PROTOTYPING A NEW INTERACTIVE EBOOK CONCEPT FOR ADULT FANTASY READERS

BY

Lauren C. Maricle

THESIS

Submitted in partial fulfillment of the requirements for the degree of

Master of Fine Arts in Digital Arts awarded by Digipen Institute of Technology

Redmond, Washington

United States of America

August

2018

Thesis Advisor: Randy Briley

ABSTRACT

The eBook market is largely dominated by limiting standardized formats that do not make use of the freedom digital publishing provides. Though artistic interactive eBooks exist, they are largely targeted for child audiences, rather than adult readers. Despite this trend, I believe adults would be equally receptive to interactivity and art assets in their eBook experiences, if formatted for a more mature audience. Through this project, I want to explore the untapped adult market for interactive eBooks. Using Unity, I created a prototype application for a new format that combines video game inspired assets with traditional eReader functionality similar to a Kindle or iBook. In addition to the reader, the application contains separate pages for an interactive map of the world and 3D models of characters from the story. Though this project is only a prototype and was limited in scope due to my ability and time frame, the basic concept allows room for expansion and improvement. My hope is that this project provides the foundation for more inclusion of art and interactive elements into reading experiences for adult fantasy readers.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

First, I'd like to thank to Ryan Finnerty, who played a huge role in my decision on what my thesis would be and helping me narrow my crazy ideas down into something somewhat feasible. I'd also like to give a special thanks to Jami Lukins who helped me realize UI is actually quite important when creating a full application, and then taught me how to do it right. Lastly, I'd like to thank all of the wonderful members of my committee, Randy Briley, Christopher Orth, and Mark Henne, who helped me solve countless problems and kept me on track through this whole process.

INTRODUCTION

Over the last decade, the eBook market has steadily gained traction among readers of all ages. The Kindle handheld eReader's 2007 release marked the beginning of this transition, and while the demand for print has not disappeared, the promise of quick and easy purchase and the ability to store a large number of books on a small device has turned a significant percentage of consumers towards the digital trend. While in 2013, the digital book market amounted to around 12% of worldwide book sales, 2018 projections suggest eBooks will account for approximately one quarter of all book sales worldwide.[1] However, the large majority of eBooks are simply digital recreations of print, and therefore are limited to text and still images reproduced on screens. The use of digital technology has standardized some features that aid or enhance the reading experience, such as text scaling, page reformatting based on screen aspect ratio, linked navigation within the text, search functions, etc. However, a very small number of eBooks truly capitalize on the flexibility and creativity that digital publishing can provide.

By this flexibility and creativity, I am referring to the ability to create immersive and interactive experiences that add to the reading experience. This is not to say that these more complex eBooks do not exist: in fact, while they only make up between 10% and 20% of the eBook market[2], there are many interactive digital books being published that are advancing our understanding of what eBooks can be. Currently, there are many platforms and types of paid software that help creators make interactive content under the epub3, Kindle, Kobo, or interactive pdf formats that are meant to be read using standardized reader apps that store a library of books. The creation of the epub3 format in

---

[1] "Ebooks - Statistic and Facts," *Statista*, 2018. https://www.statista.com/topics/1474/e-books/.

[2] Robert Springer, "The State of Ebooks 2017," *EContent Magazine*, January 18 2017. http://www.econtentmag.com/Articles/Editorial/Feature/The-State-of-Ebooks-2017-115709.htm

2011 allowed for the inclusion of audio, video, and widgets, but the method of inclusion is highly regulated by the HTML5-based structure of files.[3] While this marked a large step forward for the future of interactivity in standard format eBooks, the structure inherent in these standardized formats prevents the creator from exercising creative control over the general reading experience, and also severely limits the types of interactivity that can be included.

As a result, there are many creators publishing their books as stand-alone applications, sold through marketplaces like the Apple and Microsoft App Stores. Using this method allows for a much higher degree of control over the structure and interactive features of the app. Mark Coker, CEO of Smashwords, an eBook distribution company, claims that independent publishers have many competitive advantages including "faster time to market and more creative and promotional flexibility for their books."[4] Many children's and educational book creators have used this method to create narrative or educational reading experiences that expand beyond how we normally think of books, blurring the line between book and game. Richard Dawkins' "Magic of Reality" is one excellent example of an interactive eBook app, where the user can interact with the content through games and demonstrations while reading through the text. The interactivity helps to enhance the reader's understanding of the scientific concepts described in the book, such as the demonstration of light refraction properties with different types of lenses.[5] The use of a standardized reader would have made this type of

---

[3] "EPUB 3.1 Overview," *International Digital Publishing Forum*, January 5 2017. http://www.idpf.org/epub/31/spec/epub-overview.html.
[4] Robert Springer, "The State of Ebooks 2017," *EContent Magazine*, January 18 2017. http://www.econtentmag.com/Articles/Editorial/Feature/The-State-of-Ebooks-2017-115709.htm
[5] Transworld Books, "The Magic of Reality for iPad by Richard Dawkins – trailer," Youtube, September 23, 2011. https://www.youtube.com/watch?v=eBrP3-Ep3ww.

enhancement impossible, but by publishing as a stand-alone application, the creators had full control over the design, art, and structure of each page and feature.

While most of these books, particularly those that fall under the narrative category, are aimed at children, I believe that similar ideas can be applied to eBooks for a young adult and adult audience. As someone who grew up consuming a large amount of fantasy content, books and games alike, I always felt that there was a missed opportunity for adding interesting visual content to the novels that I read. The tradition of including artfully drawn world maps and scattered illustrations is long standing in the fantasy novel canon. Brandon Sanderson's Stormlight Archive series, particularly his most recent book Oathbringer, serves as a great modern example of how the addition of illustrations and maps can be used to compliment the written story. Sanderson includes multiple detail maps of specific locations in addition to his world map, and many illustrations that depict subject matter such as current fashion trends in his world, technological designs, and sketchbook pages from some of the main characters. Many readers, including myself,

Figure 1: Illustration of fashion trends, from Brandon Sanderson's Oathbringer

Figure 2: Illustration of airship designs, from Brandon Sanderson's Oathbringer

found this content to be a captivating addition to the text. The unique quantity and subject matter of the illustrations made *Oathbringer* uniquely exciting, but I believe that enthusiasm for artistic content could be increased even more by adding a level of interactivity and design beyond what standardized formats like Kindle and ePub are capable of.

Figure 3: Map of Kholinar, from Brandon Sanderson's Oathbringer

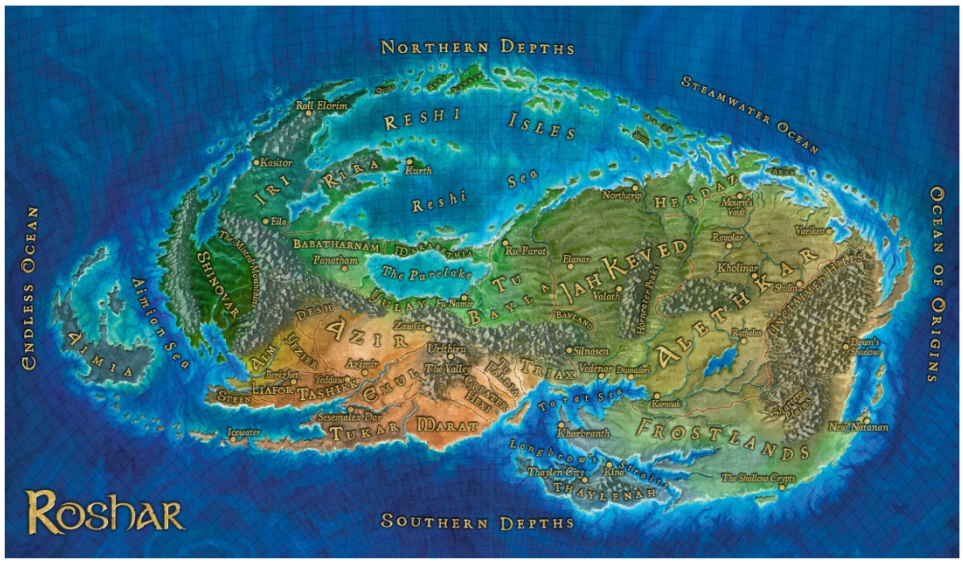


Figure 4: Map of Roshar, from Brandon Sanderson's Oathbringer

I have also found, through personal experience, that there is a large overlap in the audiences of “nerdy” entertainment, by which I mean fantasy and sci-fi novels, video games, board games, and tabletop RPGs. Because of this overlap, I feel that adding interactive content inspired by video-game style art assets would strongly appeal to this crossover audience. Using this principle as a foundation for my idea, I wanted to build a digital book product targeted for an adult demographic that would merge the visual traditions of the fantasy novel canon with the capabilities of digital publishing, using assets and features based on video game content.

