Research Report

Name: Rose Nichols

Student Number: 09019365

Module: Digital Media Project

Supervisor: Michaela Palmer

Project Title: Interactive Storytelling iPad Application

Introduction

Project Aim

I aim to produce an interactive storytelling iPad application for children aged 3-5. I have chosen this project to learn more about mobile development.

Research Questions

1. What are the inherent features and characteristics of traditional and interactive storytelling?

2. Who are the competitors to my application and how can I improve upon the experience created by their applications?

3. What methods are available for producing native iPad apps, and which is the most appropriate for my ability?

4. How are mobile features such as touch events and animation implemented?

Traditional vs Interactive Storytelling

Traditional Storytelling

Traditional storytelling has followed largely the same model for thousands of years.

Following Aristotle's analysis of plot, in which he stated that for a story to be whole it must have a beginning, middle and end, Gustav Freytag in the 19th Century considered that a plot could be broken up into five parts. At the beginning, all of the characters are introduced, next, the conflict is introduced, followed by the climax of the story, the antagonist appearing to have the upper hand, and finally the final confrontation between hero and antagonist, where one prevails.

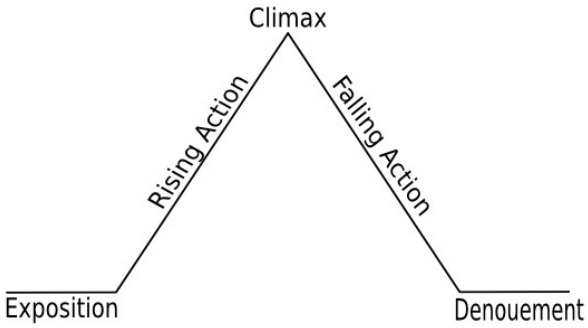

Fig 1.1 Freytag's Triangle

This somewhat simplistic structure has been since developed in the light of further understanding. In the 1940's, Joseph Campbell wrote a book entitled, 'The Hero with a Thousand Faces,' in which he identified the underlying patterns in myth, stories and tradition, and the underlying principles in the structure of stories. Christopher Vogler then expanded on Campbell's work in his book, 'The Writer's Journey: Mythic Structure for Writers,' developing a framework for plot and character development in stories.

Children's stories follow this same model, with the addition that simplicity is especially valued. For example, good characters in a story wear white, bad black.

Interactive Storytelling

Compared to traditional storytelling, the interactive storytelling genre is still relatively new. There are many definitions to be found for 'interactive storytelling,' however some are outdated. I found Chris Crawford's book, 'Chris Crawford on Interactive Storytelling,' to be the most useful both in terms of background research, and principles to carry with me into the actual development of my application. I learnt many things about the genre.

Firstly, two things an interactive story is not: a game with story tacked on, or a story with interactivity tacked on. According to Crawford, having slammed a definition from the book 'Pause and Effect: The Art of Interactive Narrative,' as being part of a school whose 'long-winded phrasing and polysyllabic terminology that pretends to erudition through obscurity,' gives his definition of

interactivity as, ‘a cyclic process between two or more active agents in which each agent alternately listens, thinks and speaks,’ in which the terms, ‘listens,’ ‘thinks,’ and, ‘speaks,’ are metaphorical. Both agents must participate effectively for interaction to take place.

Crawford outlines many factors which combine to create a high quality interactive story. To condense what it a wealth of information on the subject, three factors determine the degree of interactivity: speed, depth and choice. Slow applications destroy interactivity. A mentally engaging application can be drawn from “human” characters. Where traditional stories must present readers with decisions that can only go one way, an interactive story must present players with decisions that hang on a razor’s edge.

Interactivity requires the audience to involve itself directly in the plot. This may appear to render interactivity incompatible with plot, however plot in this context, ‘metaplot,’ as Crawford dubs it, refers to rules, rather than a rigid structure. Rather than specifying the path, the rules define what a user can attempt to do. This allows a great deal more freedom for the user than they are given during a traditional story.

Summary

Traditional stories follow a linear development. This is due to the very nature of language, as neatly summed up by Crawford: ‘linearity is the inevitable outcome of using language to relate the story.’

While interactive stories can also use language, computational devices are the perfect medium for interactive storytelling; they grant the ability to use moving images and sound, and, most importantly, facilitate user input. Interactivity depends on the choices available to the user.

Competitor Analysis

I have identified three competitors to my application in this market. All three are available for a range of iOS devices, including iPad.

What follows is a discussion of the strengths and weaknesses of each app, before a summary of my key learning outcomes.

Stepworks' Kidztory 'The Ugly Duckling': Interactivity

Stepworks' Kidztory's main focus is on presenting well-known fables in an interactive format. Their adaptation of 'The Ugly Duckling,' is fairly simplistic, but this is fitting for the audience.

I found this app largely boring (I could also say 'safe,' which wouldn't sound so bad) but for two things: the sound effects and the character's expressions. Arguably, I shouldn't comment out of my personal opinion, however I would reason that if it can't keep me entertained, how much less so will it be able to keep a child with a shorter attention span entertained?

