Development of a Tactics RPG for Android Tablets

A Major Qualifying Project Report:

Submitted to the faculty of the

Worcester Polytechnic Institute

in partial fulfillment of the requirements for the

Degree of Bachelor Science

By:

_____

Dillon Lankenau
Interactive Media and Game Development (Art)

_____

Gregory Tersoff
Computer Science, Interactive Media and Game Development (Tech)

Date: Thursday, May 1, 2014

Approved:

_____

Professor David C. Brown, Computer Science, Major Advisor

_____

Professor Britton Snyder, Interactive Media and Game Development (Art), Major Advisor

Abstract

Our project consisted of creating a Tactics Role-Playing Game to be played on Android devices. This entailed creating our own rule set, game engine, menu interface, art assets, and game instance. In particular, there was an increased focus on optimal use of touch interfaces, good interface design, and potential for use by non-programmers. Supplementing our game, we also developed 3d printed models to act as building blocks to construct a board in physical space.

Acknowledgements:

We would like to thank Professor Britton Snyder and Professor David Brown for their continued advice and assistance throughout our project. We'd also like to thank Professor Chrysanthe Demetry for her assistance during the initial stages of our project. Finally, we'd like to thank Daniel Irizarry for allowing us to use his tablet during early testing, the numerous freshman we bribed with cookies to take our surveys, and our usability test and playtest volunteers, who helped even without the offer of baked goods.

Table of Authorship

Section Name	Author	Reviewer
Introduction	Dillon Lankenau	Gregory Tersoff
Summary	Dillon Lankenau	Gregory Tersoff
Our RPG	Dillon Lankenau	Gregory Tersoff
Aim of Application	Gregory Tersoff	Dillon Lankenau

Background Information	Gregory Tersoff	Dillon Lankenau
Conventions for Tabletop RPGs	Dillon Lankenau	Gregory Tersoff
Conventions for Tactics RPGs	Gregory Tersoff	Dillon Lankenau
Touch Screen/Mobile Device Conventions for Games	Gregory Tersoff	Dillon Lankenau
Game Design/Balance	Gregory Tersoff	Dillon Lankenau
Application Usability	Gregory Tersoff	Dillon Lankenau
Intuitive Use	Gregory Tersoff	Dillon Lankenau
Efficient Layout Design	Gregory Tersoff	Dillon Lankenau
Balance of Potential Simplicity	Gregory Tersoff	Dillon Lankenau
Development Tools	Gregory Tersoff	Dillon Lankenau
Art Background	Dillon Lankenau	Gregory Tersoff

Methodology	Gregory Tersoff	Dillon Lankenau
Study of Game Design	Gregory Tersoff	Dillon Lankenau
Gameplay Design	Gregory Tersoff	Dillon Lankenau
Artistic Techniques	Dillon Lankenau	Gregory Tersoff
Survey Techniques	Gregory Tersoff	Dillon Lankenau
User/Task Analysis	Gregory Tersoff	Dillon Lankenau
User Profile	Gregory Tersoff	Dillon Lankenau
Task Analysis	Gregory Tersoff	Dillon Lankenau
Evaluation of Prototypes	Gregory Tersoff	Dillon Lankenau
Usability Test	Gregory Tersoff	Dillon Lankenau
Playtests	Gregory Tersoff	Dillon Lankenau

Game and Application Design	Gregory Tersoff	Dillon Lankenau
User Interface Design	Gregory Tersoff	Dillon Lankenau
Menu Design	Gregory Tersoff	Dillon Lankenau
Rule Set	Gregory Tersoff	Dillon Lankenau

Implementation	Gregory Tersoff	Dillon Lankenau
Development of Application	Gregory Tersoff	Dillon Lankenau
Code Structure	Gregory Tersoff	Dillon Lankenau
Interface Design	Gregory Tersoff	Dillon Lankenau
Development of Assets	Gregory Tersoff	Dillon Lankenau
Game Environment	Dillon Lankenau	Gregory Tersoff
Game Icons	Dillon Lankenau	Gregory Tersoff
3D Models and Printing	Dillon Lankenau	Gregory Tersoff
Constructing an Instance	Gregory Tersoff	Dillon Lankenau
Guiding Principles	Gregory Tersoff	Dillon Lankenau
Difficult Curve/Pacing	Gregory Tersoff	Dillon Lankenau

Evaluation	Gregory Tersoff	Dillon Lankenau
Usability Tests	Gregory Tersoff	Dillon Lankenau
Interpretation of Data	Gregory Tersoff	Dillon Lankenau
Post-Experiment Comments	Gregory Tersoff	Dillon Lankenau
Adjustments	Gregory Tersoff	Dillon Lankenau
Changes to Application/Instance Design	Gregory Tersoff	Dillon Lankenau
Playtests to Determine Effectiveness	Gregory Tersoff	Dillon Lankenau
Format of the Playtest	Gregory Tersoff	Dillon Lankenau
Specific Responses from Playtest	Gregory Tersoff	Dillon Lankenau
Adjustments Based on Responses	Gregory Tersoff	Dillon Lankenau
Future Adjustments Beyond the Immediate Scope of the Project	Gregory Tersoff	Dillon Lankenau

Results	Gregory Tersoff	Dillon Lankenau
Usability Evaluation	Gregory Tersoff	Dillon Lankenau
Asset Evaluation	Dillon Lankenau	Gregory Tersoff
Gameplay Evaluation	Gregory Tersoff	Dillon Lankenau
Conclusion	Gregory Tersoff	Dillon Lankenau

Table of Contents

Abstract ……..2
Acknowledgements ……...3
Table of Authorship ……...4
1. Introduction ……8
1.1. Summary ……8
1.1.1. Our RPG ……9
1.1.2. Aim of Application ……9
2. Background Information ……11
2.1.Conventions of Tabletop RPGs ……11
2.2.Conventions of Tactics RPGs. ……..13
2.3.Touch Screen/Mobile Device Conventions for Games ……14
2.4. Game Design/Balance ……14
2.5. Application Usability ……..15
2.5.1. Intuitive Use ……15
2.5.2. Efficient Layout Design ……..15
2.5.3. Balance of Potential/Simplicity ……16
2.5.4. Development Tools ……..16
2.6. Art Background ……..16
3. Methodology: ……..18
3.1. Study of Game Design ……..18
3.1.1. Gameplay Design ……18
3.1.2. Artistic Techniques ……19
3.2. Survey Techniques ……..23
3.3. User/Task Analysis ……..26
3.3.1. User Profile ……..26
3.3.2. Task Analysis ……...27
3.4. Evaluation of Prototypes ……..38
3.4.1. Usability Test ……...38
3.4.2. Playtests ……...39
4. Game and Application Design ……40
4.1. User Interface Design ……..40
4.1.1. Menu Design ……40
4.2. Rule Set ……44
5. Implementation: ……...46
5.1. Development of Application ……46