SECTION 1: Overview of Process

This project consisted of several different separate but dependent components: the technical portion of building the app, the artistic design of the map and illustrations, the design and modeling of three characters, and the design and implementation of the user interface. One of the most challenging aspects of the project was maintaining a proper balance of time and effort for each portion, while also creating a cohesive feel between all of the separate elements within a single application.

The design of the application structure itself strongly informed the way the rest of the elements developed. The original concept was to build inside the ePub3 format because of its accessibility across a variety of platforms and easy ability for distribution. However, while ePub3 provides many significant benefits in publishing and distribution, it imposed too many restrictions on my ability to control the visual design and content structure of the product. Next, I researched other, less popular standardized formats and creation software that help creators make interactive content, such as Kotobee or Adobe InDesign. In the end, I also found them to be too limiting. Because I wanted to have freedom of design in the navigation and structure of the application, as well as the ability to include 3D assets, I decided to build it myself using a game engine that had the ability to publish for a large variety of devices.

My research on free-to-use game engines pointed me towards Unity as the best option for developing the type of app I was looking to build, as it had a strong forum support base and it comes pre-equipped to handle 2D programs and simple UI. I began this project with no game engine experience, so a large portion of my work for the first few months consisted of learning how to work within the engine's pre-built systems, as

well as learning C# to fill the gaps where I would need to build my own custom functionality.

Though I wished to plan and execute the creation of the application and artistic aspects, I did not want to create all the text and worldbuilding content myself. This was partially because I wanted to save time, but also because I believed an author would be able to come up with more interesting content than I am capable of creating. I was lucky enough to able to work together with an author who is currently writing a fantasy series, and he provided the written content, worldbuilding and characters, and also gave me input on design choices. The author, Dorian Zimmerman, is currently in the process of writing the first novel but had already drawn out a rough map and written out a huge amount of worldbuilding information. Since the product is only a proof of concept, we were able to work with the existing information and he wrote a few of the chapters specifically to showcase content included in the app.

Using the Unity game engine and Dorian's content, I built a prototype for an application that showcases the new type of interactive eBook I am trying to create. Because I wanted the reading experience to feel as close as possible to a traditional book, the reader itself is one page of the app, while the interactive artistic elements—an interactive map and character information—were created as separate pages. The goal was to make the application feel like an enhanced reader rather than a game. Within each of these separate pages, I developed the technology and art assets to fulfill the purpose of that page. The main menu had to establish the look and feel and capture the user's interest, while linking to all the pages. I wanted the reader page to function as close as possible to a Kindle but feel more like a traditional book. The interactive map had to have a visually appealing map of the world that the user could interact with and learn many details about the world. The character page needed to display 3D models and biographical information of multiple characters without spoiling any information the user

had not encountered in their reading experience. I wanted the application to include art and functionality that would showcase a new way to use digital publishing to enhance the fantasy novel reading experience.

SECTION 2: The Reader

While there were a lot of options and resources to help with many of the technical elements of building the app, I found few resources to help with the reader portion of the app. My original goal was to try to replicate the Kindle format as closely as possible while adding a bit more visual interest. The first idea for structuring the reader consisted of creating a script that would read a .txt file containing all of the text with html style tags to define font changes and line breaks. The script would then output the content of the .txt file as rich text that would autofill the pages, automatically detecting the number of pages and where they would break without creator input. This was partially for versatility in reapplication, and also for easy adjustment of the text during the creation of this proof of concept. I also hoped that using this method would allow me to add user-interactability that would enable the user to change the font size and color, similar to the Kindle's customizability.

Unfortunately, due to my limited knowledge of programming languages and lack of time to learn more advanced scripting, I had to make compromises on the reader's adaptability. After looking into the methods needed to set up the reader according to my original idea, I found that the process required knowledge of basic SQL for setting up a database. Though this was not impossible for me to learn and implement, I felt that my project would benefit from putting more time into some of the art and design components of the project. As a result, I decided to simplify the process of building the reader and proceeded with a manually formatted text box setup.

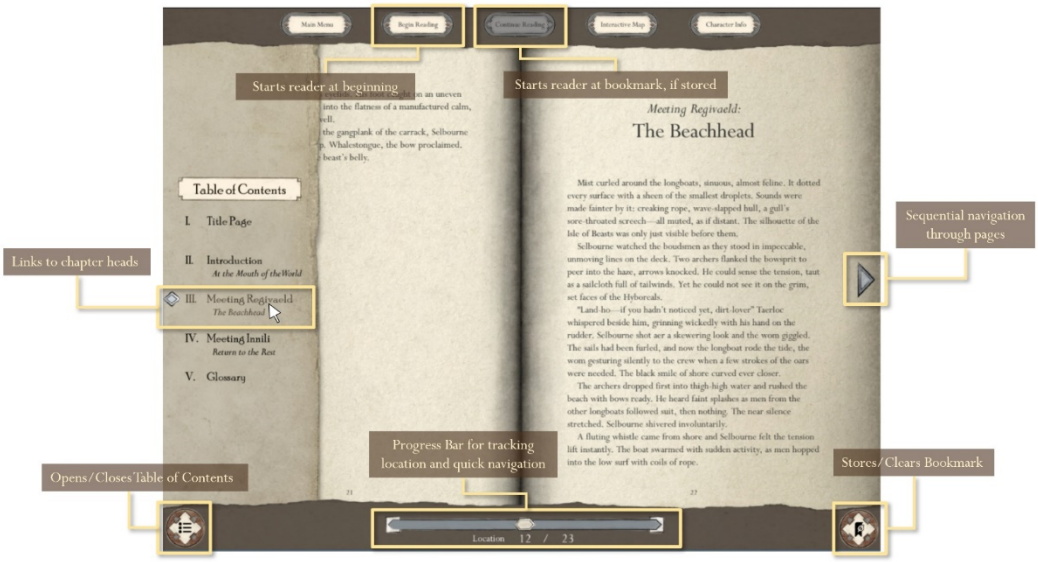


Figure 5: UI layout for reader page

The current architecture of the reader page is set up to mimic the two-page spread of a book. I used a combination of Unity's pre-built text box asset and empty game objects as anchors to create the layout. Every two-page spread consists of a left and right page anchor. Each anchor object serves as a parent object to a text box containing the main body of the page, a text box containing the page number, and—if the page is the beginning of the section—two additional text boxes that contain the section name and title. Each pair of pages was then parented to a single game object.

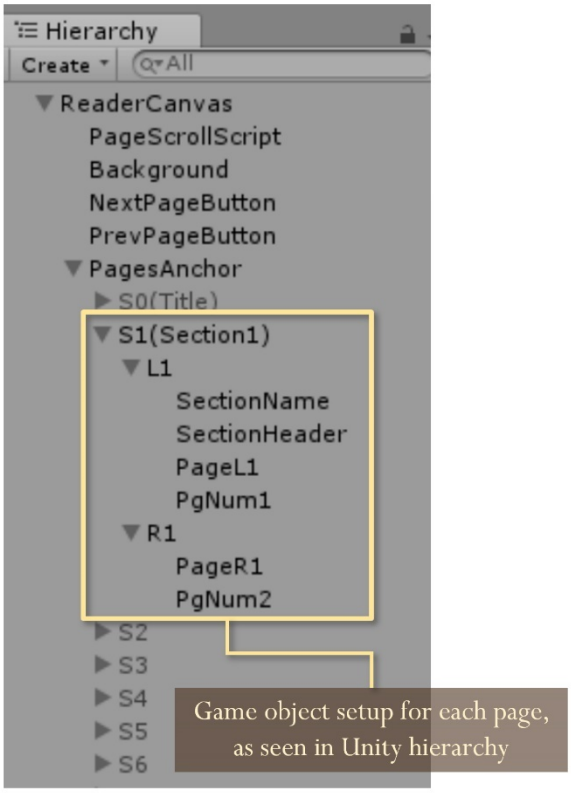


Figure 6: Hierarchy of game objects for each two-page spread of the reader

The reader script creates an array that is filled with those game objects, and then proceeds to cycle backwards and forwards through that array, setting the game objects active when they are the current page and inactive when they are not. There are three different ways to cycle through the pages in order. First, there are the arrow buttons on either page. Second, the scrollbar at the bottom of the page, which also shows the reader's relative location in the entire body of text as well as a live counter that tells the reader their page location out of the total number of pages. Lastly, the reader can navigate through the pages with arrow keys. These are all different applications of the same page-scroll function in my reader script.