The graphics of this app are charming, if sparse, and fit with the cute, gawky style of the app as a whole. They convey an arts-and-crafts, collage style; they almost look like something a child would make.

The thing that this app does best, though, is the way it uses graphics and sounds t convey the personality of its characters. The sheer joy on the ducklings' faces as they pop excitedly out of their shells can't help but bring a smile to your face. It manages to be at once adorable and comedic. Perhaps it is the unintentional value of the comedy that makes it so appealing.

Throughout the story, each character on the screen will make a sound when tapped. This simple user interaction, as opposed to having all sounds played out automatically, makes a significant contribution to the enjoyableness of the app.

Kidztory push the benefits of their apps as encouraging a love of reading, discovering sounds and learning valuable lessons. They have an impressive collection of book to market already and no doubt a loyal fan base. They promote their other stories in each app. They also promote themselves through testimonials and positive reviews.

If I had a criticism, it would be that the company's apps as a whole appear to be a bit formulaic, and that once you've seen one app, the others would lack anything new. The child's voice used to narrate the story is also quite annoying.

Fig 2.1 Pop! A duckling emerges from its shell

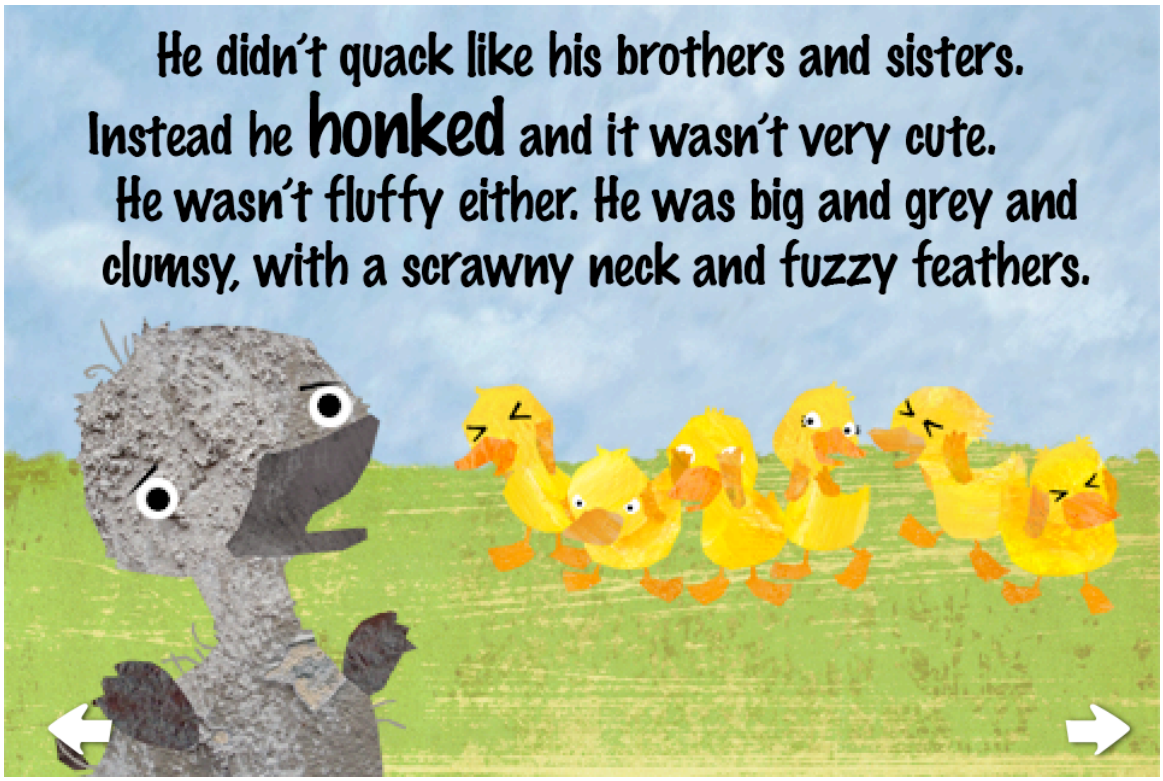

Fig 2.2 Ugly Duckling makes a heartbreaking sound when tapped

Tui Studio's Little Bella's I Close My Eyes: Theme

As a bedtime story, this app has a slightly different purpose than my application, however I was able to take away learning point from it despite this.

As soon as the story begins, you are transported to a tranquil, dreamy, innocent world; the scene and mood is immediately set. What plays out is an extremely simple concept, yet one which encompasses to many positive emotions such as aspiration, happiness and dreaming. The repetition of 'I close my eyes,' at the beginning of each video enforces the purpose of the app.

The character of Little Bella is very endearing and would appeal to children. Although she is seen only at the beginning and end of the story, her voice is heard throughout. The app conveys enough character and emotion to prevent boredom.