5.1.1. Code Structure ……46
5.1.2. Interface Design ……...50
5.2. Development of Assets ……52
5.2.1. Game Environment ……..59
5.2.2. Game Icons ……..64
5.2.3. 3D Models and Printing ……...68
5.3. Constructing an Instance ……..71
5.3.1. Guiding Principles ……...71
5.3.2. Difficulty Curve/Pacing ……...71
6. Evaluation: ……...74
6.1. Usability Tests ……74
6.1.1. Interpretation of Data ……74
6.1.2. Post-Experiment Comments ……...77
6.1.3. Adjustments ……78
6.2. Playtests to Determine Effectiveness ……...78
6.2.1. Format of the Playtest ……79
6.2.2. Specific Responses from Playtest ……...79
6.2.3. Adjustments Based on Responses ……...80
6.2.4. Future Adjustments Beyond Immediate Scope of Project ……..80
7. Results: ……81
7.1. Usability Evaluations ……...81
7.2. Asset Evaluation ……..81
7.3. Gameplay Evaluation ……...82
7.4. Conclusion ……...82
8. References: ……..83
9. Appendices: ……84

1 Introduction

Video games have steadily become a greater part of our culture. They can act as casual entertainment, educational tools, and are sometimes even played professionally. As a new medium, however, video games still have a long ways to go before they are as widespread or accepted as movies and novels, and are often relegated to "children's games." Games are often social activities, bringing a group of people together to act towards a specific goal while inspiring competition and presenting fantastical stories for players to experience.

With a wide variety of games being released for a growing number of devices, video game developers are constantly trying to come up with new and better control systems. Because there is no agreed upon convention, especially in terms of a games aesthetics, it's important that the control systems are as clear and understandable as possible, relating complex game mechanics in the simplest ways possible.

These games also serve as an innovative platform for combining traditional narratives with visual storytelling, making something like an interactive movie or a "choose your own adventure" novel. Games can function in the same ways that visual novels, or comic books, function by using traditional rhetorical theory with visual illustrations to convey something. They can also be purely mechanically driven, using a minimum of aesthetics to deliver a gameplay experience to the player. Many modern games are full of artwork, using everything from traditional paintings and illustrations to pictures used as particle effects for magical spells or explosions.

Video games can also be used to introduce classical pen-and-paper games, such as Dungeons and Dragons, to new audiences. Such games generally have a complicated set of rules and have a steep learning curve for new players, which caused them to dwell in relative obscurity. With the introduction of videogames, however, those pen-and-paper games have been re-created with all rules handled by the program, allowing new users to more easily figure out what is and is not allowed in those games, at the same time acting as a platform for user content.

The purpose of our MQP was to create a video game that could serve as a platform for future user created narratives. Combining the rules and gameplay of traditional tabletop games with the accessibility of a mobile touchscreen device was how we hoped to maintain the social interactions between players, and we wanted to further engage players with an interesting narrative and game world.

1.1 Summary:

Our project created a platform for interactive narratives, something that players could create their own stories for, with a game and its assets already created to serve as a demo version of what the game could become. In addition, we wanted to develop an application that allows new users to let their game be managed by the program itself, *or* allows for a player to control the game.

The application was developed to suit players who were new to the medium of video games, where some functionality that might be useful to experienced users could be downplayed in some game playing sessions. We believed that making a customizable Role Playing Game (RPG) would allow us to teach, and encourage interest in, these schools of thoughts more effectively than currently existing games. We also believed this would allow us to develop an

interesting game, by creating a relatively new experience for fans of computer games without losing as much of the interactivity and longevity that many game developers do away with.

The artistic direction of the final game was designed to convey a fantasy setting to players. The game required several core components to suit this goal. It needed a game environment, represented by a tile-set, character icons, and portraits/paintings. The game also needed user interface artwork, specifically actions/skills icons to relate gameplay mechanics, as well as things like menu's and status display elements.

The game that we produced is a Tactics RPG with fantasy elements. It follows the adventures of four characters searching for the lost city of utopia. In the story they are hired to verify claims that the city actually exists, and through their adventures must leave the city via underground tunnels, journey through the forest full of various monsters and animals, and eventually arrive at location specified to find a temple full of enemies and monsters. Each one of the characters is meant to serve a different purpose, and to be controlled by different players. Another player is meant to coordinate all of the enemies and monsters, working to both defeat and engage the other players in the gameplay.

1.1.1 Our RPG

Our game best fits into the genre of Tactics RPG. The game handles the movement and combat of characters and monsters, as well as host an environment that players can explore.

While an element like movement can be straightforward combat can be very complicated. Each of the main characters is unique, and has their own special abilities to make the play style associated with the role they play within the game. For example the barbarian is a very physical and close-combat character while the mage works better by using skills from afar. Dynamics like this drove our design of how the world works and looks.

Our project was originally developed to teach players about philosophical and scientific concepts through game play and story progression. We wanted the game mechanics to show how things like gravity or thermodynamics worked, while the story was going to introduce classical philosophy in the form of exposition. Over the course of the project our focus shifted to be more modular, allowing users the ability to create new narratives, art assets, and rule-sets to invent their own game play experiences.

The project added another element when we began seriously considering the physical interaction portion of the game. Since all the players were sharing a tablet and talking with each other, the game needed a way to engage the players in the game even when they weren't actively performing an action in the application.

1.1.2 Aim of Application

In addition to our goals of creating an enjoyable game, we also set out to create a highly functional game engine. Our game engine is designed to run on Android personal tablets, with an interface designed specifically for touch screens. While there are many games in the same genre as ours which are available for Android OS, many of these games make poor use of touch screen controls, as their layouts are intended for other input systems. Because our game is specifically designed to make full use of touch screens, the control scheme is set up to be easier and faster for a

user to use with practice, while still being as intuitive and easy to learn as interfaces designed for video game consoles and PCs.

2 Background

This section explains concepts and conventions relevant to our game. This provides context for many of the ideas that are brought up when explaining our development process. In particular, we will explain what some of the common expectations of our game will be amongst gamers and designers, so as to better address those expectations when explaining our thought process.

2.1 Conventions of Tabletop RPGs

Tabletop Role Playing Game (RPG) fall into the genre of interactive media, much like board or video games, where the player personifies a character inside of an imagined world to experience a story and complete some objective.

A player of a tabletop RPG is in charge of their own character, often through a "character sheet." A character is described with a set of statistics to explain their physical and mental capacities, and a list of items or skills available to the player through their adventure. Players often have a wide variety of in-game actions available to them while playing through the game session, and are the driving factor behind the game-world narrative.

A Game-World is the pervasive setting and mechanics of the RPG experience. It could be a fantasy style landscape where creatures like Orcs or Dragons exist, or it can be a futuristic world where robots and space travel is possible. The Game-World is also based in the rule-set, which describes game mechanics such as physical statistics of entities, non-player character (NPC) interactions, and specific rules about solving puzzles or completing actions. The rule-set can vary in detail, being so intricate as to describe the exact method by which an NPC moves from one side of the room to another via special rules, or be abstract enough to allow a non-descript NPC to interact in the game-world without needing special rules to cover their every action.

Dungeon Masters (DM) act as the arbiters of the game world; they interpret the rules and the story to create an adventure for the players. In a tabletop game the DM's are responsible for guiding the players through an "instance," which is an open-ended narrative experience. These instances could be an adventure where the DM fills in the placement and relative strengths of the monsters, or describing deadly puzzles in the game-world, or the DM could personify an NPC to talk to the players and give them information about their quest.