The table of contents opens to show a set of buttons that link to the beginning of each section in the text. These buttons trigger a function that cycles through the array of

game objects, setting all pages inactive first and then activating the page that they to link to. I built in a page counter that helped me regulate the tracking of the pages between the multiple navigation methods, and also provided the live counter paired with the bottom scrollbar that prints out the reader's numerical location.

The last major function in the reader script allows the reader to create a bookmark that they can return to after navigating to a different scene. Clicking on the bookmark button at the bottom right of the window will create a bookmark by storing the current active page count as an int variable. The variable is stored using Unity's Player Prefs API, and sets a global static boolean variable, isBookmarked, to true. If a bookmark has been stored, the button will switch out to become a clear bookmark button. Clicking this button resets the PlayerPrefs int variable to back to the start page and sets the global static boolean, isBookmarked, to false.

When the scene is loaded, the isBookmarked boolean variable is read to determine whether the starting page should be set to the bookmarked page or the first page. There are two different buttons on the main navigation menu: "Start Reading" and "Go to Bookmark." Using the "Start Reading" button, in addition to loading the reader page, sets a global boolean BookmarkStart to false. The reader scene, on load, reads the BookmarkStart variable and determines that the start page should be set to 0. The "Start Reading" button is always interactable and will load the reader scene at the first page, whether there is a bookmark or not. The "Go to Bookmark" button is only set to interactable if the "isBookmarked" boolean value is set to true. If there is no bookmark stored, the user can only open the reader from the first page. However, when there is a bookmark stored in the Player Prefs, loading the reader scene with this button sets the BookmarkStart global boolean to true, and when the scene loads and reads this variable, it will set the start page to the bookmarked location. As a result, when a bookmark is

stored, both the option to open the book to the start page or open the book to the bookmark are available.

Though somewhat convoluted, these were the simplest solutions I found to mimic standard eReader structure and store the user's location in the reader when they navigate to the other pages, like the map or characters. Because the author and I decided to only include small excerpts of the book's text for this proof of concept, entering and formatting the text took up little time. However, if this project were to be implemented on a full scale for the novel on release, I think the app would benefit from the time investment of building the database for the text. On the scale of hundreds of pages, the method of manually formatting the text boxes would be extremely time consuming and if a change that needed to be made affected a page break, the rest of the pages after that changed page would also have to be reformatted. If the reader was set up using this database structure, all of the scripting work for the features would have to be restructured or completely redone. However, give the current setup of the app, these features all function well and help demonstrate the features I will want in a full-scale application.

SECTION 3: Interactive Map

The creation of the interactive map can be divided into two main stages: the first researching and creating the artistic rendering of the map, and the second was the technical implementation of the map and its interactive features inside the app. Both parts provided many challenges, and for this reason this portion of the application took the longest amount of time to create.

As the artistic direction of the map was one of the earliest things I worked on, it played a large role in informing the style of the rest of the app. My first few attempts at creating the style for the hand-painted map were largely unsuccessful in creating a style that fit with my idea for the product. Originally, all of my reference images were pulled directly from modern games and fantasy novels, which were stylistically inconsistent and did not provide a clear direction for creating the antique or aged feel I wanted. Many of them also did not include color, which I felt was necessary for my map to help convey the drastic variety in the different climates and zones.

After a few unsuccessful attempts at adding color, I revisited my reference collection and decided to gather more reference from real historical cartography. After looking through many eras of maps, I found that the Age of Exploration, particularly the late 16th century, gave birth to many interesting examples of cartography that fit the aesthetic I was looking for. The 16th century maps that influenced my map the most were Abraham Ortelius' Islandia (see Fig. 7) and Olaus Magnus' Carta Marina (see Fig. 8).

Figure 7: Islandia by Abraham Ortelius

Figure 8: Carta Marina by Olaus Magnus

Looking at these maps helped me decide on a “watercolor” style of loose and thin color patches. I also chose to desaturate the colors and balance them towards a more yellow hue to mimic the look of watercolors on aged paper—though I chose to make my map appear slightly less “yellowed” than the actual reference so that it did not feel overpowering.

I also used the Age of Exploration maps as direct reference for some of the symbolic representations of geographical elements, like the mountains and water. However, as most of the maps from the 15th and 16th centuries did not include much information about climate or terrain beyond mountains and water features, I turned towards modern cartography and fantasy maps for examples of more specific symbols, such as the dunes, cliffs, and forests. I used a combination of mimicking modern fantasy maps and my own abstraction of photographic representations of different types of terrain to decide on how to color the different regions. Ulla Thynell and DeviantArt user “Tiphs,” both modern fantasy map artists, provided some inspiration for my decision on how to handle color, particularly in their use of loose edges on geographical boundaries and patches of higher and lower color saturation.

Figure 9: Map of Midgard by Ulla Thynell

Figure 10: Hibendrill Worldmap, by DeviantArt user Tiphs

I also wanted to include more design elements that conveyed an Age of Exploration feeling. Carta Marina contained the interesting feature of multiple compass roses in the water that marked points of intersection for sailing lines.[6] As the layout of Aracem is extremely ocean-dominant, I included this style of compass rose and sailing lines to provide an interesting design element that would help break up the large masses of water, while also adding to the historical feel of the map.

[6] H. Thomas Rossby and Peter Miller, "Ocean Eddies in the 1539 Carta Marina by Olaus Magnus," The Oceanography Society, 2003. https://tos.org/oceanography/assets/docs/16-4_rossby.pdf.

Figure 11: Full illustrated map of Aracem

In addition to creating a map that was aesthetically pleasing and stylistically cohesive, I also needed to contain a large amount of details and information that—when viewing the image as a whole—became illegible. I found that game maps from fantasy RPGs, like Skyrim or Horizon: Zero Dawn, contain huge amounts of informational markers that become barely legible when viewing the whole map, but give the player the ability to control zooming and panning or toggle visibility on specific types of information to change the level of detail—and therefore increase legibility. I wanted to follow a similar concept on my map. My original idea was to have LOD's on the map, similar in concept to LOD's on models in video games. I had planned to use Unity's built-in LOD system to achieve this, but ran in to a couple of different issues that led me down a different route.

The first problem was a conflict of overlapping information. I wanted to include a visual representation of political borders in addition to the geographic information conveyed with color. However, I found no examples of a map that included both of these types of color information in a way that I found successful. This led me to the idea of layers that could toggle on and off; that way the user could decide when to view the

political regions but turn them off when they were hiding details or distracting the user from what they wanted to see. Once I had decided on toggling layers for the political regions, I also thought it was a beneficial feature to include for other elements as well, giving the user an added level of control. In the end, I landed on four different toggling layers: political regions of control, cities, roads, and landmarks. Figure 12 shows each of the toggling layers independently turned on. However, the user can turn on as many of the toggling layers at the same time as they wish.