The interactions in this app are very minimal, with their only being need to metaphorically turn the page, or press the arrow to advance to the next scene. Again, this is in keeping with the purpose of the app, too much excitement would be counter-something???

The story has no real plot in the traditional sense, more a series of ???, but this isn't detrimental in this context, the end still has a resolution.

Little Bella's is very successful at what it sets out to do.

Fig 2.3 A dreamy world

So Ouat's Eleanor's Secret: UI

The main lesson I took from this app was drawn from the way the UI functions. Namely, that it doesn't always function as well as it could. The problem with the UI in Eleanor's Secret is that it's not clear what anything in the interface does. It is necessary to discover by trial and error.

The options that are present on the screen are confusing. The flags are the most obvious, however perhaps not to a child who mightn't know the symbolism. Questions are constantly raised in the user's mind, like, why is the menu icon a cat? What does the swirly orange circle do? What is the number on the bottom right of the screen?

It is a particular shame what the creator's did with the title screen (Fig ?.?). They went to all of the trouble of creating rich, immersive HD graphics throughout the whole story, and then ruined the experience with a gaudy title screen. The grey border, bouncing social media icons (which, I would argue, is inappropriate for the target age group anyway), newsletter registration and automatic scrolling app icons all detract from the main feature. It's like they forgot that their app will sell their other apps without their having to do anything. Quality sells itself.

One area where the app shows promise, though, is the educational aspect it provides. This app pushes its educational aspect much more than the preceding two. The storyline is geared towards learning, that is its theme. Dual-language…. If you can figure out how to get the transcript of the story on screen, it has a useful feature called 'Explain to Me' which provides a definition for some of the words. Again, though, this is blighted by usability issues. The app requires you to click the X to close the current definition before you can select another. As the definitions sometimes cover other selectable words, I can see there was logic behind this, however I found it consistently counter-intuitive, as I would forget the necessary intermediary stage and become quickly frustrated. The other two aspects, the 'Vowels,' and 'Show Me,' sections, lacked consequence and felt like unnecessary add-ons.

