advances currently happening in the field of mobile processors and other technologies. However many current-generation tablets have broadly similar specifications, as shown by Table 3.

Name	Processor	RAM	Front-Facing Camera?	Android Version	RRP
Asus EeePad Transformer Prime TF201	Nvidia Tegra 3 Quad-Core 1.3GHz	1GB DDR2	Yes	4.0	£499.99
Asus EeePad Transformer TF101	Nvidia Tegra 2 Dual-core 1GHz	1GB	Yes	3.2	£429.99
Motorola Xoom	Nvidia Tegra 2 Dual-core 1GHz	1GB	Yes	3.2	£479.99
Samsung Galaxy Tab 10.1	Nvidia Tegra 2 Dual-core 1GHz	1GB	Yes	3.1	£479.99

Table 3: Table comparing current generation Android tablets

Several tablet devices have been left off this table, these devices are not considered true tablets because they are running older custom versions of Android (v2.1-2.3) and have no official Google support. As a result the definition of an Android tablet is constrained to include only those running Android 3.0 and higher, devices which are considered by Google to be “tablets”.

The Motorola Xoom has been chosen as the development device in order to target the lowest class of processor present in current tablets. This should allow for good performance across all future tablets as the Nvidia Tegra 2 is the lowest class of system on a chip processor supported by Android 3.0.

USER INTERFACE

There is no set layout for the user interface, except what Android stipulates as a part of the operating system. This is because the nature of an interactive storybook is that there can be interchangeable graphics, text, colours and other interactive elements. As a result the engine is designed to allow publishers and individuals to create stories which have little semblance to each other. The processes and explanation of this part of the user interface is elaborated on later in this chapter.

However Android stipulates several user interface features that are shown in Figure 7.

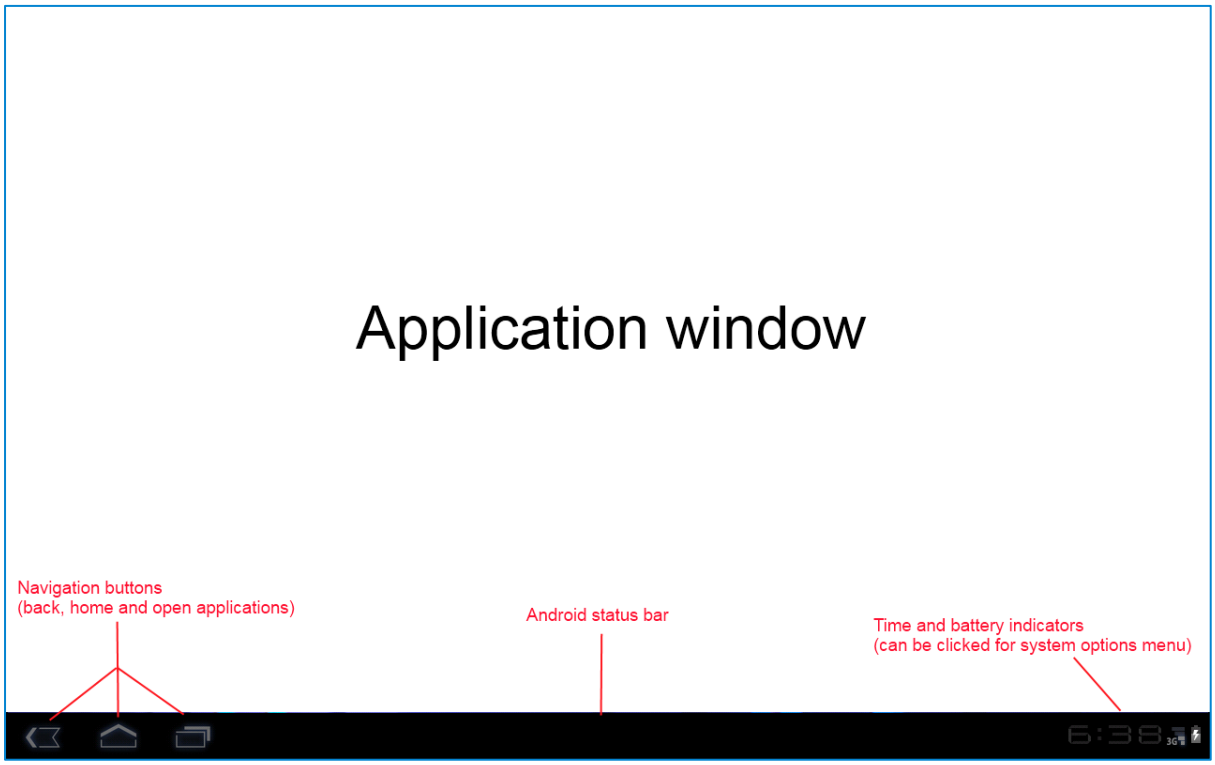


Figure 7: Standard Android user interface elements.

There are two distinct view elements, the application's window which can be changed by an application programmer, and the Android status bar which is an operating system owned view and is responsible for various system tasks, including application navigation through a series of call-backs to the application in focus and access to system-wide settings and options.

ANDROID FRAGMENTS

Fragments are an Android API introduced in Android 3.0 specifically and is available for earlier versions of Android through a freely available compatibility library. The idea of Android Fragments was to support multiple interface elements which can be dynamically added and removed and can be visible on screen at any one time. These "Fragments" of the user interface, such as a list of names in a `ListFragment`, a Fragment containing a `WebView` showing a webpage and so on are a necessary abstraction to allow for attractive user interfaces on large-screened devices such as tablets in a manner that makes sense to a developer (Hackborn, 2011). A Fragment also has other advantages, including facilitating a global application state and as a result the Fragment API has been chosen as an integral API in the project. An example of a user interface made up of Fragments is shown in Figure 8.

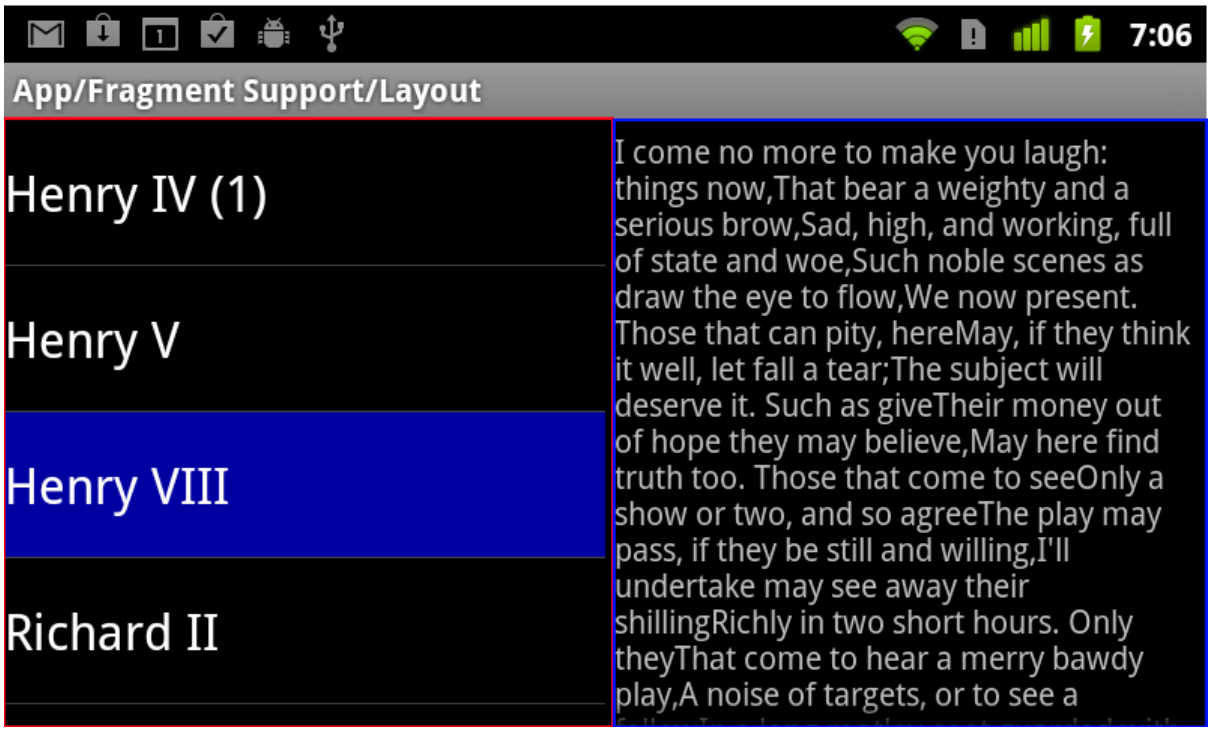


Figure 8: Android UI highlighted with red and blue showing different user interface Fragments

In the application's engine Fragments are used in order to maintain a global state and also in order to pass Page objects into their constructor to allow for their inflation based on a set of generic parameters, this would not be possible in the traditional Activity based application on which earlier versions of Android are based.

IMPLEMENTATION OF GESTURE RECOGNITION

OVERVIEW

The word "gestures" has a variety of definitions, but for the purposes of this application a gesture is defined as hand gesture which ranges from simple actions of pointing and moving to more complex gestures such as clenching, gripping, flexion and the twisting of the wrist. In order to provide useful rehabilitative benefits to a child with hemiplegia any gesture recognition must be able to detect with reasonable accuracy any desired gesture, and it must be able to do this by examining relatively small movements as affected children with hemiplegia may only be capable of low hand mobility.

OpenCV does not provide any support for gesture recognition at a library level, it simply provides helper functions for the processing and extraction of details from images, which is essential in the larger application of gesture recognition. The basic gesture workflow is to use the Android camera callback to receive a frame from the video whenever one is available, the frame being stored in a buffer. The camera still is then processed and converted into a format that OpenCV recognises, and then areas of colours are extracted and can be used to recognise any movement from frame to frame. This process is similar for most gestures, the actual processing of the image is separate from the gesture recognition and therefore the core can be reused for all gestures.