Figure 12: Four toggling map layers and UI toggle box

The introduction of toggling layers did not, however, completely remove the need for a LOD-like system, where certain parts of the "permanent" text—meaning text not pertaining to cities, landmarks, roads, or political regions—disappeared when they became illegible. Rather than using Unity's built-in LOD system, which I found didn't work with my scene's camera setup, I decided to use animation curves. Rather than basing animations on a time variable, I found I could base the animations on the zoom (or rather the field of view, which is the actual changing property) of the camera. I set up multiple layers where my text was grouped by relative size and then controlled each layers' alpha transparency based on the camera's field of view. I ended up with four layers of "permanent" text that would fade in or out on a curve set between two values on the camera's field of view. Figure 13 shows a screenshot of the inspector with the animation curves for each text layer, as well as an enlarged editor window for one of the animation curves. The alpha (between 0, transparent, and 1, full opacity) is set on the vertical axis, and the camera's field of view is on the horizonal axis. You can see on the enlarged animation curve that the alpha starts to fade when the camera's field of view is at 150, and the alpha value hits zero, meaning the text is completely invisible, once the camera's field of view is set to 190. This method had the added benefit of creating a subtle fade to the text layers' appearance and disappearance, rather than having it "pop" in and out like it would with an LOD system.

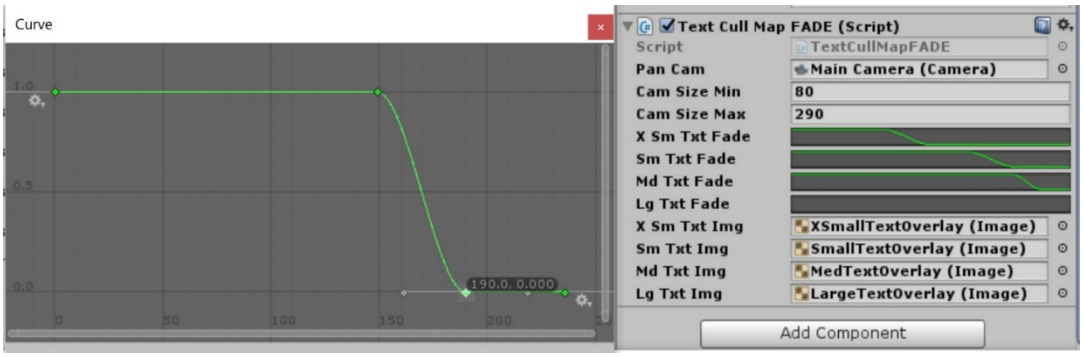


Figure 13: Text layer animation curves

I also wanted to include another level of interactivity and further depth of information on the map by adding interactable location markers that would, when clicked, pull up a window that provides the user with detailed information about that location. The idea for these popups originated from very early research into the world of interactive eBooks, where I found an app called “Barefoot World Atlas,” that I thought showed an interesting concept for simply conveying political and geographical information about different places all over the globe.

Figure 14: Map location pop-up

Using “Barefoot World Atlas” as an example, I condensed relevant statistics and a short description provided by the author for a few particularly important cities. I intended for these popups to add another level of interest to the map and provide readers with a easy place to go for collected information about specific locations. While “Barefoot World Atlas” has lots of information that is relevant to modern culture and environmentalism, the author and I chose pertinent categories of information that matter

in the world of Aracem. On three of these popups, I also included small thumbnail illustrations of those locations to give the reader an added layer of visual information about the city.

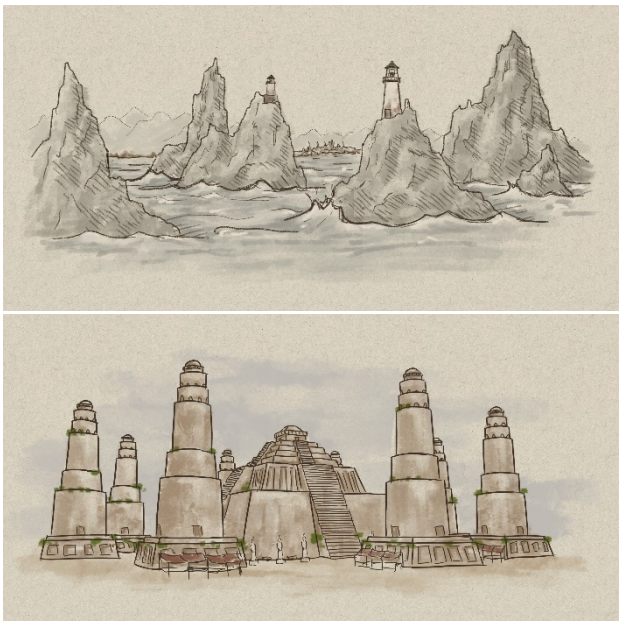


Figure 15: Location Illustrations for map popups

If this were to be taken to production, I'd like to include thumbnail illustrations like these or small maps like Brandon Sanderson's map of Kholinar (see Fig 3) on every location where they might help the reader better understand locations discussed in the books. These illustrations, while small, add another layer of visual information that cannot be included on a large-scale map, and can help supplement the readers' understanding of the important elements of specific locations.

Creating the popups was, from a technical standpoint, fairly simple. Each one consists of an empty game object to which an image, title, and scroll view are parented. Clicking each location's marker sets the parent object to active, and then I have a separate button on the popup window to close it. To prevent the user from accidentally moving the map while interacting with the scroll view, I added an Event Trigger component to each popup to disable the camera move script when the mouse enters the popup window and

enable when it exits (see Fig. 17). I also added an On Click event to the close window button (see Fig. 16) that re-enables the camera script in addition to closing the window, as the Pointer Exit event cannot occur if the popup window game object is no longer active.

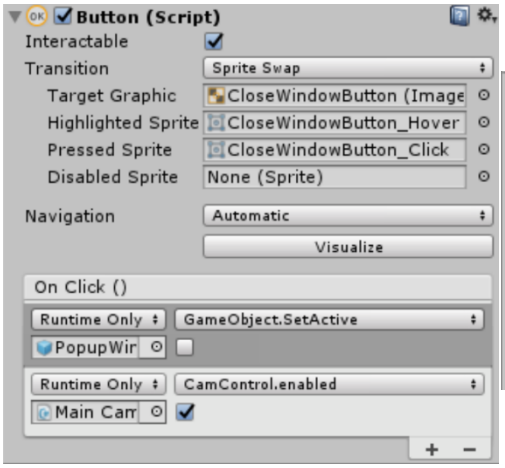


Figure 16: Pop-up close button sets window game object to inactive and also re-enables the camera move script

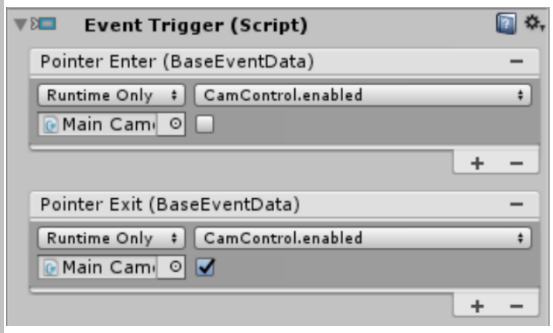


Figure 17: Event trigger disables/enables camera movement when mouse pointer enters/exits pop-up window

I ran into a few issues while setting up the technical functionality of the map. The first of these issues was setting up the location markers to move with the zooming and panning of the map, while the rest of the UI elements, like the toggle window and the location popups, stayed static with respect to the camera. In Unity, the UI elements are set up on Canvases, which can be rendered to a variety of different modes. For the elements that I wanted to remain static, I needed my canvas to be rendered to "Screen Space – Overlay," which means that the canvas is always rendered in front of your camera view and is not affected at all by camera changes. The location markers and map layers, however, needed to be rendered to World Space, as they needed to move with the map relative to the camera. As a result, to make all of my elements function properly, I needed to set up two separate canvases, one in World Space and one in Screen Space – Overlay.

The last piece of the map functionality was the user-controlled zooming and panning camera. The camera script went though a variety of iterations before I landed on the type of functionality I was happy with. I started the process of setting up the camera by downloading two scripts from the Unity Asset Store: Pan and Zoom Orthographic Camera by Lazy Cat and RTS Camera Pro by Voxel Frog. Both scripts contained elements I liked and elements I disliked. The Orthographic Camera script was more relevant to the type of 2D setup I was looking for, but method of controlling the navigation was strange and not intuitive to users. The RTS Camera Pro script had much more intuitive navigation but the functionality was much more complex than I needed and set up for 3D. I had hoped to create a zoom and pan function that worked more intuitively, like a traditional click and drag camera. However, despite doing a couple months of research and many botched attempts, I was unable to create a script that would provide a more traditional click and drag functionality. As a result, I decided to compromise and to devote my time to other aspects of the project. I settled on making slight modifications to Lazy Cat's Pan and Zoom Orthographic camera script, as it functioned in a way that was closest to what I wanted.