Fig 2.4 Title screen

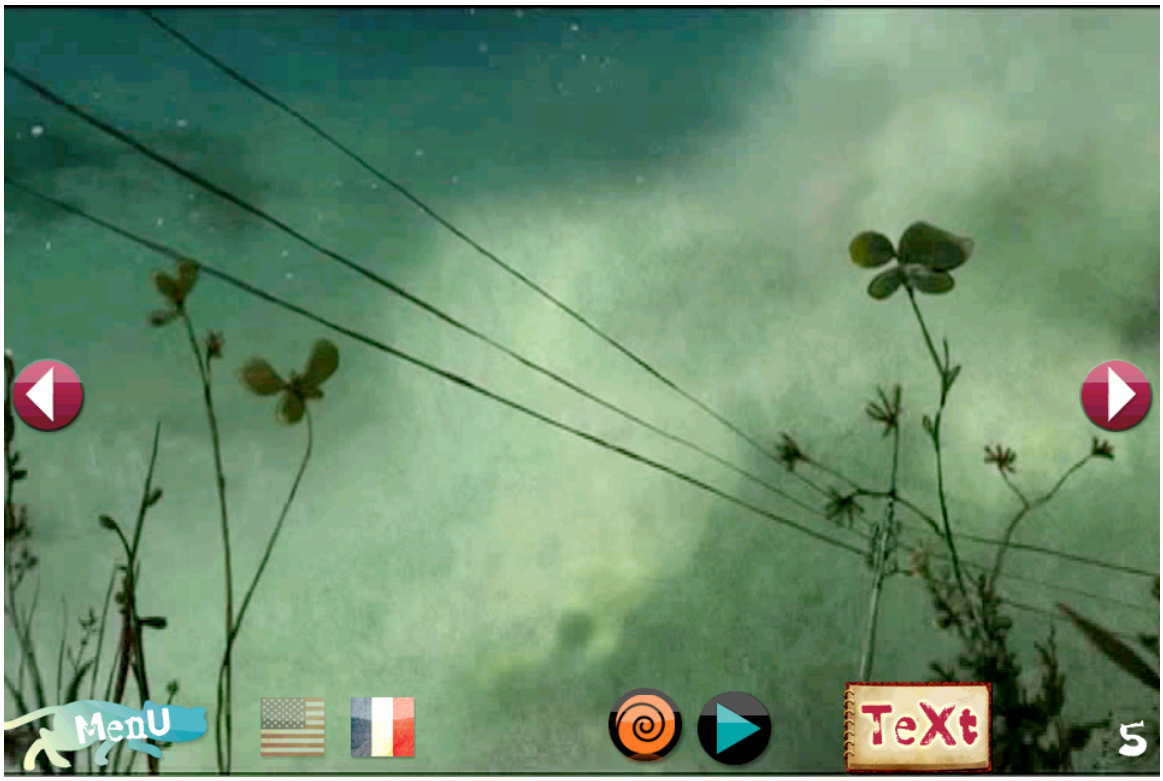

Fig 2.5 Confusing icons

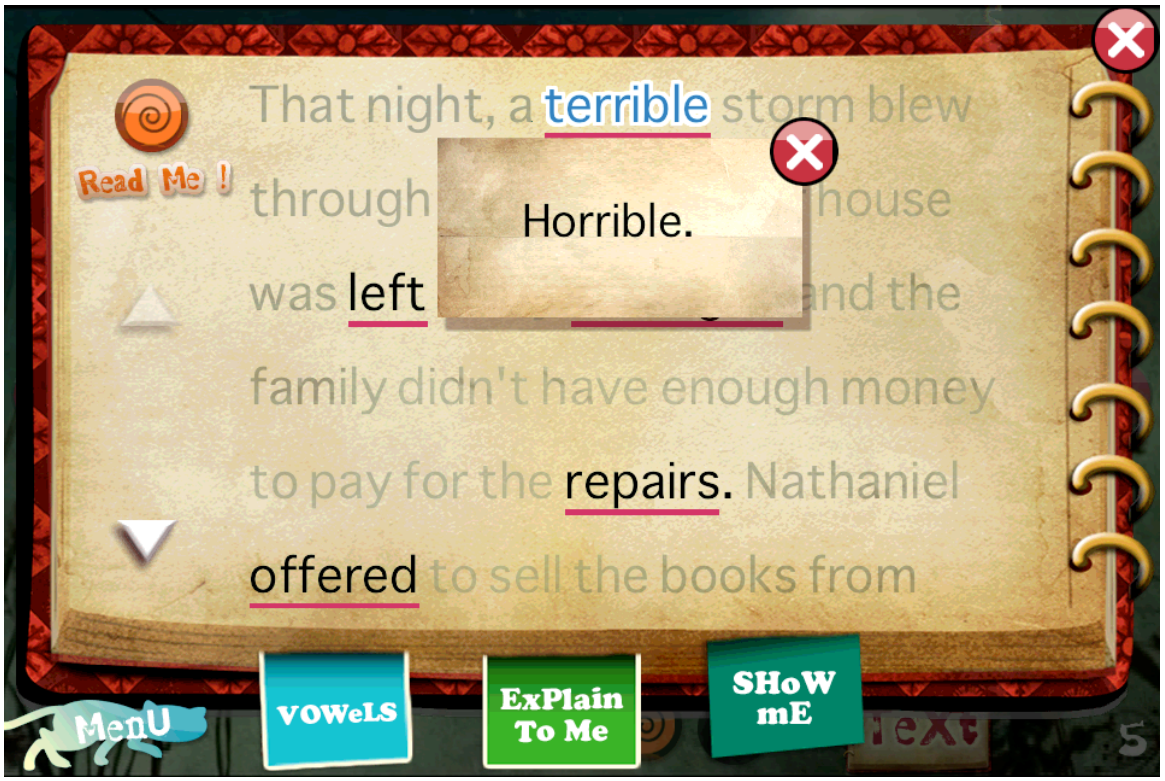

Fig 2.6 The 'Explain To Me,' section

Fig 2.7 The 'Show Me,' section

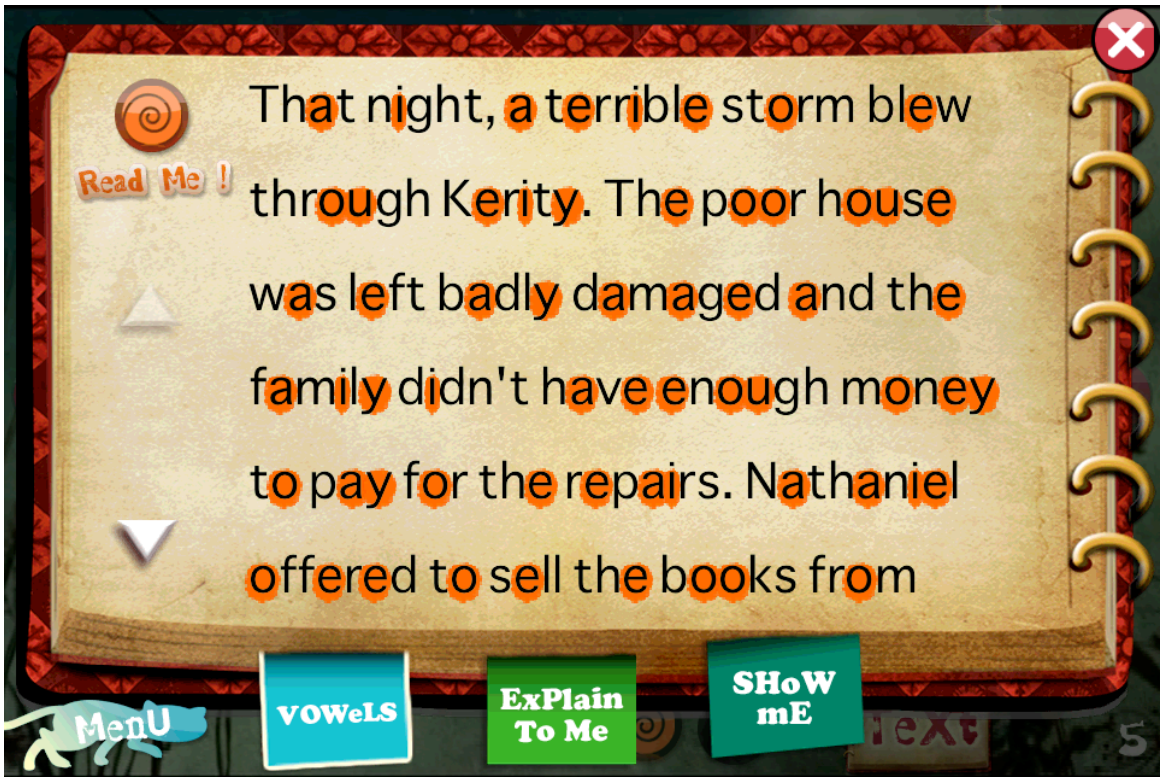

Fig 2.8 The 'Vowels,' section

Key Learning Outcomes

Story & Characterisation

Both story and interactions must work together for an app to be successful. None of the apps had a great deal of interaction, with only The Ugly Duckling providing interactions additional to moving backwards and forwards through the story. The lesson I've taken is although it's important not to get hung up on interactions, both should be given equal time and consideration in the development process.

Using classic fables as a storyline for an application is a solid strategy, these stories have endured for so long for a good reason. Parents are able to recognise the titles and are thus reassured that they know the content of the stories. Undeniably, there is satisfaction to be gained from reading a story to which you know the ending. On the other hand, an original story can create a heightened level of engagement through curiosity and discovery.

Characters are highly anthropomorphised. This is in agreement with what Crawford put forward about stories being about people, even if the references are 'indirect or symbolic.'—(Crawford, 2004)

Interactivity

Although all three apps claim to be interactive stories, none of them truly are because the user cannot influence or change the plot in any way.

Purpose

Parents are ultimately making buying decisions. In order to part with their hard-earner cash, they need to be reassured of value for money. An app with an educational purpose will contribute to this impression, as will assurance that the app will be engaging and fun enough to keep their child entertained for more than a few minutes.

Genre

Further to the three I chose to feature in detail, I examined supplementary apps in order to gain a wider picture of the apps existing for this market. I concluded that, in terms of features, the dominance was towards either stories *or* games, with a gap for an app which fuses the two: the narrative aspects of a story and the interactive aspects of a game.

Navigation

The navigation in all three apps was very linear, with options only to advance or reverse through the scenes one at a time. When the apps are closed and reopened, some will ask the user whether they wish to continue the story from the point they left off, or to start again from the beginning. There is a gap here for an app which offers multiple story branches.

Narration