VARIETIES OF GESTURES IMPLEMENTED

A variety of gestures were implemented, all of which already form a role in the rehabilitative process and as such would have a proven effectiveness in helping to improve the motor skills of children with hemiplegia.

- Basic hand movement – tracking of the relative position of the hand on the screen, allowing for mouse-like pointer interaction with UI elements.
- Flexion and extension – flexion is the bending of the wrist to decrease joint angle, and extension is the opposite of this.
- Grasp and release – grasping an invisible object and releasing it in a clasping motion.
- Pinch and grasp – similar to grasp and release but only using the finger and thumb to pick up an object on-screen.
- "Roly-poly" – moving hands in a circular motion around a central point similar to winding up a ball of string.
- Scurry – moving one hand and arm up in tandem with moving the other hand and arm down and repeating, similar to the scurrying of a rabbit or a dog.
- Supination – twisting the wrist from a position of the palm facing up to facing down.

These gestures are designed to carry rehabilitative benefits and emulate those that are used in the current rehabilitative process (Millard, 1966). By allowing for a table to detect these gestures you can begin to relax the supervision aspect of rehabilitation and allow for older children to gain feedback on gestures they would otherwise be performing in a clinical context.

ALGORITHMS AND TECHNIQUES USED

The method chosen to track gestures is colour segmentation due to the relatively precise tracking with a relatively low processing overhead. Colour segmentation involves taking a camera still, considering every pixel in the image and seeing if it falls within two colour scalars, if it does then the pixel is kept (and replaced with a white pixel), otherwise it is discarded (replaced with black). This produces a black and white image as shown in Figure 9, which can be processed by an algorithm and compared to the results from previous frames to determine if a gesture has taken place. A full breakdown of the performance advantages of the colour segmentation approach are present in Chapter 4 along with benchmarks supporting the decision.

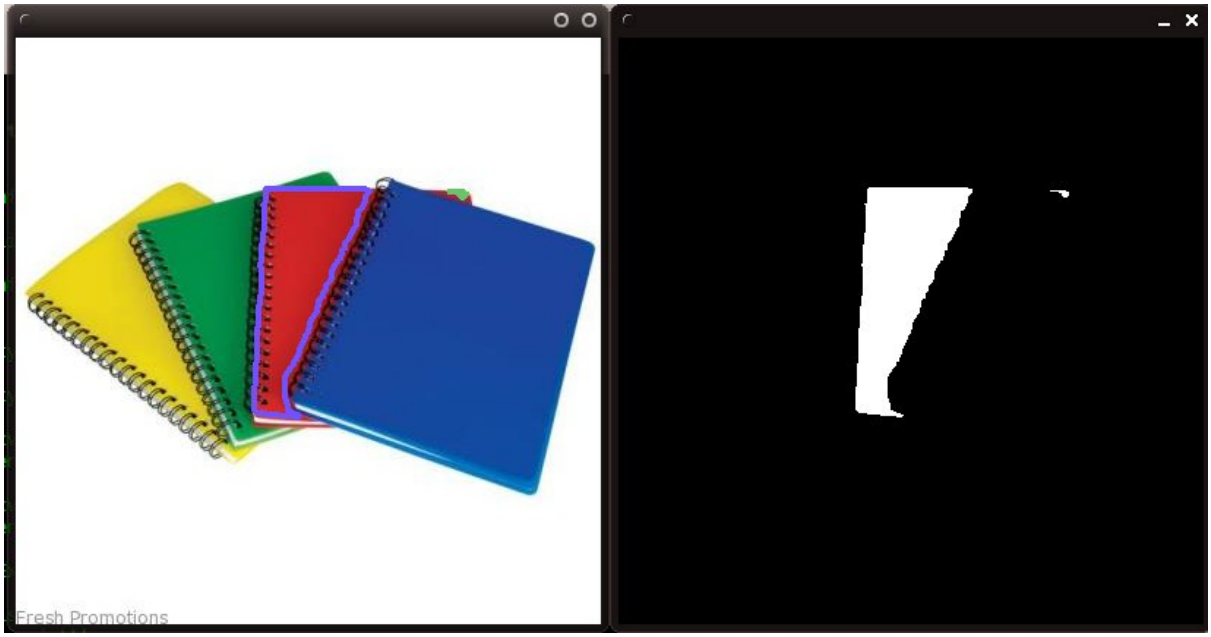


Figure 9: Colour segmentation of an image to extract the red component. (picture from http://blog.damiles.com/wp-content/uploads//2010/01/captura.jpg)

All algorithms are derived from a common base, and at its most simple is the algorithm for just tracking hand movement independent of any gestures which may be performed. This gesture is listed as “basic hand movement” above, and is the simplest gesture that can be performed and all further gestures are simply extensions of the algorithmic process. The algorithm tracks the movement of the hand by segmenting the screen using two colour scalars representing the coloured marker on the child’s finger and then using OpenCV to extract “blobs”, which are areas of contiguous white colour in the image. The blobs can be examined and the blob determined to be the correct size and shape is assumed to be the finger and the position of this blob determines the screen coordinates at which to display a pointer. Due to the effect of external lighting on the camera input these colour scalars represent a band of colour which may be picked up to within some tolerance, so an effective red scalar would pick up dark reds to very light reds in order to give the best results.

All other gestures are an extension of this, and while the basic hand movement tracking does not track movement over time, just the current frame, all other gestures need to compare processed results across a number of previous frames in order to be of any use. For example the algorithm for the grasping gesture uses a state machine and by measuring the distance between the index finger and thumb it can be determined if a clench is being performed by switching through states depending on the distance.

GENERIC AND RESUSABLE STORYBOOK ENGINE

BOOK LIST

Because the storybook engine must be capable of loading and displaying a variety of different stories from different publishers there is a need for the user to have an easy way of browsing and selecting from a list of books which currently reside on their device. This necessitates a book list of some form, and there are several examples of such lists already on Android, including the Kindle application (Figure 10).

Figure 10: Amazon's Kindle software running on an Android tablet (picture from http://www.everythingabouttablets.net/wp-content/uploads/2011/04/wpid-P20110422160155.png)

As books would be stored on the device's internal storage or SD card the application would have to scan the storage directory for any files with a file extension matching .nbk, and then read those files to access the metadata file contained within, using this information to load a thumbnail and some basic information (title, author and so on) so that the user can select which book they most want to read.

As shown in the user-interface prototype in Figure 11 the application displays the book's title, the author's name and a thumbnail designed to draw the eye.

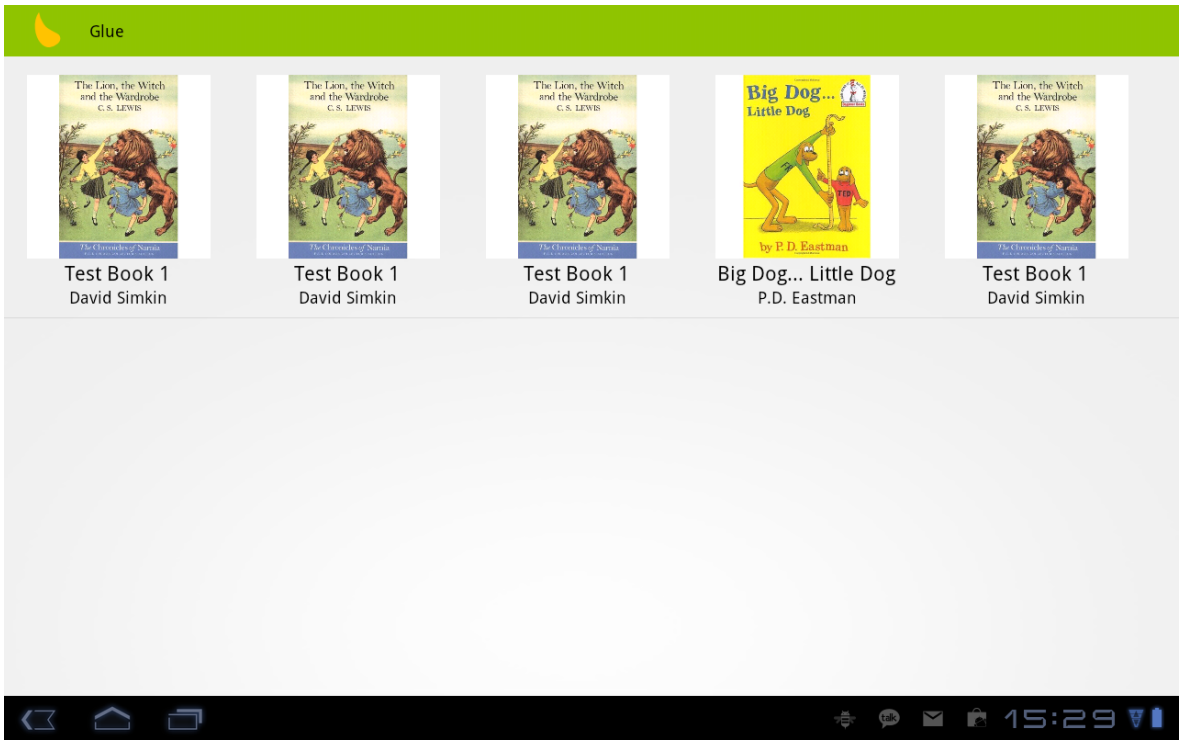


Figure 11: An early prototype of the book selection screen.

This satisfies requirement 2.1.1, and allows the user to select from multiple stories which may be present on the user's device.

FILE FORMAT

Because the interactive storybook application has been designed to allow different stories to be created by a publisher or user this means there needs to be some standard way of loading a story into the application. In order to allow the easy duplication and transmission of stories a simple file format was chosen as the most logical and end-user understandable method of doing so. The application loads stories from a file format (with a .nbk extension) which is split into four parts, the wrapper, the "book" file, art assets and other resources, and the metadata file. The wrapper is simply a zipped container for the latter three parts in order to encapsulate all the data in one file for ease of use and organization for the user. Figure 12 shows a high level overview of the book format.