From start to finish, the map was the most complex and involved portion of the app to design and build. As I had no experience with both the artistic and technical elements of the map, it required the most research and the highest number of iterations. Because of the complexity of establishing just the basic level of technical functionality, I had to make a few compromises on some of my loftier technical goals. If I had a better grasp of C# and Unity's features, I would have liked to create a more intuitive pan and zoom camera system, and also include the character tracker I originally proposed in my first presentation. This character tracker would show you the location of the main character and a selection of other important characters based on the reader's location in the novel. The reader would be able to see the location as it corresponded to the current

"Last read page," and have an interactable slider that would allow the reader to see past locations as well. As the reader progressed though the novel, more locations would unlock and each character's path would get longer. This idea is based on a website built for the A Song of Ice and Fire universe called Quartermaester.com.[7] Unfortunately, including this feature was too complex for me to figure out in the time I had to complete the project. However, if this concept was taken through to completion for a full novel, this is still a feature I would like to include.

[7] "Interactive Game of Thrones Map with Spoiler Control," http://quartermaester.info/. Last accessed Aug. 29, 2018.

SECTION 4: Character Page

The character page, another core creative feature of the app, gives the user a way to visualize and better understand characters from the novel. As an aspiring character artist, I was the most excited about the creation of the character assets. Though illustrations are and excellent way of presenting a visual representation of a character, I wanted to pursue the idea of interactivity and create a dynamic 3D experience that capitalizes on the abilities of digital publishing. Rather than using traditional 2D representation, I created 3D scenes, containing models of the characters, where the user could control the camera like a Marmoset viewer—meaning they can zoom, pan, and rotate around the character model.

While the goal for a full shipped eBook product would be to include models of all major characters in the novel, I was again limited by time constraints and decided to only do three characters for this proof of concept. The author, Dorian, and I spent a while deciding on a selection of three characters that best represented a selection of different races, cultures, genders, and character roles in the novel. The three characters we landed on were Innili, the Gilifolk (human-shark hybrid) smuggler; Sami'oye, the ex-pirate-turned-mercenary from the Thalasseon Archipelago; and Regivaeld, the warrior king from Hybor. The choices also allowed me to design and model a variety of body and costume types to help me diversify my skill as a character designer and modeler.

The characters were taken through a full game-ready pipeline so that they could be easily displayed with Unity engine's real-time rendering. Each model began as a Zbrush sculpt, where I first completed the high-poly model. Regivaeld's armor pieces were a small exception as they were first built in Maya, and then brought in to Zbrush, which helped to keep them clean and simple. After sculpting the high poly model for

each character, I also used Zbrush's polypaint feature to create the color base for the materials.

Figure 18: Screenshot of Innli's high poly sculpt with polypaint in Zbrush

Once I had completed the high poly sculpting and polypainting, I retopologized and UV unwrapped each character with Maya's quad draw tool to make them low-poly enough to render in real time. Innili, who had the fewest costume elements, has the lowest polycount at approximately 24,000. Regivaeld's model was about 38,000 polygons. Sami'oye had a much more complicated environment, and also had quite a few small detail pieces. like her jewelry, which drove her polycount up significantly. She landed at about 81,000 and her environment was about 36,000, for a total of 117,000. Luckily, as the scenes were very simple, this polycount, though high, was still low enough for Unity's renderer to handle.

Figure 21: Regivaeld low poly wireframe

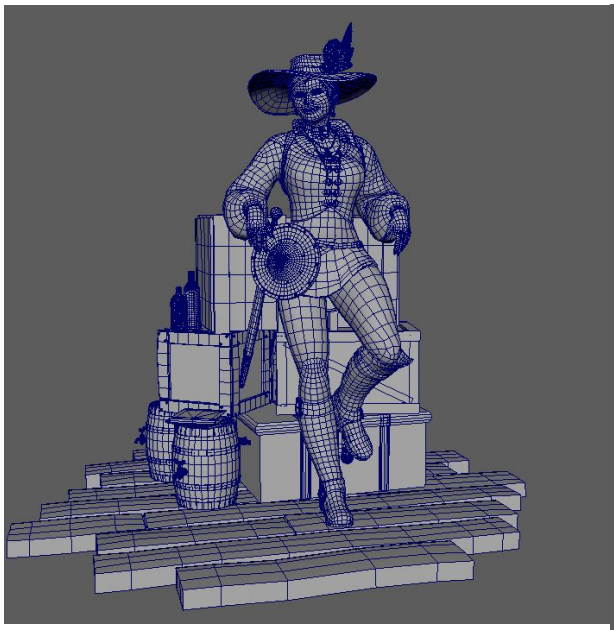


Figure 19: Sami low poly wireframe

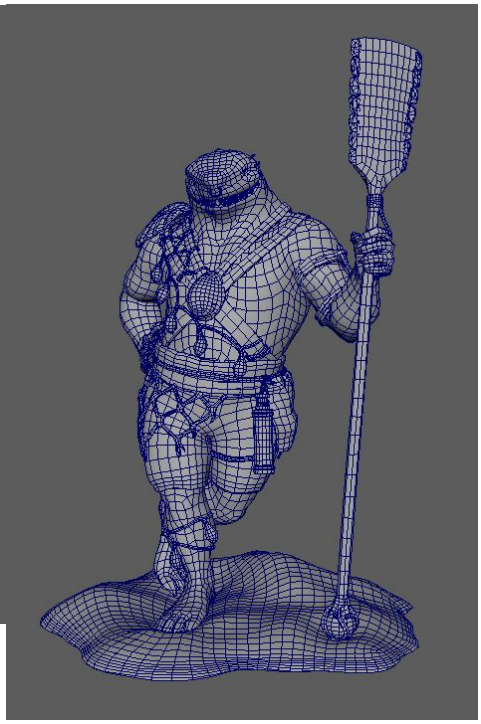


Figure 20: Innili low poly wireframe

After the retopologizing process, I used Substance Painter to bake the normal maps from the high poly model and vertex color from the polypaint. I found Substance's

built-in baking system to be the easiest to use because of its ability to match by mesh name. In addition to baking my maps in Substance, I also did quite a bit of material tweaking. All my materials were exported as a PBR Metallic Roughness profile. On many parts of each model, I added to or completely overwrote my polypaint data as I saw necessary. For Regivaeld's armor, I didn't do any polypainting and did all material work from start to finish in Substance Painter. The most difficult material for me to transfer from high to low poly was Regivaeld's chainmail. Even at its lowest possible poly count, including 3D geometry for each link of chainmail was over 400,000 polygons. As a result, I had to bake the information down to a one-sided mesh. In addition to baking the standard maps in Substance, I had to also create an alpha transparency map so that the holes in the chainmail mesh were visible. For this process, I tried a variety of methods, but the most successful method was using vertex color. By making the chainmail links one vertex color and the base mesh another, I created a black and white image. Then, using photoshop, I created an alpha channel on the albedo map file based on the baked vertex color map, which made the mesh transparent between the links.

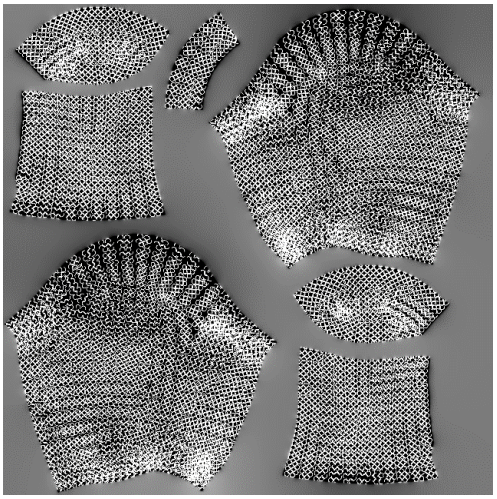


Figure 22: Baked vertex color map used to create alpha channel for chainmail

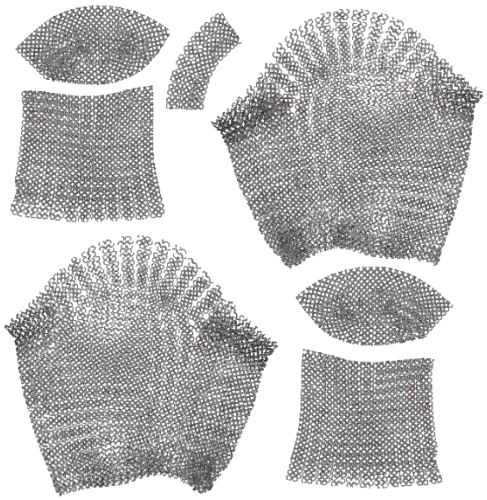


Figure 23: Chainmail albedo map with alpha channel from vertex color applied

Because Unity's lighting and rendering setup is drastically different from Substance Painter's, getting the materials to look correct in the final display required a lot of back and forth between Unity and Substance, re-exporting my textures a number of times to make sure things looked right in the Unity engine.