Narration is much more important than I had initially thought. It is possible that an app could have success without narration, however if this is undertaken, the type of voice given to an app can make a large impression of users. Transcripts should be easily accessible, if not permanently available on screen.

Graphics

The majority of the apps I looked at used a cartoon style, although I have discovered that a range of styles can be effective as long as they are attractive.

Native iOS vs Web-based applications

As of June 2011, there have been 200 million iOS devices sold around the world, there are 425,000 apps in the App Store, with a total of 14 billion apps having been downloaded since the introduction of the App Store. 5,200 developers attended Apple's Worldwide Developer's Conference this year.

User experience

While perhaps the average user mightn't be able to tell you whether an application is web-based or native, they *could* tell you whether they enjoy the experience of using the app or not. Native apps usually provide a better user experience. Everything else is essentially a web page.

The iPhone sells itself on its user experience. iOS is a carefully crafted mobile OS. Apple has meticulously considered usability through every aspect of its design. iOS is also remarkably consistent, with only minor UI differences between devices. Consistency supports ease of use; the user has a mental model already in place when picking up any Apple branded mobile device.

The consequence of this is that users expect the same high quality in applications they download from the store as the stock applications that form the phone's basic feature set. Anything less will leave them disappointed and dissatisfied.

Convenience

There is also consideration to be taken with regard to what the user will prefer. People like convenience. Apple's App Store provides a very streamlined process to buy and install new apps to an iOS device. The process for monetising a web app, on the other hand, does not automatically provide an easy experience. Charging users for a web app may require them to enter lots of details into a web page.

Native applications are installed physically on a device, and thus are always available to the user, even when they do not have internet access. Furthermore, distributing apps through the App Store provides users with a launcher icon, which is far more difficult to forget about than a bookmarked website.

Trust

The Apple brand instills confidence in its users. Apple's history of people-driven, usable design has instilled confidence in their systems. Anything bought through Apple's App Store carries a mark of trust. Users know that each app has gone through an approval process controlled entirely by Apple. Additionally, the transparent system for rating apps allows a type of recommendation service through other users of the app.