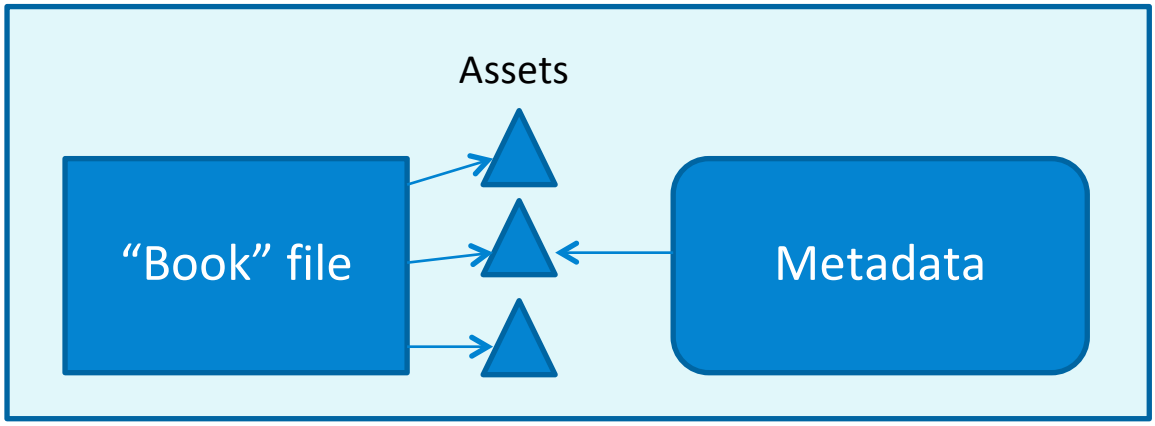


Figure 12: Diagram of the book format

Assets can be any piece of media, be it an image, a video, or a sound file or so on. These are referenced by their location relative to the root of the file (eg. /images/example.bmp). These references are used by both the book file and metadata file.

The book file is a JSON formatted plain text file with the extension '.nbe' and contains all the information needed to inflate and create any page needed by the story, including links between the pages and references to assets where needed.

```
{
   "kind":"Book",
   "data":{
      "num_pages":1,
      "pages":[
                  {
                    "page_type":"main_menu",
                    "page_num":0,
                    "menu_items":[
                         {"type":"menu_button", "text":"Start", "page":1},
                      {"type":"menu_button", "text":"Credits", "page":2}
                   ],
                     "image":"NaughtyNoodle/Assets/banner.jpg",
                      "title":"Naughty Noodle - Main Menu"
                   }
            ]
        }
}
```

Figure 13: An example book file with one "`MainMenuPage`".

The file is broken up into an array of pages, each which have a type and a page number and other associated data depending on what type of page they are; for example in Figure 13 a `MainMenuPage` may have an image, a title and an array of menu items which may be buttons to perform some task. The parser first considers the page type and then looks for different data items depending on that type. The file may contain anything from a piece of text or an image to colours and relative positions on the screen. The book file is the most important part of the file format as it lists exactly how the user interface and the gesture layer should act on each page in the book, and ultimately makes up the core of the story.

Metadata is defined loosely as “data about data” (NISO, 2004) and the metadata file, in this case, contains a list of the book’s title, author and an asset reference to a thumbnail. The metadata file is used when populating the initial list of books in the application, and can be read quickly and needs very little parsing as it is simply a plain text file with information delimited by line breaks. For the book’s thumbnail there is a reference to an assets file expected. An example metadata file is shown in Figure 14

```
Naughty Noodle
Limbs Alive
thumb.jpg
```

Figure 14: An example Metadata file

Using a file format of some form is necessary in order to allow for different stories to be loaded and for the engine to be of any practical use. The above structure was chosen because it allows for a separation of large assets, layout information and initial metadata while being contained in one, easy to handle file. Using JSON allows for the file format to be easily expanded while maintaining backwards-compatibility, as the addition of new key-value pairs will not affect the parsing of the document. There is one cause for concern with the file format however, if when the files are compressed into the one “.nbk” file if the level of compression is too high there may be a large performance hit during the loading and parsing phase of the application. This can be avoided by using a minimal level of compression on the file when packaging it into the “.nbk” format, which will reduce the time it takes to extract and parse. Using this file format satisfies requirement 2.1.

PAGE LOADER AND PARSER

When a user has selected which book they wish to read the application needs to load and display this book. There are two approaches that could have been taken when designing this part of the application, either streaming the pages from the file, loading each page one at a time and re-accessing the file whenever a new page needs to be loaded, or to load the entire book into memory in an object format and to then simply inflate each view when required.

For this project, the second, whole book approach was chosen, and in order to not run into low memory problems any assets required are simply loaded as needed and only a textual reference to their location is stored, and once the asset is no longer required it is destroyed in order to free the memory. A diagram of the page loading process is shown in Figure 15.

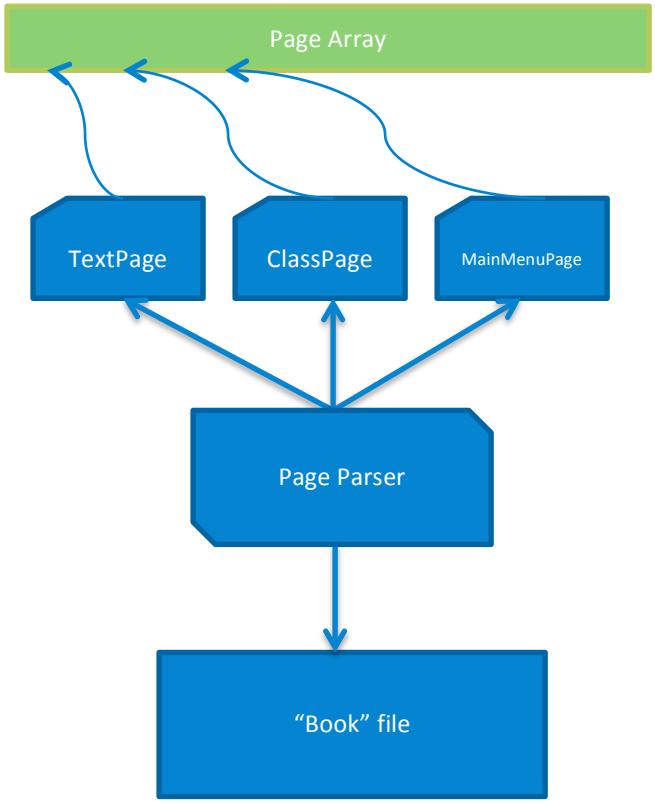


Figure 15: Diagram showing a high level overview of parsing pages.

The book file is first given to the page parser, which then loops through all the data contained within the book file, using a textual key property “type” which is present in each page within the book file, in order to determine which generic type of page object needs to be created in the Android application by the parser (eg `"page_type":"text"` would map to a `TextPage`). Once the generic page object has been created, the object’s properties are then initialised based on the preceding information stored in the page in the book file. The created generic page object is then added to the page array, which makes up a book of pages.

The array, having been filled with pages is then passed onto the inflator which initially inflates the 0^{th} indexed page in the array as the beginning of the book. Each index in the array is treated as the page number, so the 0^{th} page is always the first page to be loaded, and any page transitions from that point can be to any other arbitrary page.

PAGE INFLATOR

The page inflator is responsible for the taking of a Page object, creating an Android Fragment object, initialising the gesture and UI layers of the application according to the properties present in the Page object and finally replacing any existing pages currently visible to the user with the new inflated page fragment.

This process can be broken down into a number of different steps carried out by different levels of the application; a diagrammatic representation is shown in Figure 16.

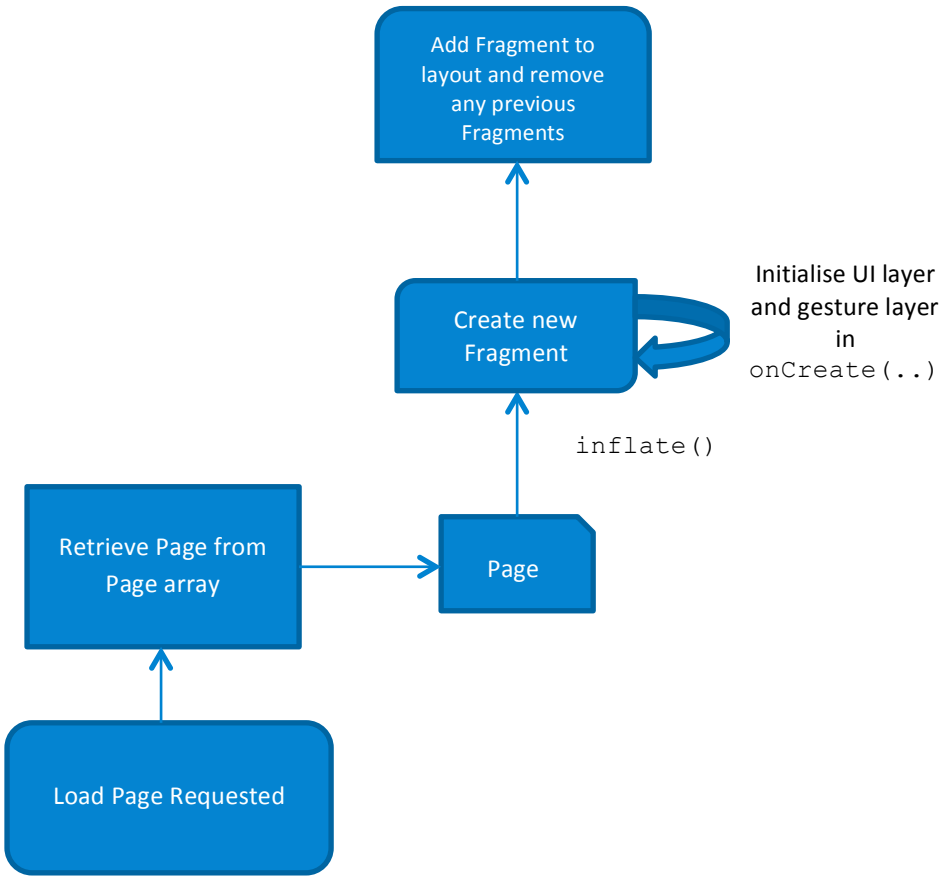


Figure 16: Representation of process which occurs when loading a page to be displayed.