To give the characters all a bit of personality and story, I posed them and put them in small environments that gave context to their personalities and backstories. Due to my limited knowledge, the first two characters, Sami and Innili, were posed in Zbrush before they were taken through the rest of the pipeline for retopology and materials. However, with Regivaeld, I waited to pose him until after I retopologized him so that I could give him a basic rig. While this wasn't necessary for the scope of this project, as the characters are static, I wanted to be able to animate Regivaeld in the future.

Figure 26: Regivaeld model, rendered in Unity

Figure 25; Innili model, rendered in Unity

Figure 24: Sami'oye model, rendered in Unity

I also needed to create the platform in which to view the characters. My goal was to mimic the flexibility of user control a Marmoset viewer provides. To replicate this functionality, I needed to set up user controls to rotate, zoom, and pan the camera around the model. The camera system is set up as follows: an empty game object is placed in the scene, centered on the character model. The main camera is then placed at a suitable distance from the model and parented to the empty game object. The script I created to control the camera actually rotates and translates the empty game object, making the camera swivel around the character. I introduced limits on the vertical rotation to prevent the camera from looping around and reversing the controls. The zoom is handled by manipulating the camera's field of view rather than moving the camera closer or further from the character model. As the axis of rotation gets offset when moving the camera, I introduced a reset function that places the empty game object back into its original placement and reset's the camera's field of view to the starting point.

Figure 27: Character select screen

Because I wanted each character to be viewed separately on their own page, I needed to create a parent page where characters could be selected for viewing. The selection page is very simple in design, containing a single window with buttons that link to each character's viewer. I wanted to structure this scene based on character selection screens you often see in fighter games like Mortal Kombat, so each character's button is a portrait of that character with a nameplate telling the user who they are. However, instead of displaying the selected character adjacent to or above the character icons like you often see in games, I decided that isolating the character viewers from the rest of the page would give the added benefit of more viewing space for the character and their bio. As a result, each character viewing page opens as a separate scene after clicking the character's headshot. Once a character viewer is open, the only navigation option is to go back to the character select page, and from there you can navigate to the rest of the scenes.

Figure 28: Character viewer screen

In addition to the character viewer, I also wanted to include small biographies of each character, giving some relevant information about their culture and religion, age, societal position, and a brief summary of their backgrounds and personalities. I asked the author to consolidate this information for each character we chose. My task was figuring out how to display this information without detracting from the viewer function. Because I wanted to give the artistic assets as much screen space as possible, I decided to put the biographical information in a popup window, rather than have it permanently take up screen space. I created a button at the bottom center of the screen that would make the popup appear and disappear.

Originally, the button would set the game object containing all the bio popup information to either active or inactive. This would make the window simply “pop” in and out of existence. However, after putting this method into practice, the “popping” felt jarring and disconnected from the aged aesthetic, especially since the popup window is placed at an angle to mimic the look of a note on a piece of paper. To make the popup feel more like a note, I created an animation for the popup’s transition in and out of the viewable frame. The popup starts in an idle state outside of the viewable screen and clicking the bio button triggers an animation that rotates and translates the game object in to the camera’s view. Clicking the bio button again triggers a reverse animation, rotating the popup back outside the viewable area.

Figure 29: In-app screenshot of the Bio button and popup

Another issue with the character viewer was the lack of stylistic cohesion between the 3D, realistically textured characters and the 2D, ink and watercolor feel of the rest of the art assets in the application. To address this disconnect, I used two different methods that eased the transition between the 2D and 3D styles. The first method I used was a simple application of post-processing on the camera. Unity provides a free post-processing add on that can be downloaded from the Asset Store, which made the process easy to accomplish. Using this method, I created a post-processing profile that I added to the camera on each character viewer screen. Using this profile, I made small adjustments to the Tonemapping, and desaturated and lowered the contrast of the image. I also increased the graininess of the image, added a subtle vignette effect, and bumped up the red channel to imitate an old sepia photograph.

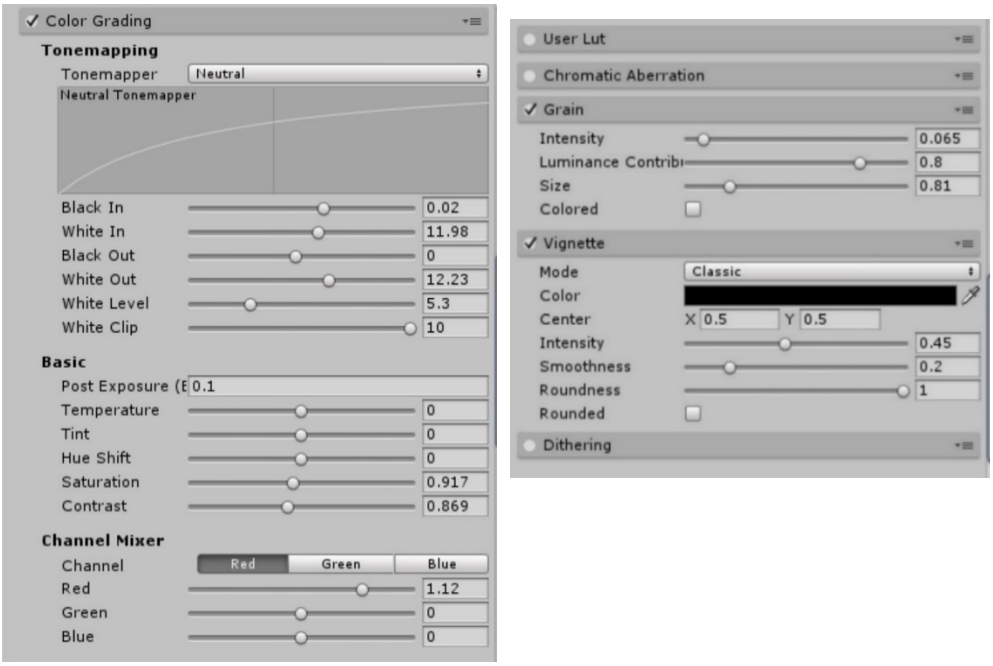


Figure 30: Post Processing profile settings in Unity

Figure 31: Before (left) and after (right) Post-Processing Profile is applied to camera

Figure 31 displays the default camera settings on the left, and what the scene looks like after applying the post-processing profile on the right.

In addition to addressing the lack of stylistic cohesion with the camera settings, I also created splash images for each character viewer that matched the ink and watercolor

style of illustration used in the rest of the app. I drew the splash images to match up perfectly with the scene's starting camera view, so that when the user opens the character viewer, the splash image slowly transitions into the 3D scene underneath (see Figs. 24 and 25). Because the illustrations and models needed to line up perfectly for the transition to work properly, I created a function in my camera movement script that freezes the user's controls until the 3-second long fade has completed. The camera reset function also triggers the splash image to reset, so the controls are frozen for 3 seconds after triggering the reset as well. Adding the splash images and the post-processing effect were both extremely useful steps in creating a sense of unity between the character models and the look and feel of the rest of the application.