Role Playing Games often follow common storytelling tactics, like the hero's journey or a fairy tale. Using a widely known narrative technique allows the Dungeon Master an environment they can personalize their players experience. These stories make up the narrative, or the general flow by which the overall plot of the adventure progresses. Many adventure "instances" are

included with the game manuals to get the players started, but DM's can also substitute their own narratives or make them up dependent on how their players interact with the game.

Interactive storytelling has the potential to be very complicated due to the sheer variety of actions a player could theoretically perform. A player could attack someone with a sword, steal and hide jewelry using sleight-of-hand, or (in extreme cases) attempt to use a lightning spell on specially-crafted metal rods to power a rail-gun. In order to bring the number of actions down to a reasonable scale, rule-sets are designed to clarify the actions that players can take in the game-world. Although the players have discretion about their own actions the DM is actually in charge of guiding the story and giving choices to the players about the general flow of the plot. Interactive narratives are often driven by what the game-world and tone of the story can allow, and is in some ways dependent on the gameplay mechanics to drive the plot forward.

Gameplay mechanics, such as the actions that a player can take or the instance that the DM constructs in the game-world, are based on the rule-sets. Within those rule-sets are the guiding rules for the game-world, such as the physics of gravity or the actual movement of energy. Many RPG's expand on supernatural themes that have no basis in reality, and their rules have to be clearly defined before the players can interact with them. The DM also has to understand the basic rules of the world, in order to better describe the puzzles, traps, or monsters that they might use in their stories.

Tabletop RPG's are similar to choose-your-own-adventure novels in that the player works within a writer's framework for the game, except players have an untold number of choices available to them. The dungeon master simply lays out a scenario, describes the suggested goal or purpose of the game-play session, and then allows the players to choose where they want to go. If a player were to walk into an area of the game-world, it would be the dungeon masters job to detail what the setting is, whether there are any non-player characters of enemies, and then step out of the way and let the player interact with the world. When dealing with non-player characters the dungeon master would actually personify the character, and interact with the players acting in the role of a bitter mercenary or a noble king.

Players who are engaged in an interactive storytelling experience often do not actually question the direction and development of the story. They are too busy enjoying the narrative and the game-play to try and change their experience. When a game narrative is done correctly, a group of players can be so interested in a horror adventure they completely forget where they are, and forget the fact that they are not in any danger whatsoever, leading to genuine feelings of fear.

Inside the game mechanics there is also a balance between a challenge being too hard for the players to overcome, and a challenge being too easy to be worth the player's time. In a narrative this can mean that at the beginning of an instance the players would expect to be

introduced to the story and characters involved, in addition to learning how the rule-set functions in the game-world. Around the middle of the story the players might expect more of a challenge, now that they've gained an understanding of how to play and why they're doing certain actions. Near the end of the "instance" the players should see the end of the story, the climax of the narrative, and should be able to use their characters to overcome challenges that at the beginning of the game would seem impossible.

The purpose of a tabletop RPG is to allow the players the ability to interact with the game world, and play a role in exactly what's happening. This can be fun, and it can cause the players to invest in their gameplay experience and characters. People play games to be entertained and to gain rewards for their progress, normally characterized in many different types of incentives. Making their in game character stronger, for example, or acquiring a better item for their next played "instance" is a strong incentive to win the game. Other players might be more interested in the narrative, and would be interested in the outcome of their game session's world.

2.2 Conventions of Tactics RPGs

In contrast to Tabletop RPGs, Tactics RPGs tend to function primarily as a game, with story a secondary concern with very little influence on mechanics. Tactics RPGs use clearly defined and inflexible rules and have a much stronger emphasis on combat than diplomacy or role-play, focusing on the "challenge" aspect rather than the "narrative" aspect of games. While Tactics RPGs and Tabletop RPGs have many similarities between their core combat rules, the two games take those rules in very different directions.

Character representation in Tactics RPGs tends to vary from game to game, but rarely takes on the role-play aspect of Tabletop RPGs. For single-player Tactics RPGs, players generally control a number of characters, with small team sizes averaging at 4-6 and large team sizes approaching 10-20. The size of a team is usually inversely proportional to the types of actions a character can take. A Tactics RPG with 4 characters playable at a time is likely to allow each character to have access to a large number of special abilities, with a gameplay emphasis on choosing the right action for a given situation. In comparison, a game with 20 characters per battle will generally give each unit a default attack with one or two special abilities, and will focus on group movement and positional advantages. If a player is ever given a specific character to act as a representation in-game, this character is generally a pre-defined leader or an emotionally-neutral tactician, both cases giving the player a reason why they have control over their side without having to rely on players themselves to drive the story.

Tactics RPGs are generally single-player or competitive; they are very rarely co-operative unless designed to be team vs. team. This means there is less of the adaptive gameplay that a Dungeon Master provides. All actions the characters can take both in and out of battle are clearly

defined, with little to no room for interpretation. As a result, a player's goal is generally more focused on completing a challenge than advancing a story or exploring an imagined world.

With a stronger focus on challenge completion, Tactics RPGs are given a specific level of intended challenge, and games rarely make themselves easier based on difficulties a player has been having. While the pattern of changes in gameplay difficulty over the course of the game may be similar to Tabletop RPGs, the difficulty is rarely adjusted based on player performance, unless certain "bonus" objectives allow the player to make the game easier or harder as a reward.

While combat in tactics may be a more simplified version of that of most Tabletop RPGs, this is largely done to add consistency to battle. The goal of a Tactics RPG is to give a player a challenging scenario, and to require that player to come up with an intelligent solution to that scenario. While Tabletop RPG players are driven by a desire for progression and acting out other lives, Tactics RPG players are driven by the feelings of success that come with completing a challenge.

2.3 Touch Screen/Mobile Device Conventions for Games

Many games have been developed for touch screens and mobile devices, and a number of expectations and conventions have been established. One of the most obvious expectations is that any application specifically for mobile devices should not require a user to use any input aside from the touch screen. Every task should be performable using only the basic inputs of taps, slides, and drags that most mobile devices use to navigate the OS' menu system.

Games on mobile devices also have specific conventions for how these inputs are used. When players are trying to view an image larger than the screen, most games allow players to adjust which portion of the image they can see by dragging the image to a specific point, or swiping the image to cover a large distance. If a user accidentally selects the wrong button, but has not released the button yet, they can drag off of the button before release to avoid selecting it.

2.4 Game Design/Balance

When designing a game with any kind of tactical combat, it is important to make sure that one single tactic does not work in every situation. If the same tactic can be applied in every battle with a near guaranteed success rate, this greatly detracts from the challenge the game provides, and winning a battle is no longer a satisfying experience. Likewise, if every tactic works at a reasonably intelligent level of execution, than a player is no longer rewarded for having come across an intelligent tactic, which reduces the satisfaction a player feels. In order to make sure a player is satisfied with winning a game with tactical combat, a few key goals should be kept in mind:

- A single tactic should only work consistently if that tactic is very well executed, thus rewarding a player for intelligent use of a preferred tactic.

- Different tactics should have different rates of success in different battles. This rewards a player for analyzing a situation and determining the best course of action.
- A player should have to think tactically, with ideal play only becoming obvious once the opponent has made their move. If a player can decide on a plan at the start of a battle, they should need to make some adaptations as the battle continues. No strategy should be so good that consistently poor choice of actions will still result in victory.