There is, of course, trust imbued by web apps, particularly where the brand is well known, but this can't match to the trust people place in Apple.

Summary

Native apps provide a seamless experience, but will likely require more work in coding if a cross-platform solution is required. Producing a web app has advantages, but may lack the professional touch. The decision comes down to the individual requirements of the project.

For me, for the aforementioned reasons, I will be creating a native application. I also want my app to have the credibility that inclusion in a native app store provides, and to learn about the process of building and bringing native apps to market.

Comparison of Available Technology for Building Native Applications

Introduction

Objective-C is the primary language for traditional native iOS development. Apple provides their own IDE for developers to use, XCode. As I discovered in my research, though, this is not the only method for building native apps.

There are web-based application frameworks available which provide web developers with the tools to build native or native-like mobile applications using their existing skillset.

Both Appcelerator's Titanium and Nitobi's PhoneGap (acquired by Adobe in October 2011) have merit as tools for mobile development using standards-based web technologies. The decision of which to use is project-specific; each has strengths and weaknesses which make it more or less appropriate for different project requirements. The suitability of each depends on the scope, technical specification and platform requirements of the project.

From my research into the two, it very quickly became evident that Titanium would be the most appropriate for my project.

Cross Platform Support

The goal of cross platform development: code once, run everywhere.

PhoneGap applications are built in HTML, CSS and JavaScript, making it very accessible to web developers. By leveraging web views or Webkit native to the mobile devices, PhoneGap allows you to build for cross-device compatibility and with the ability to gracefully degrade for lower end devices, all with the same code base.

PhoneGap supports an impressive array of platforms, including iOS, Android, Blackberry, webOS, and Symbian. Their use of web views means that apps created are not truly native, although they do have access to mobile phone functions such as GPS, accelerometer, camera and contacts.

Apps built in Titanium become native apps. Titanium has comprehensive APIs that can be called to create and control all kinds of native UI components. Titanium currently supports iOS, Android and Blackberry development.

In particular, Titanium handles iOS really well. Unlike PhoneGap, however, it's not the case that you can code once and deploy seamlessly to multiple platforms. Code will likely need to be changed

between one platform and another. It makes sense given that one of Titanium's key features is its ability to utilise *native* UI components.

Documentation

Documentation for Titanium is provided through Appcelerator's website. Although the documentation isn't always up to date, it provides a solid enough starting point. My main gripes would be that at times it is hard to tell which version of their SDK an example applies to, given code examples are far too simplistic, and sometimes Titanium objects don't behave in the way the documentation indicates. In spite of these issues, I have yet to encounter a problem that hasn't been solved wither with the API documentation, or a Google search of the problem or error message.

Appcelerator have also provided an application called 'Kitchen Sink' which demonstrates the majority of the available functionality of the software. The code for the entire app is available to view and modify. Although at times sparsely commented, I have found this to be a fantastic resource.

Summary

PhoneGap excels as a cross-platform solution, with a straightforward development process for people with a web background. Titanium excels at creating convincing native experiences at the expense of cross-compatibility.

My decision to use Titanium was based primarily on these reasons: I'm not seeking to build a cross-platform application, I wish to build a truly native application, and Titanium's superior support of iOS native components

Titanium iOS Development Runthrough

The main elements needed for basic development using Titanium, without getting into table views and tab groups which I am not dealing with in this project, are 'windows,' and 'views.' Serving similar functions, both are essentially empty drawing surfaces or containers. An application must have at least one window, one of the few elements which can exist without needing to be the child of something. Views can be thought of as tags in HTML and as such are very versatile. Views must be the child of another element.

Create a Window

A window is created using the following syntax. Element properties are separated using a comma. The background colour of the window has been set to white, however it is not necessary to specify this. Colours can be specified using hexadecimal codes, or keywords, as shown. For my application, I want windows to display full screen, so I added that property to the declaration.

```
var window = Titanium.UI.createWindow ({
   backgroundColor: 'white',
   fullscreen: true
});
window.open();
```

Add Event Listeners and Animation

Event listeners receive triggered events from instances of objects. This simple example 'listens' for a button to be clicked, which will fire an alert box to the screen containing the given message.

```
button.addEventListener('click', function(e) {
   alert("message goes here");
});
```

In place of the 'click,' a number of other keywords can be used to fire events based on user input. These are: `dblclick`, `doubletap`, `singletap`, `swipe`, `touchcancel`, `touchstart`, `touchmove`, `touchend` and `twofingertap`. The event listener method, along with these keywords, form the basis of interactions, with the other foundation being the Animation object (Titanium.UI.Animation).

“The Animation object is used for specifying lower-level animation properties and more low-level control of events during an animation. The Animation is created by the method Titanium.UI.createAnimation.”—(Titanium API, 2011)

Titanium can handle the animation of certain properties on certain objects automatically. As in the following example, it can transition an element’s current colour to a different colour without you having to specify all of the colours in between. It can also handle changes in dimension, position and can even handle some 3D effects such as spinning. The animate method is used to accomplish this.