Figure 32: Sami'oye splash image

Figure 33: Sami'oye 3D scene

The unlocking functionality was the last piece of the character page to be put in place. As it was dependent on tracking the reader's location in the reader, the unlocking script was one of the last parts of the technical functionality to be realized. However, once I had set up the ability to track the user's last read page, which I previously mentioned that I stored as a global variable, I simply had to create a script that checked the value of that variable when the scene loads and set each button's interactability based on that variable. I set each character to unlock once the user reaches the first page of their introduction chapter in the reader, except for Sami'oye, who is always unlocked because

she does not appear in the text excerpts. Because I used the “Last Read Page” variable, which never decreases or “goes backwards,” rather than the bookmark, which can be set to any page in the reader based on the user’s input, once the characters unlock they stay unlocked until the app restarts.

Overall, setting up the character models went more smoothly than any other piece of the project. Overall, I’m pleased with how this turned out and cannot think of any changes I would make in future iterations of this project. The only thing that would change is the number of characters, based on however many characters would be needed for the full model.

SECTION 5: User Interface Design

Though it ended up becoming one of the most essential pieces of the app, user interface design was not originally one of the focuses of my project. As I began progressing though the actual assembly of the application and its technical elements, I found that I underestimated the huge role it plays in the usability of a product and the amount of time it takes to create all the necessary assets. Due to my underestimation of its value, I did not begin working on the UI elements until more than halfway through the project.

The first step in designing the UI elements was deciding on a visual style for the app as a whole. As I mentioned previously, the map played a large role in informing the stylistic choices for the rest of the app. By combining my Age of Exploration inspired map style with the stereotypical fantasy trope of the old manuscript look, I decided to use desaturated “thin” color inspired by watercolor painting and lightly eroded linework on aged paper texture backgrounds. This helped me choose my neutral color pallete of browns, reds, and grays, and helped to emphasize the “bookish” feel of the app.

Figure 34: Color Palette for UI

After choosing the line style and color pallete, the next step was deciding how to design the buttons and other UI elements. The author and I sat down to talk about possible directions to take when it came to deciding shapes and designs to incorporate into the designs. We both felt that the majority of UI design in fantasy games was unappealing and not what we wanted in this product. With that as a guiding factor, I wanted to create simple designs that only used line and flat, loose color. The author and I spent a while looking at different inspirations for linework and shape language, and

decided on geometric-style linework based on a combination of Celtic design and Medieval illuminated manuscripts. I made proof sheets with a bunch of different button styles based on our discussion, and we decided on the design we liked the most from those sheets. That main button style served as the jumping off point for the rest of the designs, such as the marginalia, scrollbars, and other buttons and icons.

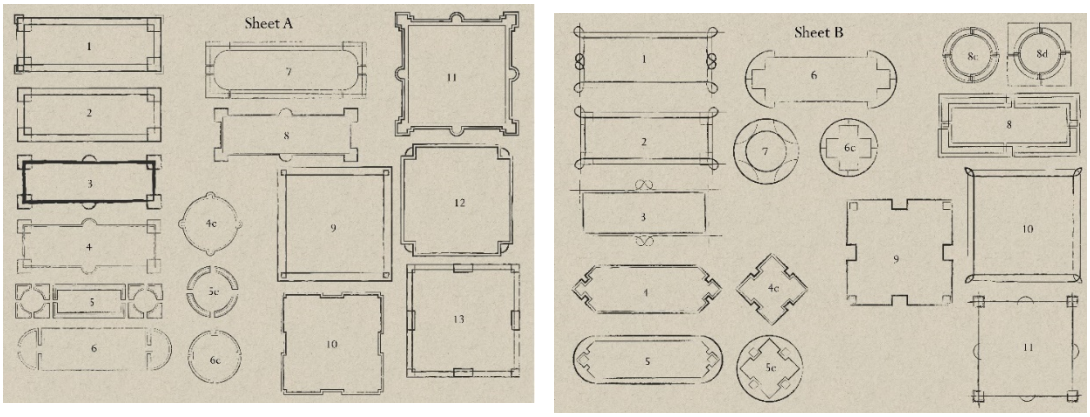


Figure 35: Original button proof sheets

Once I designed the interactable elements, I began working on creating the layouts for each page. The design for the main menu page was important, since it needed to set the tone for the rest of the app. However, because it didn't contain much functional content, there was a lot of empty space. The original concept felt somewhat plain and empty, but I found that creating a hand-drawn title and an animation of a caravel—the

main character spends much of the novel sailing around the world— helped flesh out the page and give it some interest.

Original Proof Final In-App Screenshot

Figure 36: Main Menu UI

The reader page, as I mentioned previously, was heavily based on the Kindle UI layout, so the placement of the buttons, scrollbar, etc. was not difficult for me to figure out. One small issue come up regarding the background. Meant to look like two sides of a book, I had originally used a paper texture and created gradients in photoshop to mimic the look of the spine. However, I found that using the gradient tool in photoshop created too “perfect” of a gradient, therefore making it look too computer generated. Since my look and feel tended towards an old book feel, it seemed out of place and jarring. Therefore, I hand painted the gradients with a texture brush to give it a more organic feel. Other than the gradient issue, the only major change from my original reader page design was the inclusion of the table of contents as a popup sidebar, rather than being its own separate scene. After building the app and navigating around it, I realized that an entirely separate scene for the Table of Contents would be overkill, and the screen space would be hard to fill. It also would be more jarring for the reader to completely navigate away from the reader to access the table of contents, so placing it inside the reader scene made using the Table of Contents easier. I had also never included any mockup in my original

layout for the bookmarking button. Adding the two buttons, one that triggered the Table of Contents window to animate into the scene and one that stored and cleared the bookmark, ended up providing a nice balance to the bottom bar that contained the scrollbar and location tracker.

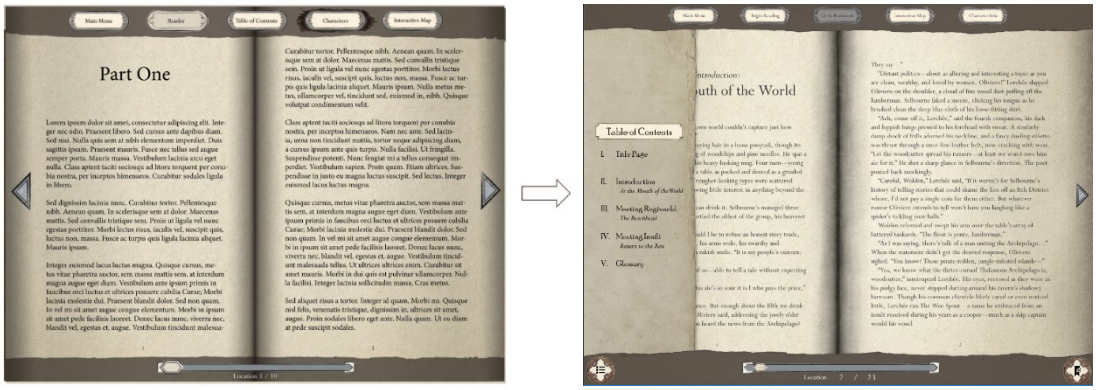


Figure 37: Reader page UI

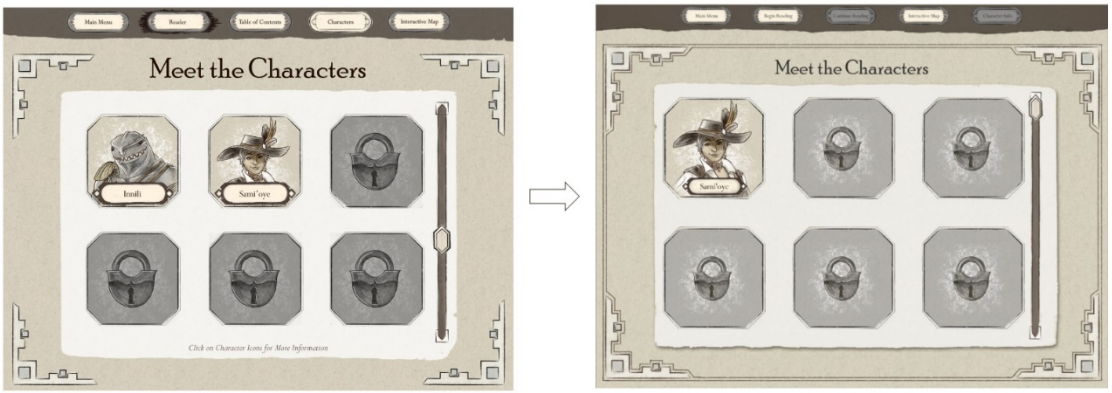
The design for the map page's UI takes cues from a combination of old and new cartography. The choice to use the ruler-style border and the designs of the name plate and toggle box were inspired by the borderwork on my Age of Exploration map references, while reflecting the shape language and linework style chosen for the existing UI elements. The location marker design is simply a slightly different take on the standard map marker that is used in Google Maps, providing a familiar point of reference for users to know what they mean. As I had already determined the main navigation bar would extend across the top of the reader page, and I wanted this to stay consistent through all of my scenes, the scene navigation button's locations were already determined. I altered the color layout from my original plan to make the map more similarly balanced to the other pages, as it was originally flipped. Otherwise, the basic layout of all my elements, such as the world title box, the toggle box, and the help button all stayed in the original planned locations.