If these guidelines are not met, it becomes very difficult to make a game with satisfying level of difficulty. While there are many ways to make a game too difficult, what counts as *too* difficult varies from audience to audience. If a game *is* too difficult, designers most likely need to allow for more error in making optimal play, or make a greater number of tactics viable. See Section 5.3 for an example of the balance guidelines we used when developing our game.

2.5 Application Usability

Because Tabletop RPGs frequently involve a large number of rules with complex interactions, along with extensive planning and organizing of the scenarios used in an instance, the ability to have a program handle calculations and information storage would greatly aid a dungeon master. However, if the program is poorly designed, the difficulty of use would offset any benefit from easier calculations or a stored map. This is especially true for new users, who could become more easily confused by a poor layout without any understanding of the game to provide context. As such, these elements of design will be focused on, in order to make sure there is a net gain from using the application. Section 6.1 has examples of these metrics being used to evaluate an interface.

2.5.1 Intuitive Use

A core asset to any interface is how intuitive it is to work with. An application that allows for any course of action to be taken is useless if users are unable to figure out how to get that application to work. Having a clear design that follows the expectation of a user greatly aids in usability. Creating an interface that is intuitive to use can be done through several means: designing the interface to be similar to programs that perform a similar task, making sure that symbol and word associations are clear to the user, and having tasks be presented and/or performed in an order that would mimic similar tasks in the real world. All of these factors can make it easier for a user to understand and expect how different elements of the interface will work.

2.5.2 Efficient Layout Design

While an intuitive interface may be easy to understand, it can still be frustrating to operate. An efficient layout design allows users to quickly and consistently perform the tasks they need to do, once they have established what their task is. Efficiency can come in many forms, such as

reducing the number of actions or steps taken to perform a frequent task, helping the user search for tools by grouping similar ones together, contrasting different features with different colors or brightness to help the user visually separate tools, and having buttons or menus be appropriately sized and spaced to reduce selection error and speed up selection time. How efficient an interface is can be measured using multiple metrics, such as the average time it takes to perform a given task, and how likely the user is to select an intended tool or option. In each case, the goal is to allow the user to move through the program faster.

2.5.3 Balance of Potential/Simplicity

No matter how perfectly designed an interface may seem, there is guaranteed to be a potential change that would aid a specific type of user compared to an "average" user. While someone who wishes to show large amounts of information on the screen at any given moment might prefer small text, a user with a visual impairment may want a larger font size, or even sound files as an alternative. Likewise, an interface that immediately presents every tool to the user may be confusing for a beginner, who will be unsure of which tool is the most appropriate, while at the same time making the process easier for an expert user. Knowing what is expected of your user can allow you to make adjustments as needed, and optimize an interface for those who you believe are going to be using it.

2.5.4 Development Tools

While generating useful interfaces can be challenging, there exist many tools to help application developers create Graphic User Interfaces (GUIs). Many of these, such as the Java Swing library (http://docs.oracle.com/javase/6/docs/technotes/guides/swing/), are designed to allow developers to add GUI elements with tools without having to write out code that draws buttons pixel by pixel, detect mouse clicks, or generate windows. These allow the developer to focus on design instead of code.

2.6 Artistic Background

The very first inspiration of our project came from the webcomic Dresden Codak, in the form of a dungeons and dragons style RPG titled "dungeons and discourse," which used concepts and names from real world science and philosophy to populate the fictional game world. We wanted to engender this rule set and take it from a comic narrative to a complete game.

The game Dungeons and Dragons is a prototypical role playing game. It blends together a concrete set of rules with a fluid style of narrative. From our experiences playing Dungeons and Dragons we knew that the game is capable of having many different stories told in it, and that players could have a direct impact on the progression of the story. This was important to our project because allowing players to take control of their own experience makes for a fun and engaging gameplay session, which would make people want to continue playing the game.

Dungeons and dragons- DnD exists as the quintessential role playing experience. It blends together flowing narrative into the very hardest game rules. If a dungeon master wishes they can ignore components of the game altogether if only to create a more engaging story for their players. Our game is taking this framework, specifically their system of controlling and interacting with the players characters in game, and adding an interface which makes it easier to envision and interact with the game world.

Deus ex- The core component of Deus ex is that it's a video game with a very high level of engagement for players. If they want to do something, and they have the resources to do it, they can. Anything from sneakily bypassing giant robots to openly fighting armed militia is possible, and we want to take that feeling of interactivity, which Deus ex derives from the dungeons and dragons format, to apply it to an easily accessible and fun game.

Warhammer 40K- The Warhammer universe is full of narrative, blending a neo-apocalyptic style with a rich lore and an even more extensive thematic elements. There are gods and titans, space ships, ancient horrors and brutal monsters, all of which combines neatly into an ever growing story that drives gameplay across the many different iterations of video games. There are first person shooters that focus on the individual soldiers, massive strategy games that force the player to act as a tactician, and more straightforward role playing games that create a more structured narrative existing in the diverse and interesting game world. The story that we're trying to accomplish takes elements from Warhammer in that players are not "chosen ones" sent to accomplish a certain goal. Our characters are surviving as heroes and villains, and could choose to play the game as pacifists if they want. There is no concrete narrative, and to the extent of storytelling there doesn't need to be an end to the adventure.

Fantasy- The impact of a non-realistic world on a story is not something to be easily overlooked. The story could influence the gameplay as much as it could the player's engagement with the game world. Fantasy functions as a sort of blanket term, but for our game it means that there is a supernatural element to the storytelling. Elves, Dwarves, Androids, magic, etc… are commonplace beings in our game.

The original purpose of a blend between fantasy and science fiction elements in the game world was to incite players to think more deeply about complex topics like ethics and science, but over the course of the project these elements serve more of a contextual purpose than an educational one.

From just a few references we now know the tone of the game environment, the core mechanics that will drive gameplay, and the fundamental relationships between the major characters the player will interact with.

3. Methodology

3.1 Study of Game Design

Once we had decided on the type of game we would make, we decided to look at other games in the Tabletop RPG and Tactics RPG genres in order to establish a framework for our game. By studying other games of a similar genre to ours, we could analyze the pros and cons of popular conventions in order to make a more informed decision about what should or shouldn't be included in our game.

Much of our studies were based on past experiences, as both project members are avid fans of the genres our game resembles. Thus, we were able to look at Tabletop RPG rule sets like Dungeons & Dragons, and Tactics RPGs like *Final Fantasy Tactics*/*Final Fantasy Tactics Advance*, *Disgaea*, *Fire Emblem*, and many more.

3.1.1 Gameplay Design

One of our early decisions was to base our rule-set around many of the conventions of various Tabletop RPGs, in particular Dungeons and Dragons. Thus, we looked at various popular rule-sets in order to determine what was effective. At the same time, we looked at popular Tactics RPGs, since Tactics RPGs are usually videogames and thus would better match the medium we would be using in our final game, in order to look for overlap.

When creating a basic rule-set, we immediately decided on the basic actions of moving, attacking, and using skills, which exist in some form in almost all examples of both games. While we decided on what kinds of skills we would give to characters, we instead looked at what out-of-combat rules we would include. We decided to include rules about what a player would have to do in an out-of-combat challenge, designed to be adaptable to numerous forms of challenges, similar to the skill checks found in Dungeons & Dragons. From here, we started to branch out into more specific rules.