The `inflate()` method of the `Page` class involves starting a `FragmentTransaction`, where Fragments are exchanged, removed and added to the layout, as well as allowing for automatic back-stack handling (when a user clicks the back button the last `FragmentTransaction` will be reversed, removing Fragments which were added, adding Fragments which were removed). During a Fragment's lifecycle the Android system calls various methods at different stages in the lifecycle as defined in the documentation, methods such as `onCreate(…)` and these allow for the initialisation of the gesture and UI layers to take place at the correct time (for example initialising the UI when the Fragment has been added to the layout, and the gesture layer when the UI has been created). This allows for very processor heavy tasks to be completed when needed and not all at once, which prevents system unresponsiveness caused by large initialisation overheads with the OpenCV library.

GESTURE LAYER

The gesture layer encapsulates the opening and closing of the device's camera, the real-time detection of gestures as explained earlier in this chapter and facilitating the correct UI layer calls depending on the outcome and timings of gestures. This is achieved through an inheritance tree (as shown in Figure 17) and different types of pages can extend at different layers of abstraction in order to access different levels of gesture recognition.

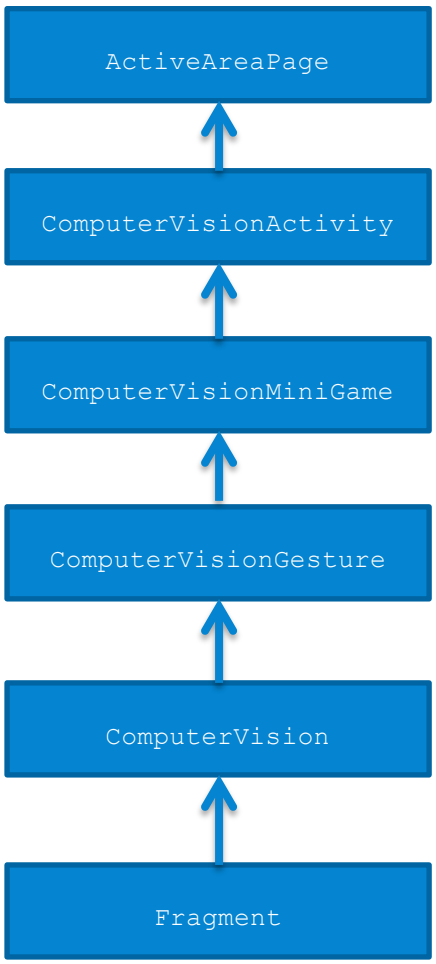


Figure 17: The class structure for an ActiveAreaPage

At the base level is an Android Fragment. Inheriting from this class allows for the addition, removal and replacement of the Page and other features supported at the operating system level. Then the `ComputerVision` class which inherits from Fragment simple initialises OpenCV and ensures that the device's camera is closed and opened at appropriate times, allowing for more complicated gesture support at a more abstracted level. `ComputerVisionGesture` builds upon this by enabling Gesture support for a variety of gestures, using the methods described earlier in this chapter. `ComputerVisionGesture` implements a callback which allows for inheriting classes to simply override the default handler and react appropriately to the progress of gestures and the location of tracking points.

`ComputerVisionMiniGame` and `ComputerVisionActivity` encapsulate the UI layer tasks, and the lower level classes described ensure a degree of abstraction when rendering the storybook.

UI LAYER

The UI layer is responsible for taking the parsed book and building upon gesture recognition to display it in a consistent and scalable manner and to enable user-interactivity within the story. A series of generic interactive pages have been created which can be customised by story developers, utilising multiple gestures, animations and games to make rehabilitation more interesting for the affected child. These generic interactive pages are built upon the abstractions provided by the gesture layer of the application.

One of the core interactive pages is the `ActiveAreaPage`. This page allows the child to select a highlighted area within the scene from a series of possible areas, for example the bin in the kitchen in Figure 18 below, which then starts a game or story scene to continue the story according to the user's choice.

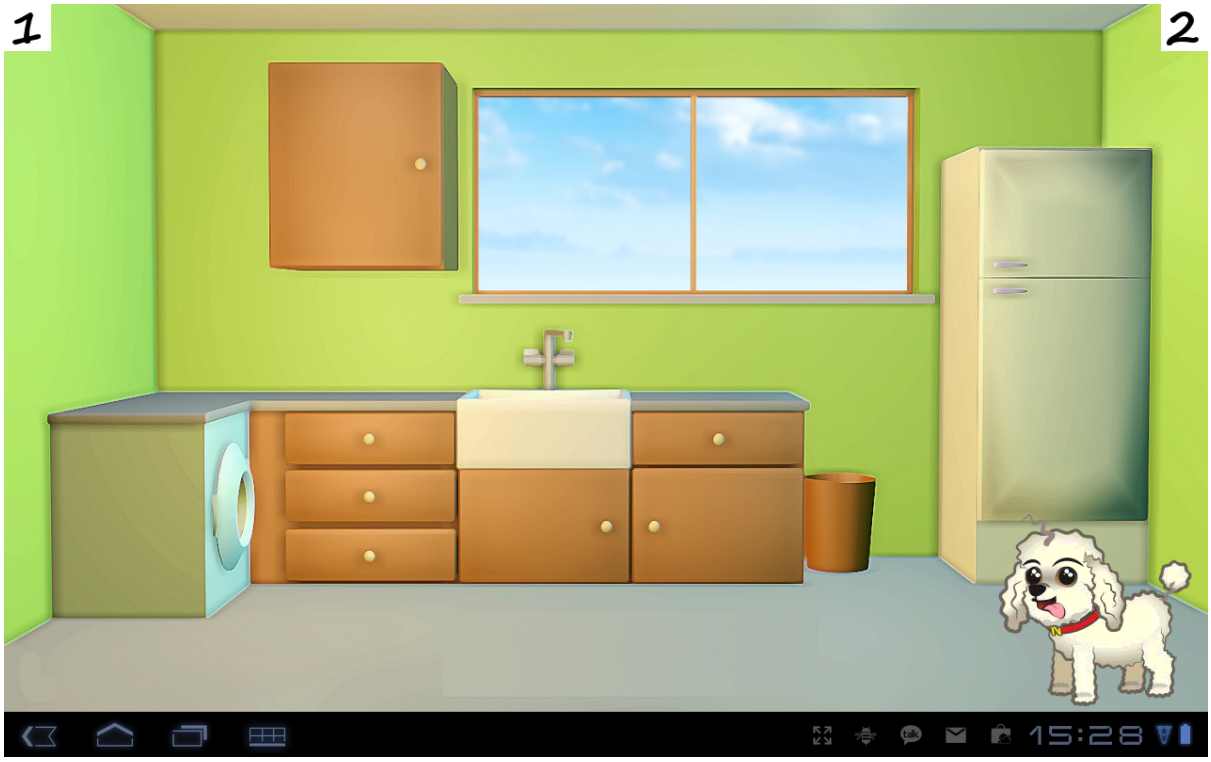


Figure 18: A customised `ActiveAreaPage` with different areas for the user to select.

The `ActiveAreaPage`'s highlighted areas are specified in the JSON "Book" file as described earlier in this chapter, and an example of this is below:

```
"active_areas":[

      {"drawable":"NaughtyNoodle/NaughtyNoodle/Assets/bin3",
"coords":"999,553,78,116,1492,786", "trigger_page":2},

      {"drawable":"NaughtyNoodle/NaughtyNoodle/Assets/sink3",
"coords":"553,461,225,104,1492,786", "trigger_page":3}

      ]
```

Figure 19: Specifying an `ActiveAreaPage`'s highlighted areas.

When the page instance is being loaded the asset URI (eg `"NaughtyNoodle/NaughtyNoodle/Assets/bin3")` is passed back to the global Glue object through a call to its `Glue.requestBitmapResource(String URI)` method, which returns the extracted Bitmap. This Bitmap is then laid out at the specified coordinates and when interacted with an event handler switches the page to whichever page was specified in the `"trigger_page"` key.

This example also demonstrates the ability of the system to scale to different screen sizes and aspect ratios. All coordinates specified in the Book file must be specified in the format `x1, y1, x2, y2, w, h` where `w` and `h` are values which represent the resolution of the screen the coordinates correspond to. In this manner it is possible to take those values of width and height and to calculate the position on the current device's screen at which the asset must be laid out. This allows the system to scale images and to lay out user-interface elements in a manner the designer of the storybook intended, regardless of screen size. This satisfies requirement 2.2.

Like the highlighted areas the same system is used for every aspect of the UI within the engine, from the background of pages, to the placement and content of story-telling and informational text. This allows for every page to be customised in order to fit the vision of anyone creating their own storybook.

SUMMARY

This chapter provided an overview of the design and implementation of the system and how it provides solutions to the requirements outlined at the beginning of this chapter.

Implementation was completed using the iterative development model. A base application was built and then features were added and improved to form the final system. All requirements were satisfied by the final system.

Java, in tandem with the Android API, was used during the implementation of the system, with the OpenCV C++ library for the development of gesture recognition.

In addition to this JSON was used for the file format and the Jackson JSON parsing library was used for the parsing of this file. All file processing and parsing is encapsulated in one step of the program which converts the file into a Page object which is used for the construction of the UI.

Gestures were implemented in OpenCV and used a colour segmentation and detection approach to provide the best performance while still being able to pick up small movements. The gesture code was encapsulated and generic to allow for maximum portability, so it can be used in a variety of different contexts and storybooks without the need for any further modification.