Original Proof

Final In-App Screenshot

Figure 38: Map UI

The character select screen barely changed at all from the original concept. Its structure is simple: consisting only of the top navigation menu, the title and framing marginalia, and the scroll view window containing the icon buttons to select each character. Though I only included three characters in this version of the product, I wanted to include slots for many more characters to illustrate what this would look like when applied to a full product with all the main cast represented. This is why I chose to include many more icons than necessary and put them in a scrolling window. They are set up so that they are in order of when they unlock in the novel. The page layout, with lock icons that change into the character's portrait with a nameplate, is, like I previously mentioned,



Original Proof

Final In-App Screenshot

Figure 39: Character Select UI

based on character select screens from fighter-type games like Mortal Kombat or Super Smash Bros. This page was the simplest to design and execute in terms of UI.

The character viewer's changes were subtle, and the basic concept didn't change much from my original idea. However, as I learned more about designing UI and established a better understanding of my own style, a number of small aspects were changed. The most notable change was the location and orientation of the bio popup window, which I've already discussed to some extent with regards to the animation and skew. I also had originally made the window partially transparent but realized that transparency doesn't fit with the paper aesthetic. Another change, most easily noticeable on this page though it occurred in several other places as well, was the size of the buttons. Until designing UI, I had never considered the difference between scaling buttons for touch vs. mouse-based interfaces. After learning more about UI, I realized buttons for touch interfaces are significantly larger than those for mouse-based input. While I had originally thought to design this application for a touch interface, my time and skill limitations prevented me from tackling the technical side of touch input. Therefore, I realized I needed to shrink all of my buttons down, especially the buttons on this page.

Figure 40: Character Viewer UI

Issues like button scale and transparency had never crossed my mind prior to this project. However, there were many lessons I learned about designing UI that is informed by its function and creates a cohesive style. One other big lesson I learned about UI design was that the act of an asset shifting or changing on hover and click brought enough attention to that asset that the shift or change itself could be much subtler than I had originally thought. The first iterations of my button hover states, click states, and inactive states for my UI elements were extremely different in hue or value. However, I found that decreasing the differences between each state did not detract from their noticeability and looked better overall.

After completing the app, I had 6 people test the app's features to see if there were any glaring issues with the user interface, and to test whether users unfamiliar to the project were able to understand how it worked. Luckily, it seems most of the app's functionality posed no issues to any of the users. Most of the users mentioned two things that I was aware of but unable to fix: the non-intuitive navigation of the zoom and pan functionality of the map, and a bug in the character viewer script that causes the camera to get stuck. I had already created the camera reset with the spacebar as a workaround to the character camera bug, but was unable to fix the script to prevent the bug from occurring. As I have previously discussed, I was also unable to make the map camera move in a traditional click-and-drag style. This reinforced the need for these features to be fixed by a trained programmer in the completed version of the application.

One other comment I received from multiple testers was that they didn't notice or realize the purpose of the help buttons on the map and character page. To combat this, I set the help popup windows to be visible when the scene is first opened and then disappear after 5 seconds. This way, the directions of the controls are viewable when the page first opens, and the disappearing of the windows draws the user's attention to the buttons if they need to reopen them. Similarly, a few of the users were confused about the

Bookmark and Table of Contents buttons on the reader page, so I introduced similar labels to draw the user's attention to them.

On a positive note, most users said that other than the map navigation, the rest of the features of the app were easy to understand. I received one comment that presented the idea for including glossary popup windows that gave the definition of the highlighted word, similar Kindle's dictionary function. I thought this was a great idea, allowing the user to understand words without having to navigate to the glossary and back to their page. While I was not able to go through and do this for every glossary word, I did one on the first page to illustrate the concept. Almost every tester commented positively on the fade from drawing to 3D character model, and many complimented the aesthetic of the map. Overall, the user testing mostly reinforced my areas of concern, but did not present any major new issues.

CONCLUSION

The eBook market plays an important role in book consumption, and the percentage of digitally published books continues to rise every year. While there are many examples of interactive eBooks that take advantage of technology to improve the experience, they are largely targeted at children. I believe that, though untapped, the market for interactive eBooks among adult fantasy readers exists and should be targeted. This was the founding idea for this project and led me towards creating this prototype application for a new type of interactive eBook.

All in all, I think the project turned out to be a success as a proof of concept. I was able to produce a packaged application that includes all of the major assets I had planned on: a readable book, an interactive map, and unlocking character models. As I have already mentioned, I still believe there are many areas in which the application can be improved. Technical features such as the map's zoom and pan movement and character tracking function, as well as a more automated setup for the reader, are both features I think would need to be improved if this was to be completed as a full published application. I would also like to increase the number of characters, illustrations, and animations to make the artistic impact more powerful. While there are many aspects of the app that could be improved and expanded on, I feel that this prototype version demonstrates how digital publishing is capable of adding to the fantasy novel reading experience.

BIBLIOGRAPHY

"Ebooks - Statistic and Facts," *Statista*, 2018. https://www.statista.com/topics/1474/e-books/.

"EPUB 3.1 Overview." *International Digital Publishing Forum*. January, 5 2017. http://www.idpf.org/epub/31/spec/epub-overview.html

"Interactive Game of Thrones Map with Spoiler Control." Last accessed Aug. 29, 2018. http://quartermaester.info/.

Lazy Cat. "Pan and Zoom Orthographic Camera." Unity Asset Store. October 25, 2016. https://assetstore.unity.com/packages/tools/camera/pan-and-zoom-orthographic-camera-70398.

Magnus, Olaus. "Carta Marina." 1539. 1.7 m x 1.25 m. https://en.wikipedia.org/wiki/Carta_marina#/media/File:Carta_Marina.jpeg

McNeill, Sophie. "Five Key Trends in the Book Market." *Penguin Random House*. November 2015. http://authornews.penguinrandomhouse.com/five-key-book-market-trends/.

Ortelius, Abraham. "Islandia." ca. 1590. 19.5, in. x 13.5 in. https://commons.wikimedia.org/wiki/File:Abraham_Ortelius-Islandia-ca_1590.jpg

Rossby, H. Thomas and Miller, Peter. "Ocean Eddies in the 1539 Carta Marina by Olaus Magnus." *The Oceanography Society*. 2003. https://tos.org/oceanography/assets/docs/16-4_rossby.pdf.

Sanderson, Brandon. *Oathbringer*. New York: Tor Books, 2017.

Springer, Robert. "The State of Ebooks 2017." *EContent Magazine*. January 18, 2017. http://www.econtentmag.com/Articles/Editorial/Feature/The-State-of-Ebooks-2017-115709.htm

Thynell, Ulla. "Map of Midgard," *DeviantArt*. 2014. https://www.deviantart.com/ullakko/art/Map-of-Midgard-455740603.

Tiphs, "Commission – Hibendrill Worldmap." *DeviantArt.* 2015. https://www.deviantart.com/tiphs/art/Commission-Hibendrill-worldmap-567712816.

Transworld Books. "The Magic of Reality for iPad by Richard Dawkins – trailer." *Youtube*. September 23, 2011. https://www.youtube.com/watch?v=eBrP3-Ep3ww.

Voxel Frog. "RTS Camera Pro." Unity Asset Store. September 22, 2016. https://assetstore.unity.com/packages/tools/camera/rts-camera-pro-10390.