After deciding on the very basics of in- and out-of-combat rules, we then decided on rules for representing our characters' strengths and weaknesses. We decided to use a basic "stats" (statistics) system, in which most of a character's success rates and abilities are calculated based on a small number of "base stats" representing physical and mental abilities. Once base stats were established, we decided to have most calculations in game be based on a d20 system (a system where any chance is calculated by rolling a 20 sided die, and adding any bonuses to the result of the die roll and comparing the sum to the difficulty of the task) in order to allow for easily-calculated success rates when trying to use the game rules without a device. See Appendix A for an early implementation of our rules intended for pen-and-paper play.

3.1.2 Artistic Design

Video game artwork is meant to be functional as well as aesthetically pleasing. There are a lot of individual pieces that need to go into a game, from the user interface to the game environment.
The tile-set is an asset that doesn't fit within any of the major classifications of the game art. Its part environment, part user interface, and one of the most important pieces of the overall game experience.

The environment assets can be different depending on the type of game that's being made. For a 3D game maps populated with meshes and texture sheets can be used to create the game world. For text adventure games sometimes a map made of ascii graphics can serve to illustrate the world of the game. For a truly 2D game, that is one where each art asset is taken from a static image and rendered in the game, the environment can be constructed from many individual "tiles" which can then be constructed into a large image that serves as the game world.
For our game the tile-set is made of several iterations of simple concepts such as floor tiles, stairs, and water. These can be further identified as being part of a stone dungeon tile-set, where there are several different version of a floor tile that can be used to enhance the games visual appeal.
The user interface is the parts of the game that the user can directly influence. Artistically this means that each button, text box, and transition screen needs to be visually accessible.
The aesthetic details compliment the entire game. They're meant to hold the design together visually, and to entice the players to immerse themselves in the game. These are largely secondary components of the overall game, as they aren't needed to actually experience the gameplay.

Once the first round of physically drawn concepts were completed it was time to actually refine the character portraits into a usable form. The software used for these images is adobe Photoshop, coupled with a Wacom tablet. Some artists use scanners to take their work into an editing suite, but this project it seemed easier to redraw the concepts digitally than it was to scan and then "clean" the images to be ready for digital painting.
The process used to make these portraits in Photoshop starts with making a new file, or a new canvas. The actual file sizes used in the game are very small, but using digital art software makes it almost irrelevant what size the canvas is at the beginning. This can be adjusted later, either by cropping and resizing the image or by copying the image out of the original file into a new one that is the correct size. What's important, at least in Photoshop, is the resolution and the color mode. If these are set up differently for different files it becomes infinitely more difficult to edit the images, because combining different files of different color modes can completely change the colors and transparency already painted.

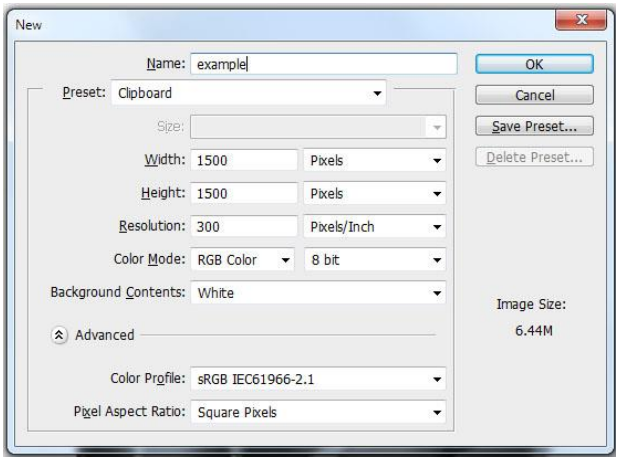

Figure 3.1: Creating a photoshop document

The next step is to take the number of layers into account. Each file is created with a locked background layer, which is a white background. Painting directly onto the background layer can make it difficult to edit later, so it's normally best to start on an empty layer, seen below as "layer 1." When an element of the painting needs to be manipulated or moved around the background layer can actually get in the way, partially because it's a filled layer above an assumed layer of transparency. This is similar to painting on a physical canvas and realizing there is a serious error embedded deep in the painting and the only option is to repaint that entire area.

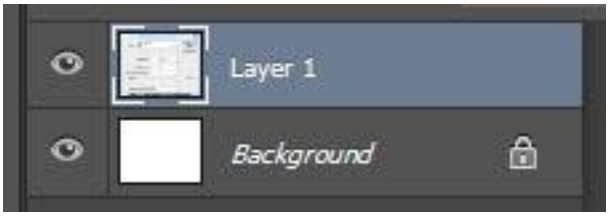

Figure 3.2: The background layer acts as a blank canvas

The method used for this project was to make an initial "sketch" layer, and then copy that layer to paint without fear of making an unchangeable action. Below is an example of the average number of layers involved.

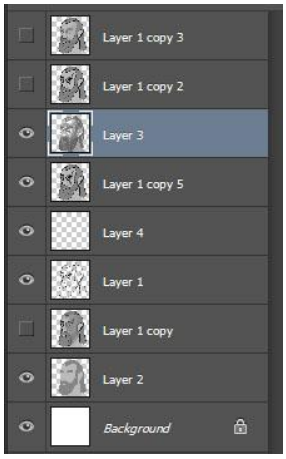

Figure 3.3: multiple layers

Using layers this way helped keep the overall editing time low and also allow for multiple different versions of a painting to be created easily.

Figure 3.4: Several layers combining to make one image

The importance of layers can be examined again with the use of colors. Each layer can be assigned to a different task within the file. Most of the concepting and painting were made on "normal" layers in greyscale, but when applying color or making something glow different assignments can be made. Using the option labeled "color" allows for any brushstrokes to only apply color data to the already painted segments of the file.

Figure 3.5: Only the already painted parts will take on color

Figure 3.6: various layer types available in Photoshop

All of the paintings made for the project were made in this fashion, with varying levels of detail and initial detail. After they were in a relatively completed state they were then resized by pixel width. Because the application was meant to work on a tablet the actual sizes of the painting had to be sized down dramatically, often losing detail/definition in the process.