CHAPTER 4: TESTING AND RESULTS

INTRODUCTION

This chapter discusses the methods and results of testing the system and provides analysis of the results obtained from these tests. The overall approach and means of testing is described first, chosen in order to ascertain if current generation Android tablets are capable of running gesture recognition in real-time, followed by discussion of the issues encountered during testing and how they may have affected the results.

SYSTEM ANALYISIS

ENVIRONMENT TESTING

In order to ensure that the system works well on a variety of tablets the final system was tested on a number of other Android tablets to ensure compatibility and acceptable performance. On all the devices listed in TABLE the system worked correctly and either had a similar or higher frame-rate to the development device. As the Motorola Xoom was chosen as the development device, and considering it is a first generation device this is not surprising; all newer 10 inch tablets will have newer, more powerful processors which should increase the performance of the system.

Tablets tested
Motorola Xoom
Samsung Galaxy Tab 10.1
Asus EeePad Transformer
Asus EeePad Transformer Prime

Table 4: Tablets tested in environment testing

Testing was carried out by installing the final system onto each device and to run it and observe the performance and that no unexpected crashes occurred. Due to the flexibility of the Android SDK and the fact that all Android applications run in a Java virtual machine the application functioned as expected on all tablets, and in the case of the Asus EeePad Transformer Prime the performance of the system was significantly better, because of the improved processor.

VISUAL QUALITY OF THE SYSTEM

An important measure of the success of the system's design is how visually attractive and engaging it is for a child, and in order to test how flexible the system's engine is in facilitating the creation of stories which match the visual fidelity of a physical story-book a reference story was created, entitled "Naughty Noodle".

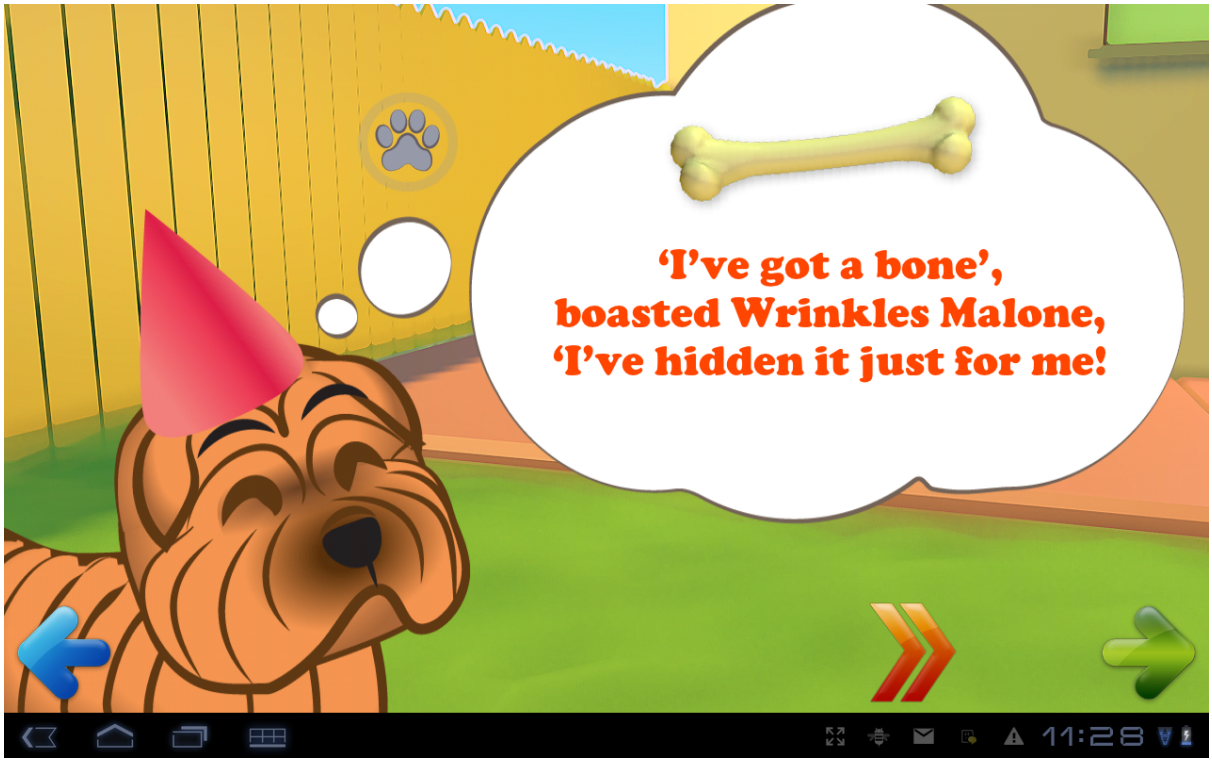


Figure 20: Storytelling segment

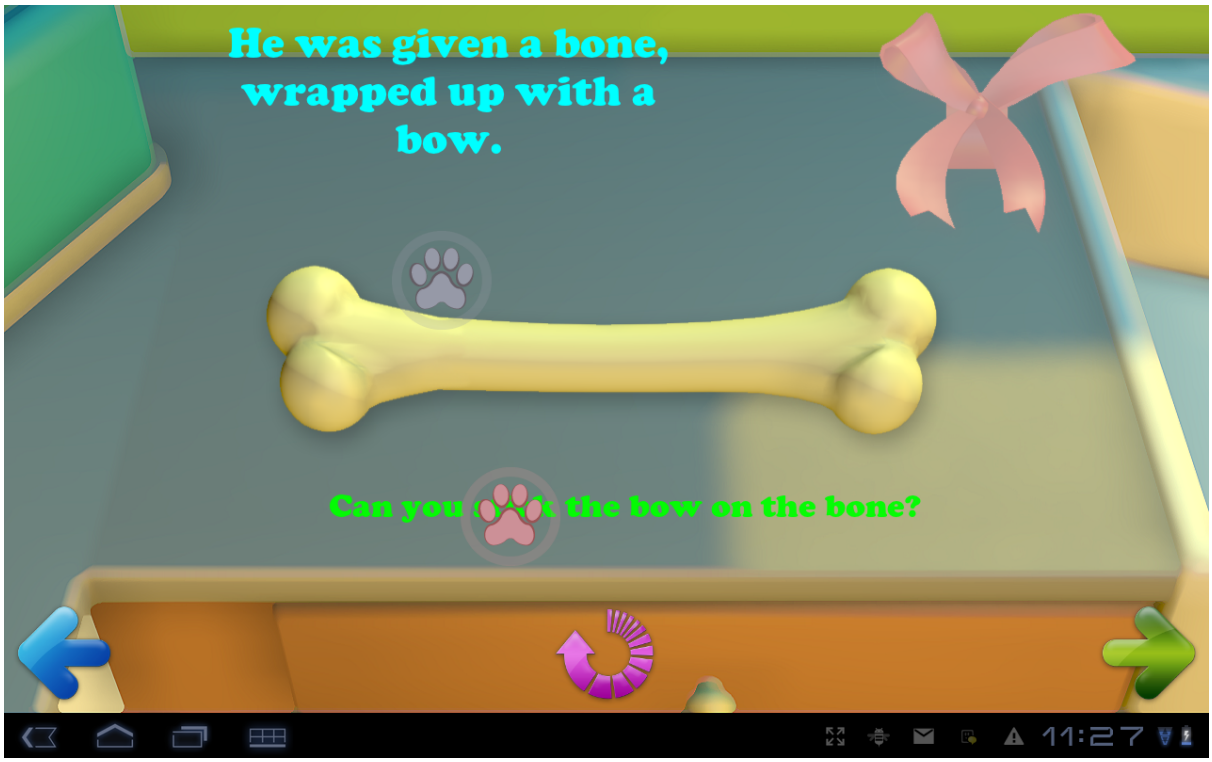


Figure 21: Interactive gesture detection page, involving grasping the bow and "pinning" it to the bone

Figure 22: Another example of a gesture page, involving inflexion to "lift" the cat-flap.

These screenshots demonstrate the high visual quality of storybooks that the system is able to produce. Through the use of fragments and the generic engine created elements can be reused and specified in a textual format, and persistent elements of the story (such as the soundtrack) can be persisted across different pages without complicated application logic to pause, transfer, and resume such elements.

PERFORMANCE

Gesture detection of the system must be of sufficient granularity to be able to accurately detect a gesture in order to be useful in the rehabilitative process. However, depending on the approach taken and the amount of detail that is able to be extracted there is a hit to the performance of the system.

Gesture Detection Method	Accuracy	Frames per second (Active)	Frames per second (Idle)
Colour segmentation	Average	15.2	18.8
Hand tracking (Skin detection) – Simple	Poor	14.4	19.1
Hand tracking (Skin detection) – Complex	Very good	0.6	8.4
Movement tracking (Simple) – Bounding rectangle	Very poor	7.1	8.7
Movement tracking (Simple) - Contours	Poor	4.9	8.7
Movement tracking (Optical flow)	Good	2.5	7.3

Table 5 illustrates the performance of the system using a variety of different gestures.

Tests were written using the same OpenCV library and were run on a tablet in a graphics free application window in order to eliminate the possibility of graphics rendering causing interference with the tests. The frame rate was measured by timing the time between subsequent `onPreviewFrame()` calls, which are called whenever the system is able to accept a new camera frame for processing, and is a measure of how quickly gesture processing takes place on the device.

Tests were done in two phases, an “idle” phase where no moving input is given to the camera, simply the same static video feed, and an “active” phase where a hand/colour/entity which would be recognised by the algorithm is shown moving at a relatively constant rate across the video feed. The frame-rate is calculated by taking a series of samples and averaging them. Comparing the “idle” phase and “active” phase allows for a distinction to be made between the algorithm’s base cost where no tracking points can be detected and the overheads from processing when tracking points can be detected.