This example creates a view and specifies the background colour as red. Over a period of one second (1000ms), the background colour is changed to black (this appears as a gradual change).

```
var view = Titanium.UI.createView({
   backgroundColor:'red'
});
var animation = Titanium.UI.createAnimation();
animation.backgroundColor = 'black';
animation.duration = 1000;

view.animate(animation);
```

Animate Images

When it comes to animating custom images, Titanium provides several methods of achieving this, however in each case it is necessary to maintain a separate image per frame. If a very simple animation is, say, 20 frames, it can soon occur that you are working with hundreds, if not thousands of images. Naturally, this many requests would be very resource intensive and slow down the operation of the application, and as I have learnt that ‘slow applications destroy interactivity’—(Crawford, 2004), this is not an option I was willing to pursue.

Titanium offers integration with OpenGL for $29.99 per month which is capable of rendering and animating complex 2D and 3D elements. Given that the scale of my application is small, I did not consider this to be a cost-effective solution. It also represents a learning curve that I don’t have time to embark on.

A better solution for me, then, is to use sprite-based animation. In a nutshell, this means that each frame of an animation is contained within the same image file, known as a spritesheet. This is a very simple yet effective solution. The only downside is that is it very time consuming to producing each frame of an animation, and very unforgiving if one of your frame’s pixel positions is unintentionally out of sync with the others.

Conclusion

This research has allowed me to make informed decisions about my choice of development tool and platform. It showed me that rather than sticking doggedly to one development tool, the decision should be guided by the requirements of individual projects. If I wished to produce a cross-platform application, I would have perhaps chosen PhoneGap over Titanium. Both are being actively developed and improving all the time.

I learnt that the genre of interactive storytelling is much more complicated than I initially thought. It requires subtlety and precision to get right. Moreover, interactivity depends on the choices available to the user. Choice must be well thought out.

From my competitive analysis, I found that there is a gap in the market for a truly interactive storytelling application, rather than one which essentially is a story with interactions tacked on. I learnt that both story and interactions must be well thought out in order for an app to be successful.

Using an existing, well-known story has its advantages, however I feel that it would be more difficult to produce a successful interactive story this way, as even with precision reverse engineering, it may well turn out that the interactions get in the way of the story. Using an original story allows greater freedom for integration of the two.

It is advantageous if an application can include a moral, or some other educational purpose.

I was surprised about how much the voice of the narrator affected my enjoyment of an application. I found it very distracting when I disliked the voice.

As a result of this, I am going to plan my story and consider the choices that can be given to the user. I will then consider the scope that I feel I can achieve in the time available. I am going to produce a minimum of two graphical styles to choose from. I am going to continue to refer to Crawford's book throughout. I am going to refactor my code where necessary to make it reusable.

Appendix

App Store Approval Process

In order to submit your app to the App Store for approval, it is necessary to register as an Apple Developer. Becoming a developer also allows you to test your applications on physical devices, a crucial part of the testing process.

Approval can take anywhere from a few days to a few months, with the average being about a week. If approval is denied, Apple will inform you of the reasons why. Having made the appropriate changes, it is then necessary to re-submit.

The App Store Review Guidelines document lists the requirements for apps in such areas as functionality, content, interface and privacy, along with any other clauses Apple sees fit to include. A review of these guidelines will prevent time being wasted in having an app rejected because of an issue that is clearly stated as a cause for rejection.

Should you wish to charge for your app, you must choose from one of Apple's pre-defined price brackets. This is not a significant restriction considering there as 85 brackets. Along with the price of the app itself, Apple offers other revenue streams in the form of Apple's iAd network, in which Apple sells advertising and serves them in your app in exchange for a minimum of 30% of the advertising revenue, and through in-app purchases, which means you can offer additional features inside your app which users can purchase for an additional price.

Project Requirements

Functional Requirements

Must have:

- Allow users to return to the same point in the story as when the application was closed
- Provide usable interactions
- Provide means to move through story stages
- Provide story script in an easy to follow format
- Take advantage of the technological opportunities afforded by iPad development

Nice to have:

- Allow for multiple users
- Provide multiple stories, and multiple branches to each story. Allow users a choice on app launch

Non-functional Requirements

Must have:

- Provide entertainment and engagement for children
- Provide a sense of progression and achievement through the arc of the story
- Visually appealing
- Facilitate children in learning
- Consider future extensibility of the application, where the scope can be widened for the future of the project

References

Books

Crawford, C (2004) Chris Crawford on Interactive Storytelling. New Riders.