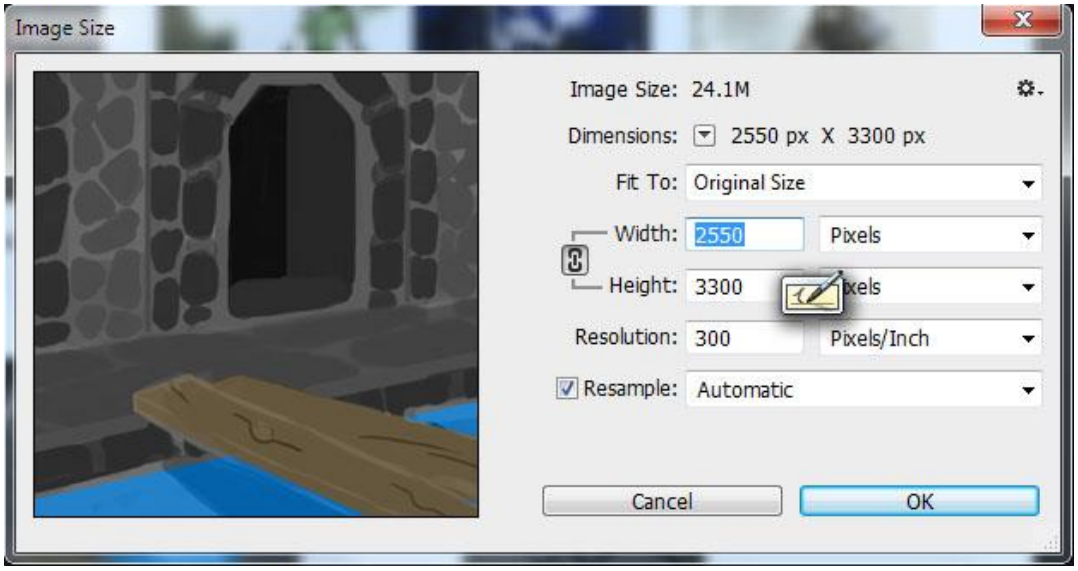

Figure 3.7: Image resizing was also done in photoshop

Figure 3.8: Barbarian high resolution portrait Figure 3.9: Barbarian low resolution portrait

Each of the images also needed to be encoded as a certain file type. The project used Jpeg's and PNG's, and all the master files were stored as Photoshop documents, or PSD's.

3.2 Survey Techniques

One of the most important features of the application was ease of use; as such, finding out user preferences needed to be done before any implementation. While there would be a variety of options in the app for different types of users, understanding whether users would want more control, such as with a more in-depth character creation system or a more comprehensive rule set, needed to be done first. Establishing preferences first gave us a solid base to work from, giving us a reference point throughout development on what the priorities should be. Had we not determined

these preferences first, we might have developed the core of the interface in a way that would be incompatible with what users actually wanted to do with the app.

We decided on surveys as the means to determine the user preferences for our app. Tabletop RPGs are frequently designed to be flexible with a variety of styles of play. In order to gather an appropriately flexible measure of user preferences, large amounts of data would be required, so that important functionalities would not be missed because a subset of play-styles had not been measured. This meant we needed a data collection method that allowed for a large sample size. Without this large sample size, non-representative sampling would be an extremely likely occurrence, especially amongst more experienced players, who would be more likely to articulate desires in the framework of a play style they have greater familiarity with. Surveys are a frequent choice for large-scale data collection, as the survey-taker does not have to be involved individually throughout the process for each individual.

The survey we developed [see Appendix B] was designed to gather information on general preferences of the three primary user types as previously defined (beginner, advanced, and expert). The questions were designed to gather generalized preferences which could be used as a guideline in a wide variety of scenarios. This helped us to avoid results from the more experienced users that would be too specific to their personal experiences. It also forced us to write the survey in a way that subjects who had not previously played a tabletop RPG would still be able to understand. The survey was designed to be confidential and anonymous, with all questions designed to give limited answers to focus on generalized data rather than specific data.

The surveys were distributed two ways: in person and via email. Emails were only be sent to subjects who specifically request distribution in that manner, and written versions of any instructions that were given verbally were instead included in the email. When the surveys were distributed by hand, the person conducting the survey first informed the participant that their results would remain confidential and anonymous, and that the user could stop taking the survey at any time, as denoted by a participation release form [see Appendix something]. The subject was then given the survey to complete. Upon completion, the survey was stored in a non-organized folder, to reduce the possibility that a survey could have been matched with a subject based on the order of distribution. If any personal markings were left on the survey (such if the subject wrote their name, unprompted, onto the survey), the subject was informed that they were leaving identifying information, and were advised to remove that identifying information.

The initial survey questions were preceded by instructions telling the subject to skip any questions they do not understand. After these instructions, the survey asked the subject a series of yes-no questions designed to gauge the subject's experience with tabletop games, so that their results could later be grouped by level of experience. These questions also serve the function of determining whether or not a subject read the initial instruction; the subject would likely be unfamiliar with at least some of the terms in the first set of questions, and if the subject displayed

confusion they could be informed of the initial instructions. The rest of the survey consisted of 7-point Likert scale questions, broken into sections covering different subjects, with each section preceded by a short paragraph explaining the top of those questions and providing context for the questions as necessary. The sections covered included depth of the character creation system, freedom of use of the application for the dungeon master, single player/AI dungeon master importance, variation in rule sets, precision of movement in the application, and play-testing capabilities.

Below are the survey results. The table contains the average of the results across all 33 surveys, as well as across the 14 survey results from inexperienced subjects, 11 results from subjects with some past experience, and 8 results from subjects with extensive past experience:

Question #	Average Overall Result	Average Beginner Result	Average Advanced Result	Average Expert Result
Average Across All Questions	5.289	5.187	5.217	5.462
Question 1	5.928	5.308	5.727	6.75
Question 2	5.53	6.14	5.2	5.25
Question 3	4.42	5.21	4.18	4.27
Question 4	4.36	4.43	4.27	4.38
Question 5	4.85	4.14	4.55	5.88
Question 6	5.26	4.43	5.73	5.63
Question 7	4.59	5.14	4.64	4
Question 8	5.38	4.93	5.45	5.75
Question 9	6.15	5.79	5.91	6.75
Question 10	4.88	4.64	4.64	5.38
Question 11	5.99	5.86	5.72	6.38
Question 12	4.70	5.14	4.45	4.5
Question 13	4.97	5.5	4.91	4.5
Question 14	5.45	5.21	6	5.14
Question 15	5.45	5.29	5.64	5.42
Question 16	4.75	4.28	5.4	4.57
Question 17	5.14	4.61	5.82	5
Question 18	5.24	5.4	5.6	4.71
Question 19	5.47	5.6	5.1	5.71
Question 20	5.65	5.2	5.89	5.86
Question 21	5.03	5.1	4	6
Question 22	5.90	5.13	5.75	6.83
Question 23	6.04	6.38	5.25	6.5
Question 24	5.78	5.63	5.38	6.33

When analyzing the surveys, it was important to remember that the individual questions were intended to be used together to generate meaningful data, rather than try to write questions that could determine a preference independent of other results. This technique was learned from Chris Krishna-Pillay, manager of the Victorian branch of the Commonwealth Scientific and Industrial Research Organization's (CSIRO) education branch, who used similar methods of analyzing *relative* survey results, rather than the direct results of each question, when collecting data to gauge public interest in educational media CSIRO would release [Chrishna-Pillay 2003]. To avoid inaccurate data being collected because a single question was worded poorly, many questions were designed to be redundant with slightly different wording, and the average of the redundant questions were used to provide useful insight into what tradeoffs would be most desired by the user. If one question *did* consistently get different scores from the redundant questions, the exact wording of that question could be compared relative to the other questions, possibly indicating a result we had not been searching for.

3.3 User/Task Analysis

Based on our surveys, along with our background research and past experiences, we came up with a user analysis and a task analysis. User analysis is crucial to determining what requirements an application needs to meet; by establishing key factors about the expected user base, we also determine what usability conditions need to be met. Task analysis gave us a framework for what types of situations and interfaces our application would need to handle.