Gesture Detection Method	Accuracy	Frames per second (Active)	Frames per second (Idle)
Colour segmentation	Average	15.2	18.8
Hand tracking (Skin detection) – Simple	Poor	14.4	19.1
Hand tracking (Skin detection) – Complex	Very good	0.6	8.4
Movement tracking (Simple) – Bounding rectangle	Very poor	7.1	8.7
Movement tracking (Simple) - Contours	Poor	4.9	8.7
Movement tracking (Optical flow)	Good	2.5	7.3

Table 5: Relative performance of different gesture detection approaches

Explanation of terms:

Colour segmentation: Colour segmentation involves taking a camera still, considering every pixel in the image and seeing if it falls within two colour scalars, if it does then the pixel is kept (and replaced with a white pixel), otherwise it is discarded (replaced with black). This processed image can then have contours extracted from it and gestures built on top of the location of a suitable contour.

Hand tracking (Skin detection) – Simple: Very similar to the colour segmentation approach, however the scalars used represent a range of skin colours, and the resulting black and white image is processed for the largest contour, which represents the hand. This is very inflexible for the purposes of gesture detection, especially when trying to deal with individual fingers, which does not provide enough granularity to work with.

Hand tracking (Skin detection) – Complex: Same as simple hand tracking, but the resulting image is processed using a variety of algorithms to extract individual details of the fingers, and a model of a hand can be built up and inferences made from this.

Movement tracking (Simple) – Compare a received frame with the last received frame, extract any pixels that are different within some tolerance, assume the shape which has moved to be the hand of the individual and take the average position of the shape to be the general location of the hand. This can be used to infer gestures from.

Movement tracking (Optical flow) – Optical flow is the pattern of observed motion of objects, surfaces and other features in a scene caused by the relative movement between the viewer and the scene. (Burton & Radford, 1978). Through the use of an optical flow algorithm (as built into OpenCV) the motion of objects in a scene can be calculated, and gestures can be calculated from this data. Optical flow algorithms however are extremely expensive to calculate on a low power device in real-time.

Colour segmentation provided the best performance allowing a reasonable level of detail to be extracted. As a result, this method was used for all gesture recognition implementation.

Some issues encountered which were hardware specific involved the relatively poor quality of a lot of tablet cameras, leading to a washed out or low contrast picture which made picking colour features very difficult. This issue was impacted by the necessity to downscale video stills before processing them in order to increase performance; this often led to a small field of view for the application, moving the coloured finger too far from

the camera could result in the gesture tracking losing its ability to track the user. Given that the method of gesture detection chosen was colour segmentation, this could, at times, be a particularly difficult issue to overcome. An initial calibration step involving selecting colours from the camera feed helped mitigate this issue, but any large changes in the environment still posed a problem.

GESTURE FIDELITY

The fidelity of the gesture recognition is an extremely important factor in detecting a hemiplegic user's gestures, as severely affected users may have only limited hand and finger mobility, and a less affected child may have significantly better mobility in their hand. As a result gestures must be capable of determining small and larger gestures and accepting both as equally valid.

In order to test the range of gestures the system is capable of detecting the inflexion gesture was carried out with large, medium and small movements at the same distance from the camera in the same lighting conditions. The inflexion gesture is tested as this is the most common point of troubles for children with hemiplegia, and if not exercised the child's hand can contract from not being used and become ineffective at many tasks.

Table 6 shows the results of the test.

Gesture fidelity (size of movement)	Results (successfully recognised gestures out of 10 attempts)	Notes
Small (<1cm)	7/10	The algorithm would occasionally interpret small incidental movements as a gesture and would occasionally fail to detect gestures.
Medium (3-6cm)	9/10	
Large (>6cm)	10/10	Worked flawlessly and did not interpret incidental movements as being a gesture.

Table 6: Table showing the results of testing gesture fidelity.

Attempting to detect very small movements was the most error prone and small incidental movements could be detected as being a gesture, even when incorrect. However the simple tweaks to the algorithm in order to accept smaller gestures was crude and with some modification could produce better results. This demonstrates however that the gesture detection algorithms used in the system are capable of detecting small gestures, even if they are not currently optimised to do so.

An issue encountered regarding the fidelity of detecting gestures on an Android tablet was the small field of view of the front-facing cameras on current-generation tablets. The cameras are designed for video-calling and as a result have a field of view designed to encapsulate the face and not the larger space that would be suited to gesture detection, where an individual's hand may not be centred in front of the device. This is unfortunately an issue which cannot be mitigated in any form, and giving adequate feedback to the user when their hands stray outside of the detectable area is the only solution in order to prevent the user getting confused and later frustrated.

RESULTS

FINAL SYSTEM PERFORMANCE

In order to check that the system runs at an acceptable frame-rate the final system was tested under a variety of different scenarios to test how responsive and accurate gesture detection using colour segmentation is while rendering animations and performing general game logic while the user interacts with the story. The results are shown in Table 7 and all tests were carried out on the "Naughty Noodle" reference story.

Type of Page	Frames per second	Notes
Pinch/Grasp and place bow on bone	15.6	Scene included a static background, several dynamic image elements and a view with a video providing instructions for the user/live feed from the camera.
Flexion/Inflexion to open the catflap.	16.3	Scene included a static background, several dynamic image elements and a view with a video providing instructions for the user/live feed from the camera.
"Scurry" to empty bin of rubbish	7.9	Scene included a static background, several dynamic image elements and a view with a video providing instructions for the user/live feed from the camera. In addition there were three or four animations triggered every second when the scurry gesture was detected.

Table 7: Table showing the performance of several common gestures.

RESULTS ANALYSIS

The performance results show that current generation Android tablets are capable of running gesture detection code through the OpenCV library, both as a standalone process and also with additional graphics and interactive elements.

While an average frame rate in the mid-teens isn't so responsive that an individual would view it as instantaneous (a good response rate for any visual process is between 30 and 60 frames per second), the gesture detection algorithms used in the final system feel responsive and do not suffer from a "lag" where users do not feel in control because the system is taking too long processing past events and are not processing current ones. Most touchscreens have a touch response time of around 100ms (Microsoft Research, 2012), and the final system running on an Android tablet has a response time of around 70ms, slightly better than an average tablet's screen touch response.

The system can become noticeably slower when multiple animations and other processing-heavy visual elements are active however. During the "scurry" bin demo the system responds at around half the idle speed, which is clearly noticeable to an end user. This, however is the exception to the norm during normal execution of the system, and scenes with interactive elements provide little slowdown. This is due to Android's user interface rendering system, which only re-renders elements and parts of the screen which have been changed, something which is very frequent during a tweened animation. During normal execution the rendering of most other interactive elements is directly linked to updates from the gesture detection code, the re-rendering

tends to be in step and not continuous such as with the “scurry” animations, leading to no great decrease in performance.

SUMMARY

Two main types of testing were used to gather a view of the overall system:

- Environment testing was carried out to ensure that the system ran successfully on a variety of different tablets with different processors and graphics chips.
- Performance testing was used to provide evidence that the gesture detection algorithms running on the device were capable of providing a responsive experience to the user and could be used to track an individual’s gestures accurately.

CHAPTER 5: EVALUATION AND CONCLUSION

INTRODUCTION

This chapter evaluates the results and findings of this dissertation, giving a breakdown of how successful each step of the process was and what could be improved. The evaluation of the dissertation will focus on all aspects of the project at a high level and a final dissertation-wide evaluation will conclude the chapter along with a discussion of further areas of work that can be researched in the subject area.

CONCLUSION

SATISFACTION OF HYPOTHESIS AND OBJECTIVES

The hypothesis of the project can be evaluated by first looking to what extent the objectives of the project were satisfied.

Objective One: Investigate current techniques involved in the rehabilitation of children with hemiplegia.

This objective was satisfied as laid out in Chapter 2, which outlines that the majority of current rehabilitative approaches involve physiotherapy in a context-free environment and exercises are often designed to mimic real-world tasks and actions which would be essential to any person capable of providing their own care. It was also outlined using previous studies how effective rehabilitation is for the long term recovery and quality of life for individuals affected with hemiplegia.

Objective Two: Identify areas in which traditional efforts are lacking or incapable of reaching.

This objective was also satisfied in Chapter 2; it was determined that traditional methods of rehabilitation for hemiplegia lack in areas of engagement and context for the child, and for young children require supervision.

Objective Three: Build an Android Tablet application involving gestures, colours and leveraging the advantages of a digital medium to help the children with their exercises while being enjoyable.

This objective's success can be measured in terms of how many of the system requirements were fulfilled during the implementation stage and were present in the final system:

- **The system must support a variety of gestures**. – The final system supports a number of different gestures exercising a number of different hand muscles and common actions that a child would need to carry out successfully in their lives.
    - **Gesture recognition must be real-time and responsive.** – As outlined in Chapter 4, the final system's gesture recognition algorithms run at a real-time pace and are reasonably responsive to an end user.
    - **Gestures must reflect a number of different clinical exercises**. – Gestures implemented by the system were implemented based on the advice of a child health professional with a large amount of experience working with children with hemiplegia. The gestures are directly applicable to real world actions and should help the child's overall rehabilitation.
    - **Gestures must be accurate and have a low number of false positives**. – The colour segmentation gesture detection approach chosen for usage in the final system ensures that

gesture recognition can be run in real time and has a reasonable level of accuracy. Issues with lighting as outlined in Chapter 4 mean that accuracy can, at times suffer, but there are safeguards built into the system to cope with that. The context of the gesture recognition, in a game format, also allows for a greater tolerance of errors; a single misread of a gesture in the context of many identical gestures being performed does not especially matter. However as the power of devices increases there is more scope for switching to a more accurate but computationally expensive manner of detecting gestures.