Vogler, C (1999) Writer's Journey: Mythic Structure for Writers. 2nd ed. Michael Wiese Production

Murray, J. H (1998) Hamlet on the Holodeck: The Future of Narrative in Cyberspace. MIT Press

Meadows, M. S (2002) Pause and Effect: The Art of Interactive Narrative. New Riders

Koster, R (2005) Theory of Fun for Game Design. Paraglyph Press

Websites

Lukasavage, T (2011) SavageLook.com » A Deeper Look at Appcelerator and PhoneGap. Available from http://savagelook.com/blog/portfolio/a-deeper-look-at-appcelerator-and-phonegap [Accessed 29 November 2011]

Patel, N (2010) Apple's App Store Review Guidelines: 'we don't need any more fart apps' -- Engadget. Available from http://www.engadget.com/2010/09/09/apples-app-store-review-guidelines-we-dont-need-any-more-far [Accessed 1 December 2011]

Apple Inc. (2011) App Store Review Guidelines and Mac App Store Review Guidelines - Apple Developer. Available from http://developer.apple.com/appstore/guidelines.html [Accessed 1 December 2011]

Locassa (2011) Apple App Store Approval Process Explained | Locassa. Available from http://www.locassa.com/index.php/2011/06/apple-app-store-approval-process-explained [Accessed on 1 December 2011]

PhoneGap (2011) PhoneGap. Available from http://www.phonegap.com [Accessed on 10 October 2011]

Appcelerator Titanium (2011) Titanium Studio | Appcelerator. Available from http://www.appcelerator.com/products/titanium-studio [Accessed on 10 October 2011]

Stepworks Kidztory (2011) Kidztory - animated children's story book apps for iPhone, iPad and iPod Touch. Available from http://www.kidztory.com/ [Accessed on 1 December 2011]

iTunes Game (2011) Little Bella's - I Close My Eyes - Animated Children's Book. Available from http://itunes.apple.com/gb/app/little-bellas-i-close-my-eyes/id304163263?mt=8 [Accessed on 8 October 2011]

iTunes Page (2011) Apps for Kids. Available from http://itunes.apple.com/WebObjects/MZStore.woa/wa/viewMultiRoom?fcId=394168962&s=143444 [Accessed on 8 October 2011]

Lee, A (2011) WWDC 2011 Stats: The Best Figures From Apple's Huge Press Conference. Available from http://www.huffingtonpost.com/2011/06/07/wwdc-2011_n_872430.html [Accessed on 5 December 2011]

So Ouat (2011) Applications ludo-éducatives pour iPad - So Ouat! Available from http://www.so-ouat.com [Accessed 2 December 2011]

Tui Studios (2010) Little Bella's I Close My Eyes | Tui Studios. Available from http://www.tuistudios.com/2010/08/17/little-bella [Accessed 2 December 2011]

Tui Studios (2010) I Love Little Bella. Available from http://www.ilovelittlebella.com/ [Accessed 2 December 2011]

Komenda, K (2011) A look at PhoneGap and Appcelerator Titanium http://www.klauskomenda.com/archives/2011/01/17/a-look-at-phonegap-and-appcelerator-titanium [Accessed on 29 November 2011]

Whinnery, K (2011) iOS Platform Overview. Available from http://wiki.appcelerator.org/display/guides/iOS+Platform+Overview [Accessed 10 October 2011]

Rowinski, D (2011) PhoneGap Creator Nitobi Acquired by Adobe. Available from http://www.readwriteweb.com/mobile/2011/10/phonegap-creator-nitobi-acquir.php [Accessed on 28 November 2011]

Kahney, L (2002) Apple: It's All About the Brand. Available from http://www.wired.com/gadgets/mac/commentary/cultofmac/2002/12/56677 [Accessed on 28 November 2011]

Appcelerator (2011) The Native Advantage. Available from http://www.appcelerator.com/products/native-iphone-android-development/ [Accessed on 28 November 2011]

Apple Inc. (2011) About the iOS App Development Workflow. Available from http://developer.apple.com/library/ios/#documentation/Xcode/Conceptual/ios_development_workflow/000-Introduction/introduction.html#//apple_ref/doc/uid/TP4000795 [Accessed on 28 November 2011]

Apple Inc. (2011) The Objective-C Programming Language. Available from http://developer.apple.com/library/mac/#documentation/cocoa/conceptual/objectivec/introduction/introobjectivec.html [Accessed on 28 November 2011]

Apple Inc. (2011) Tools for iOS Development. Available from http://developer.apple.com/library/ios/#referencelibrary/GettingStarted/URL_Tools_for_iPhone_OS_Development/_index.html#//apple_ref/doc/uid/TP40007593 [Accessed on 28 November 2011]

Appcelerator (2011) Titanium Quick Start Guide. Available from http://wiki.appcelerator.org/display/guides/Quick+Start [Accessed on 25 November 2011]

Appcelerator (2011) Appcelerator's OpenGL Module. Available from https://marketplace.appcelerator.com/apps/778 [Accessed on 15 November 2011]

Appcelerator (2011) Titanium API Documentation. Available from http://developer.appcelerator.com/apidoc/mobile/latest [Accessed October-December 2011]