3.3.1 User profile:

Beginner

- **Age:** Assuming at least 7-10 years old minimum, no upper limit on participation bound by age specifically (Based on minimum expected cognitive development, and violent, although not necessarily graphically violent, content).
- **Gender:** No expectation of gender.
- **Language:** Basic understanding of English language (read simple sentences, low bound based on minimum expected age).
- **Experience:** Little to no prior experience with role playing games, is expected to not know terminology or general standards for rules.
- **Motor Skills:** Basic fine motor control, can successfully press buttons of medium (specify exact dimensions of medium later, probably slightly larger than the average key of a keyboard but check scaling of buttons that size to screen) size.

- **Eyesight:** Capable of letting user read at least two lines in 14 point font at a time
    - Capable of viewing a lit screen for at least 1 hour without abnormal eye-strain.
- **Cognitive ability:** Focus and memory capable of letting the user read about 150 words (about half a page double spaced in Microsoft Word) in a large font, in order to keep track of gameplay/plot information.
- **Gameplay comprehension:** Ability to keep track mentally of very basic arithmetic (can figure out that a +2 bonus and a -1 penalty adds up to +1), and able to follow basic multiplication or division (if a bonus/penalty is +-1 for each 10 of a unit, such as every 10 pounds of equipment giving a -1 penalty to swimming checks, the user can follow that math if the app does it for them).

Intermediate

- **Age:** Assuming 10-13 years old minimum, no upper limit on participation bound by age specifically.
- **Gender:** More likely to be male than female, but not specifically intended for either gender.
- **Language:** Can read English at a moderate level.
- **Experience:** Expected to understand basic terminology or expectations of the game (familiar with concept of dice rolls or types of dice used, familiar with basic class archetypes, able to exploit game mechanics based on clichéd or popular storylines/game mechanics [a red enemy is more likely to use a fire spell, and is probably weak to water or ice, an enemy in a robe with a staff will cast spells, etc.], although the application may need to explicitly state the potential to exploit these game mechanics to the user).
- **Motor Skills:** Basic fine motor control, can successfully press buttons of medium (specify exact dimensions of medium later, likely to be around ½ to ¾ of an inch) size.
- **Eyesight:** Eyesight capable of letting user read at least two lines in 14 point font.
    - Capable of viewing a lit screen for at least 1 hour without abnormal eye-strain.
- **Cognitive ability:** Focus and memory capable of letting the user read approximately 300 words (about one double spaced page in Microsoft Word) in a large font.
- **Gameplay Comprehension:** Can keep track mentally of basic multiplication and division, basic understanding of probability (can guess odds of how many hits an enemy can take [damage is often calculated with dice rolls and a bonus/penalty modifiers stated in rule-set]).

Expert/Experienced

- **Age:** Assuming 10-13 years old minimum, no upper participation limit bound by age specifically.
- **Gender:** More likely to be male than female, but not specifically intended for either

gender.

- **Language:** Can read English at a moderate level (tempted to say 7th grade, but need to double check full implications of that statement).
- **Experience:** Expected to be capable of exploiting game mechanics in ways unintended or unwanted by the programmer (Capable of finding glitches or balance problems based on experience with similar games, and capable of exploiting them).
- **Motor Skills:** Basic fine motor control, can successfully press buttons of medium size (specify exact dimensions of medium later, probably ½ to ¾ of an inch).
- **Eyesight:** Eyesight capable of letting user read at least two lines in 14 point font at a time.
    - Capable of viewing a lit screen for at least 1 hour without abnormal eye-strain.
- **Cognitive ability:** Focus and memory capable of letting the user read approximately 600 words (about two double spaced pages in Microsoft Word) in a large font.
- **Gameplay Comprehension:** Can keep track mentally of basic multiplication and division, basic understanding of probability (can guess odds of how many hits an enemy can take [damage is often calculated with dice rolls and a bonus/penalty modifiers stated in rule-set]).

Modder (User who intends to develop and distribute modifications and add-ons to the game

- **Experience:** Assumed to have the experience and requirements of intermediate users; in addition has the experience of an expert user.
    - Understands concepts of downloading and patching, although does not necessarily understand the process behind them.
    - Understands concept of forums/message boards, and can use them as a resource.
- **Internet Usage:** Has consistent access to internet, and can be expected to use online resources.
- **Eyesight:** Capable of reading 14 point font for up to half an hour without abnormal eye-strain.

Dungeon Master

- **Experience:** Assumed to have the experience and requirements of intermediate users.
    - **Understanding of Meta-Game Aspects:** Can associate different actions players wish to take with relevant aspects of gameplay; can determine whether a player's unique idea fits with a condition of a skill or a constraint of an instance.
- **Eyesight:** Capable of reading 14 point font for up to half an hour without abnormal eye-strain.

3.3.2 Task Analysis:

Below is our initial state diagram, which helped to organize our thoughts on transitions within the game, and to act as a guideline once implementation began. While there are now discrepancies between the state diagram and our final code, the state diagram helps to explain the original goals of our program structure.

State Diagram:

0. Starting program: waits for input from player.

    0.1 Giving any input brings to **Main Menu State**.

1. Main State

    1.1 Options: **New game**, **load game**, **settings**, exit program.

        1.2 Selecting any of the four will immediately proceed to the respective program states, except exit program which ends the program.

2. New Game State: (State for starting a new game file, with starting-stats characters).

    2.1 Asks user if they want the AI to be the DM, with a yes, no, and back option. Going through this state should happen fairly infrequently, and an individual user who does not play with a wide variety of groups should ideally only have to do this once or twice.

        2.1.1 Selecting AI DM causes all enemies, NPCs, and challenges to be run by the AI.

            2.1.1.1 If the AI option is chosen, any "judgment calls" the DM would normally make are only available if they were pre-programmed into the instance. In the case of combat, the AI will default to a very simple "best move" algorithm for the enemies, unless simple variations have been pre-programmed into the instance (placing greater value on attacking one target over another, for example). Players will be expected to input their actions in this version.

            2.1.1.2 Selecting Player DM causes the game to be controlled by the player, with significantly more information displayed to aid the DM, and control of enemies, NPCs, and challenges given to the player. "Judgment calls" will also be available to the DM in this version. Only the DM is expected to directly control the application in this version, but an option to temporarily disable the additional

information will be provided so that the DM may display the current state of the game to players via the app as appropriate without letting them see intentionally hidden information.

2.1.2 Back option returns to main menu options.

2.2 User specifies how many (non-DM) players there are, limited to 1-9.

2.3 User specifies their experience with the style of game, has three objects each representative of potential skill levels, and a confirm option.

2.3.1 Selecting an experience level displays text describing that level of experience.

2.3.1.1 Back option unselects the experience level if one has been selected.

2.3.1.2 Selecting confirm has the game proceed to character selection. Each selection results in different instructions given whenever there are instructions.

3. Load Game:

3.1 Select previous save file with profile name previously given by user, name of instance the player(s) was/were involved in, and time spent playing (to aid in identifying multiple saves of the same game, with the same names [Note: this is an established strategy. The idea is that if a user wants to save a backup file, looking at the timestamps can help them identify which is the earlier version]). Selection needs to be made about once per session, restarting should ideally be unnecessary.