- **The system must have a generic engine which can support a variety of different stories and graphics.** – The final system has a framework in place for a generic system to load different stories loaded from a file. In Chapter 4 the visual fidelity of the system running on a tablet is described and demonstrated. However while the reference story is demonstrated, there was no opportunity to test the effectiveness of designing stories in an external context and to get feedback with that.
    - **The system should load these stories from a file-format.** –In Chapter 3 implementation details around this feature is laid out, and the final system contains support for loading stories from a file format and displaying them on the screen.
        - **Stories stored in this file format must be scanned for on the storage of the device and displayed in an easy-to-browse format.** –This feature is present in the final system, files with the ".nbk" file extension are scanned for in the storage of the device and are displayed in an Amazon Kindle-like manner for users to select from. The system also supports displaying a book cover and other metadata for easy perusal.
    - **The system must scale to different tablet screen sizes.** –While the system's design supports multiple screen sizes as outlined in Chapter 3, the final system was not tested on tablets or phones with smaller screen sizes, due to time constraints during development. In a worst case this could result in unintended and untested consequences, however the architecture of the system allows for such issues to be easily fixed in future.
    - **These stories must be interactive.** –The stories have interactive elements built in as outlined in Chapter 3. The system has support for a number of built in "types" of interactive pages, including flinging and switching the scene in response to a gesture being performed.
        - **The system must include a number of different generic pages from which interactivity can be derived, with a number of tasks and games for the user to play.** –The system has a number of built in pages which can be used by book creators to add interactive and generic user-interface elements to their stories as described in Chapter 3.

Objective Four: Compare the traditional approach with the interactive Android based system and evaluate issues and its success.

This chapter will attempt to satisfy this objective.

SUMMARY

Current techniques involved in treating children with hemiplegia were first examined, and the system was designed with these rehabilitative techniques and their associated limitations in mind, in order to provide a system which is designed to help supplement the current clinical process. The requirements then created for the development of this system were implemented, and most were successfully implemented, with a few exceptions due to time constraints. The background research and subsequent implementation were successful for exploring the aims of the project and the final system built is a functional proof of concept.

SATISFACTION OF THE HYPOTHESIS

Hypothesis: Android based tablet devices are suitable and effective at the rehabilitation of children with neurological disorders (specifically hemiplegia).

Chapter 4 examines the performance of the final system and the conclusion which can be taken from these results shows that Android tablets have enough processing power and RAM to be viable for gesture detection from a performance standpoint, and that as the power of these devices increases more accurate and advanced methods of gesture detection will become feasible for usage.

From a rehabilitative standpoint the gesture system was required to pick up movements both big and small from a child with hemiplegia. Some children may only be capable of very small movements in their hands, and others may be less severely affected and capable of larger movements. It is clear that some form of calibration step would be needed in the application to account for this. As described in Chapter 4, the system can be modified to detect small movements, but due to time constraints a user interface was not built around this feature, and the lack of significant work due to timings put into this feature means that there were a number of false positives and missed gestures as a result.

The hardware of Android tablets could also cause problems, the quality of images produced by the front-facing camera made detecting colours and as a consequence gestures more difficult, but with adequate calibration most of these issues can be mitigated as outlined in Chapter 4.

CONCLUSION

It can be concluded from these observations that Android tablets are effective at the rehabilitation of children with neurological disorders in a way that was not possible before; they can run the expensive gesture detection algorithms and detect with limited modification detect very small gestures in real time. Due to ethical constraints the system was not personally tested on children with hemiplegia, but feedback from Professor Janet Eyre in the department of Child Health was very positive. The hypothesis is satisfied in the following ways:

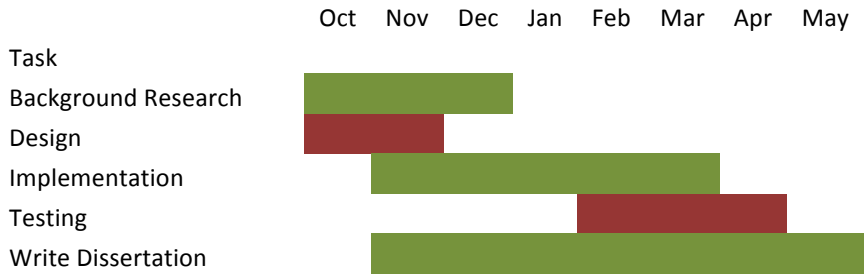
- Current generation Android tablets are powerful enough to run certain forms of gesture detection in real-time.
- The gesture detection code can be adapted to detect gestures of different fidelities, including very small movements.
- The system has an attractive user interface which can load a variety of different stories and interactive elements for the user to use.

The conclusion could be strengthened by conducting a study over a period of time with children affected with hemiplegia and determining the relative rehabilitative benefits of the system to them, including how engaging they find it compared to their traditional rehabilitation and therapy.

EVALUATION

TIME PLAN

The time plan that was originally set out for the development of the system is shown in FIGURE.



This time plan largely reflected the actual development of the system, with the notable exception of the testing stage of the system development, which was a more incremental process and happened across the development of the system from the very beginning until after the development had finished. This can be attributed to the development methodology chosen, the agile approach, which encourages quick iteration and addition of features, incorporating testing at every stage. This development methodology worked well for the project; because there were other people working on the same codebase during development of the generic storybook engine as outlined in Chapter 3 it allowed for a quick iteration over the codebase and changes were incorporated quickly and efficiently.

WHAT HAS BEEN LEARNT

A deeper understanding of Java, the Android SDK and developing for mobile devices has been attained by the planning, development and evaluation of the system outlined in this dissertation. Dealing with devices with a small amount of RAM and relatively weak processors was an interesting experience, and required making trade-offs and considering the implications of decisions made early on in the planning process and how they might impact development at a later stage. An understanding of C++ and the Java Native Interface was also gained from developing with OpenCV, and the issues around running native code through the Java Virtual Machine involved much thought when combined with the hardware limitations of the Android Tablet platform, as the Java Native Interface can demand high memory overheads and translating calls and results from Java bytecode into native calls to the OpenCV library could be a slow process.

A richer understanding of development methodologies, including the agile approach and the advantages of using such a methodology when working in a team were also gained from participating in this project. An understanding of issues which can arise from poorly designed or certain poor implementation strategies was gained, and was worked through and mitigated.

In addition researching and referencing studies and examining background research helped to broaden the author's knowledge of hemiplegia and other neurological disorders, including methods of rehabilitation and other areas where individuals are turning to computing devices in order to help supplement or replace the traditional methods of treating individuals and children who are affected with hemiplegia and other disorders which can affect mobility.

ASPECTS OF THE PROJECT THAT WENT WELL

The implementation strategy chosen for the development of the system was well chosen, an agile approach suited the way in which people would be working on the project and it helped avoid complications by allowing for the modification of the system frequently. Choosing colour segmentation as the type of gesture detection chosen on the project was also a good choice and led to implementation being easier and performance being less of an issue than it would have otherwise been if a more computationally expensive gesture detection algorithm was chosen. The colour segmentation also was suited to the problem at hand well, all gestures can be expressed in terms of changes in the position of the thumb, index and little finger and further hand position data would only be especially useful in the case of very small gestures.

The project timings were largely planned well; as described above most requirements for the final system were met and those that weren't met were partially implemented and not fully tested.

ASPECTS OF THE PROJECT THAT COULD BE IMPROVED

If the project were to be repeated the requirement of the system being able to run on tablets of different and varying screen size would have been tested from the beginning and not left until the point where getting access to different sized devices was impossible within the time constraints.

The system initially was designed in order to create a small, proof of concept storybook with gesture detection aspects and interactivity, and due to initial time constraints many assumptions and aspects of the system architecture were hard-coded into the prototype. This hard-coding caused problems when retroactively modifying the system in order to use the generic storybook engine approach, and as a result it took a lot longer to switch the system over than it would have if the correct assumptions could have been made from the beginning.

The conclusion of the project is not based upon any user studies, and if the project was to be repeated having access to children with hemiplegia would have been crucial at nearly every stage of development, especially the initial gesture prototype. Having a user's input and being able to see what aspects of the system were confusing, difficult to use or otherwise not clear to a young child would have been invaluable and could have saved some re-engineering when aspects of the system had to be tweaked to better fit interactions of this nature.

FURTHER WORK

During the research stage of this project it became obvious that there are many possible avenues of future research which could be explored given enough time and resources, and multiple ways of modifying and building upon the system designed during this project in order to accomplish different tasks. Future work in this area includes looking at the viability of current generation Android phones to run similar gesture detection code. Android tablets often have more powerful processors and more RAM than their phone counterparts, due to the availability of more physical space inside the device's enclosure for larger parts, and as a consequence the processing abilities of these Android phones would be an interesting avenue of research and future work.

A further area for work would be to investigate how effective the system built during this dissertation is at supplementing the rehabilitative process. While this is far outside the scope of this dissertation, conducting a study with several similar children with hemiplegia, half following the standard traditional approaches to their physiotherapy, and the other half supplementing their rehabilitation with the system built as a part of this project and examining which proves to be most effective. This type of experiment would be vital in

understanding the true impact of the system and if it truly achieves the purpose for which it was designed. Likewise conducting user-testing to ensure that the user interface is understandable to a small child and their parents and that navigation and basic application usability is satisfactory is important when considering changes to the application's user-interface layer.

While the development of the system incorporated a reusable system for creating interactive storybooks there is no easy way for a layman with limited technical knowledge to create a new story for the system. In order to do so a simple desktop application should be designed and built with a drag-and-drop what you see is what you get (WYSIWYG) editor for creating stories. This would allow individuals with to create books without the need for any kind of knowledge of JSON or the underlying system and method of reading and parsing storybook files. This will promote further storybooks to be created for the system and will extend the usability for the end user accordingly. Java or another traditional programming language could be used for this purpose.

SUMMARY

The design and implementation was a success, with the majority of the objectives for the system having been met and the hypothesis for the system being partially supported by the final system, that Android tablets are capable from a technological standpoint at running gesture detection code. All objectives were met and the majority of the design requirements were met, except for ensuring that the system works on devices with different sized screens, which was partially completed due to time constraints.

There are several areas of future work identified both on the system developed during this project and in the area of rehabilitating children using interactive computer systems, including testing the system in a trial to identify the real-world benefits of using the system during the rehabilitative process and modifying the system and adding tools to the system to allow for non-technical individuals to produce storybooks easily.

This project has opened up many possible questions about the efficacy of using tablet devices and touchless interfaces in a clinical context; the ability of handheld devices to run these complex computer vision algorithms in real time would allow for many new areas of work and research.

BIBLIOGRAPHY

Adamek, M. (2011, February 4). *JSON Parsers Performance on Android (With Warmup and Multiple Iterations)*. Retrieved January 27, 2012, from Martin Adamek: http://martinadamek.com/2011/02/04/json-parsers-performance-on-android-with-warmup-and-multiple-iterations/

Bach-Y-Rita, P. (1990). Brain Plasticity as a Basis for Recovery of Function in Humans. *Neuropsychologia, 28*(6), 547-554.

Burton, A., & Radford, J. (1978). *Thinking in Perspective: Critical Essays in the Study of Thought Processes.* Routledge.

comScore. (2010). *Results of Study on Apple iPad.* comScore.

Cooper, T. (n.d.). Face Detection for Self-Portrait Assistance in Android. *Dept. of Electrical and Computer Engineering*.

davidalln. (2010, May 26). *COlor Tracking with openFrameworks/OpenCV.* Retrieved December 12, 2011, from YouTube: http://www.youtube.com/watch?v=xDh4CMZVzvg

de Jong, L. D., Nieuwboer, A., & Aufdemkampe, G. (2006). Contracture preventive positioning of the hemiplegic arm in subacute stroke patients: a pilot randomized controlled trial. *Clin Rehabil, 20*, 656-667.

Denman, S., Chandran, V., & Sridharan, S. (2007). An adaptive optical flow technique for person tracking systems. *Pattern Recognition Letters, 28*, 1232-1239.

Frampton, I., Yude, C., & Goodman, R. (1998). The prevalence and correlates of specific learning difficulties in a representative sample of children with hemiplegia. *British Journal of Educational Psychology, 68*, 39-51.

Google. (2011, November 12). *Support Package*. Retrieved January 27, 2012, from Android Developers: http://developer.android.com/sdk/compatibility-library.html

Google. (2012, January 26). *Android API Levels*. Retrieved January 27, 2012, from Android Developers: http://developer.android.com/guide/appendix/api-levels.html

Google. (2012). *Android NDK Overview*. Retrieved May 1, 2012, from Android Developers: http://developer.android.com/sdk/ndk/overview.html

Google. (2012, May 1). *Designing for Performance*. Retrieved May 2, 2012, from Android Developers: http://developer.android.com/guide/practices/design/performance.html#native_methods

Hackborn, D. (2011, February 3). *The Android 3.0 Fragments API.* Retrieved December 30, 2011, from Android Developers Blog: http://android-developers.blogspot.com/2011/02/android-30-fragments-api.html

HemiHelp. (n.d.). *What is hemiplegia?* Retrieved 04 24, 2012, from HemiHelp: http://www.hemihelp.org.uk/hemiplegia/what_is_hemiplegia

Kajaani University of Applied Sciences. (2011, August 4). *Stroke Rehabilitation Game using Kinect.* Retrieved December 1, 2011, from YouTube: http://www.youtube.com/watch?v=JnObHk7iqtg

Lefever-Davis, S., & Pearman, C. (2005, Feb). Early readers and electronic texts: CD-ROM storybook features that influence reading behaviors. *The Reading Teacher, 58*(5), 446-9.

Manresa, C., Varona, J., Mas, R., & Perales, F. J. (2000). Real-Time Hand Tracking and Gesture Recognition for Human-Computer Interaction. *Electronic Letters on Computer Vision and Image Analysis, 0*(0), 1-7.

Microsoft Research. (2012, March 5). *Applied Sciences Group: High-Performance Touch*. Retrieved 2 10, 2012, from Microsoft Research: http://research.microsoft.com/apps/video/dl.aspx?id=160670

Millard, J. B. (1966). Rehabilitation and the Hemiplegic Patient. *Rheumatology, 8*(7), 244-249.

Mol, S. E., Bus, A. G., & de Jong, M. T. (2009, June). Interactive Book Reading in Early Education: A Tool to Stimulate Print Knowledge as Well as Oral Language. *Review of Educational Research, 79*(2), 979-1007.

Munoz-Salinas, R., Aguirre, E., & Garcıa-Silvente, M. (2007). People detection and tracking using stereo vision and color. *Image and Vision Computing, 25*, 995-1007.

Nielsen. (2011). *Third quarter survey of mobile users.* Nielsen.

NISO. (2004). *Understanding Metadata.* Bethesda, MD: NISO Press.

Schaechter, J. D. (2004). Motor rehabilitation and brain plasticity after hemiparetic stroke. *Progress in Neurobiology, 73*, 61-72.

Snels, I. A., Dekker, J. H., van der Lee, J. H., Lankhorst, G. J., Beckerman, H. P., & Bouter, L. M. (2002). Treating Patients with Hemiplegic Shoulder Pain. *American Journal of Physical Medicine & Rehabilitation, 81*(2), 150-160.

Toytman, I., & Thambidurai, J. (n.d.). *Banknote recognition on Android platform.* Stanford University.

Willow Garage. (2011, August 24). *OpenCV.* Retrieved December 5, 2011, from OpenCVWiki: http://opencv.willowgarage.com/wiki/Welcome

Yu, F. (2010). *Real Time Augmented Reality on Mobile Phone.* Retrieved 12 14, 2011, from Electrical Engineering and Computer Science, University of Michigan: http://www.eecs.umich.edu/~silvio/teaching/EECS598_2010/final_report/Fisher.pdf

TABLE OF FIGURES

Figure 1: Project timescale and work order Gantt Chart. 7

Figure 2: Using colour tracking to “write” a word by tracking the path of a piece of card. (davidalln, 2010) 10

Figure 3: An example of Canny Edge detection on an Android device. 10

Figure 4: Using a Kinect device with an XBOX 360 to navigate a user interface without any traditional physical input. 11

Figure 5: High level overview of the system 15

Figure 6: Some example JSON 17

Figure 7: Standard Android user interface elements. 19

Figure 8: Android UI highlighted with red and blue showing different user interface Fragments 20

Figure 9: Colour segmentation of an image to extract the red component. (picture from http://blog.damiles.com/wp-content/uploads//2010/01/captura.jpg) 22

Figure 10: Amazon's Kindle software running on an Android tablet (picture from http://www.everythingabouttablets.net/wp-content/uploads/2011/04/wpid-P20110422160155.png) 23

Figure 11: An early prototype of the book selection screen. 23

Figure 12: Diagram of the book format 24

Figure 13: An example book file with one "`MainMenuPage`". 25

Figure 14: An example Metadata file 25

Figure 15: Diagram showing a high level overview of parsing pages. 26

Figure 16: Representation of process which occurs when loading a page to be displayed. 27

Figure 17: The class structure for an ActiveAreaPage 28

Figure 18: A customised `ActiveAreaPage` with different areas for the user to select. 29

Figure 19: Specifying an `ActiveAreaPage`'s highlighted areas. 29

Figure 20: Storytelling segment 33

Figure 21: Interactive gesture detection page, involving grasping the bow and "pinning" it to the bone 33

Figure 22: Another example of a gesture page, involving inflexion to "lift" the cat-flap. 34

Table 1: Table comparing advantages and disadvantages of OpenCV and JavaCV 17

Table 2: Table showing the difference in performance between JavaCV and OpenCV 17

Table 3: Table comparing current generation Android tablets 18

Table 4: Tablets tested in environment testing 32

Table 5: Relative performance of different gesture detection approaches 35

Table 6: Table showing the results of testing gesture fidelity. 36

Table 7: Table showing the performance of several common gestures. 37

APPENDICES

SYSTEM SOURCE CODE

Source code and compiled .apk (Android package) of the final system are present in the attached supplementary material.

TEST OUTPUT

Code for calculating framerate:

The updateFramerate() function was called at the start of the gesture detection code's update method for a period of 60 seconds in all cases, before being stopped and the result being printed.

```
long lastTime;
long totalDiff;
int numFrames;

public void updateFramerate()
{
	totalDiff+=System.currentTimeMillis() - lastTime;
	numFrames++;
	lastTime = System.currentTimeMillis();
}

public float getAverageFramerate()
{
	return ((float)totalDiff)/((float)numFrames);
}
```

OUTPUT

Type of Page	Frames per second	Output
Pinch/Grasp and place bow on bone	15.6	Starting framerate tests: Finished framerate tests Average framerate: 15.6fps
Flexion/Inflexion to open the catflap.	16.3	Starting framerate tests: Finished framerate tests Average framerate: 16.3fps
"Scurry" to empty bin of rubbish	7.9	Starting framerate tests: Finished framerate tests Average framerate: 7.9fps

Gesture Detection Method	Output (Active)	Output (Idle)
Colour segmentation	Starting framerate tests: Finished framerate tests Average framerate: 15.2fps	Starting framerate tests: Finished framerate tests Average framerate:18.8 fps
Hand tracking (Skin detection) – Simple	Starting framerate tests: Finished framerate tests Average framerate: 14.4fps	Starting framerate tests: Finished framerate tests Average framerate: 19.1fps
Hand tracking (Skin detection) – Complex	Starting framerate tests: Finished framerate tests	Starting framerate tests: Finished framerate tests

	Average framerate: 0.6fps	Average framerate: 8.4fps
Movement tracking (Simple) – Bounding rectangle	Starting framerate tests: Finished framerate tests Average framerate: 7.1fps	Starting framerate tests: Finished framerate tests Average framerate: 8.7fps
Movement tracking (Simple) - Contours	Starting framerate tests: Finished framerate tests Average framerate: 4.9fps	Starting framerate tests: Finished framerate tests Average framerate: 8.7fps
Movement tracking (Optical flow)	Starting framerate tests: Finished framerate tests Average framerate: 2.5fps	Starting framerate tests: Finished framerate tests Average framerate: 7.3fps