3.1.1 Confirm loading of that save file.

3.1.1.1 Sets system to the **In-Game State** at time of save file.

4. Character selection:

4.1 Three options: Create new character, use pre-made character, and back.

4.1.1 Back has text asking user to confirm

4.1.2 Returns user to main menu on confirm

4.1.2.1 Create new character: Task is performed with low to high frequency depending on user, and with a high frequency for a group as a whole.

4.1.2.1 User specifies character race, gender (if applicable), class, and class modifiers. After each answer, a visual representation will be updated with features based on those answers, to give the player feedback on character appearance and characteristics.

4.1.2.2 User specifies base statistics (stats), with task objects representing incrementing or decrementing base stats.

4.1.2.3 Once base stats have been confirmed, program asks user to give additional bonuses to specific skills. Upon completion of all questions, program asks player to confirm character while displaying finalized stats on the player character. Upon confirmation, user proceeds through the loop described in 4.2.

4.1.3.1 Use pre-made character: User is asked to select one of several objects, each with a portrait and a name. A method to change the displayed objects is available if there are more objects than can fit on the screen, and if there are several times the number of objects that are require the use of the display changing method, then an alphabetical sort object becomes available. Individual users are expected to do this with low occurrence to semi-frequency, and as a group are expected to do this several times in a session, with sessions with this occurrence being infrequent.

4.1.3.1 Back prompts user to confirm that they want to go back to **Main Menu State**.

4.1.3.1.1 Confirming returns user to **Main Menu State**.

4.1.3.2 Selecting a character displays basic statistics, and presents the user

with buttons to change which statistics are being displayed.

4.2 Selecting confirm leads to the process being repeated for each player.

4.2.2.1 After a character has been confirmed for each player, change to **Instance Selection State**.

5. Settings State

5.1 Has options including sound levels and text size (more options may be added later).

5.2 Each option allows user to increase or decrease associated value. Changing options is expected to be an infrequent task.

5.3 Selecting back returns the user to the previous state.

6. Instance Selection State

6.1 Has a list of instances, each with a name, a difficulty rating, a picture based on the primary environment graphics used, and whether or not the instance is a tutorial instance.

6.1.1 Tutorial instances include additional instructions and prompts for the user.

6.2 Also has save and back actions.

6.2.1 Selecting back asks user to confirm that they want to return to the main menu, reminding the user that any unsaved data will be lost. If confirmed, returns user to the **Main Menu State**.

6.2.2 Selecting save sends user to the **Save Menu State**.

6.2.3 Selecting an instance brings up text describing the instance, and causes the program to ask the user to confirm that they want to begin the instance. This is expected to be done infrequently.

6.2.3.1 Player can select another instance to change the text, and which instance is begun upon confirmation.

6.2.3.2 Confirming an instance causes the program to enter the **In-Game Exploration State**.

7. Save Menu State

7.1 Shows user a list of save files, equal to the number of saved states + 1, unless current save states exceeds a memory bound, at which point it will only show the save files of already existing save files.

7.1.1 Upon selecting a save slot, program asks user to confirm that they wish to save the current game state, and if saving in a save slot that already has a save state asks user to confirm that they wish to overwrite the save state. This action is expected to be performed frequently.

7.2 Selecting the back option returns user to the previous state.

8. In Game Exploration State

8.1 Shows user an image meant to provide context for in-game location, text giving further description, and a movement, exploration, status, save, exit, and settings object.

8.1.1 Selecting the save or settings options switches the program to the respective menus, while the exit options confirms that the user wishes to exit the game while reminding them to save before they exit, and if the player confirms it moves the program to the main menu.

8.1.2 Selecting status moves the program to the **Character Status Menu State**.

8.1.3 Selecting the movement object brings up a list of locations the player may move to, each with a brief description ranging from describing the location to describing the direction the location is in. Confirming movement brings the character to either an alternate **In Game Exploration State**, or an **In Game Combat State**. Accessing the list and choosing a location to move to is expected to happen frequently.

8.1.4 Selecting the exploration object brings up a list of tasks a player may do, along with a blank skill check option if one of the users is acting as the Dungeon Master, and the program is set to advanced or higher, as determined in the **New Game State**.

8.1.5 Selecting the blank skill check option gives the Dungeon Master a

list of the default skills, along with an “other” option. The dungeon master may select as many of these as they wish. Blank option selections should occur very infrequently, unless the dungeon master and players are very experienced and are willing to deal with a complex system.

8.1.5.1 If the “other” option is selected, another list is brought up with any non-skill-specific bonuses a player may have based on skills or items, along with a +x bonus option.

8.1.5.1.1 If the +x bonus option is selected, the Dungeon Master inputs a positive or negative integer.

8.1.5.2 Once all modifiers have been selected, the Dungeon master is prompted for the skill check difficulty.

8.1.5.2.1 The program then calculates whether the skill check was met, and informs the Dungeon Master while prompting them for the results.

8.1.5.2.1.1 Results are selected from a list, which may include giving items, currency, locking or unlocking tasks or locations, or adding a bonus to the player’s skill list. Selecting currency prompts the user for an amount, either positive or negative; selecting an item prompts the user for a name, weight, value, and quantity to add, or a name and quantity to take away. Locking or unlocking tasks/locations brings up a list of tasks for the current location, or a list of locations; selecting one of them asks the user to confirm the lock/unlock of the task/location. Selecting a bonus brings up a list of applicable skills, and a blank option which prompts the user for a description. Each base option can be selected multiple times.

8.1.6 Selecting a task brings up a description of the task. Tasks may include nothing besides the description, or may include a skill check (to be handled in

a way similar to the blank skill check, but with some skills or non-skill-specific bonuses and penalties already applying, and with rewards pre-determined. Some tasks may also automatically give out rewards, either positive or negative, either once or repeatedly. Tasks may also move the program to the **In Game Combat State**, the **Dialogue State,** or the **Shop State**, or unlock other tasks/locations. Task selection is likely to occur with extremely high frequency.

9. Dialogue State

9.1 Shows user an image of who's talking, along with text representing what's being said. If the text is larger than the space the text is displayed in, the program waits for any input from the user, then displays the next portion of the text. Once all of that section of text has been displayed, the interface allows the user to select to go back to previous sets of text. If the user gives any other input, the dialogue goes either to the next block of text, with a new image for the next person talking, or it prompts the user for an option. Going through dialogue will happen with varying frequency, in some instances occurring at most a few times over an entire session, or in other instances being the most frequent occurrence. The assumption is that the user will be warned ahead of time how much dialogue there will be in an instance, and choose which one to play accordingly.

9.1.1 When prompted for different dialogue options, the player selects an object representing their preferred dialogue choice. Depending on the option, different dialogue options are displayed.

9.2 After the last sequence in a dialogue tree, the dialogue tree will switch to the **In-Game Exploration State**, **In-Game Combat State**, or the **Shop State** based on which inputs the player gave.

10. Shop State

10.1 Displays an option to buy or sell items.

10.1.1 If selling items, displays items in inventory that match what the shop is willing to buy. The sellable items can be based on item type, unique characteristics like an item material. Sell values will be at a

percentage of the item's value.

10.1.2 If buying items, displays how much money the player currently has, along with items being sold, their quantity (if limited), and their price, which is a percentage of their value.

10.2 Buying and selling items is expected to occur infrequently during play, unless an instance specifically revolves around economics

10.3 Once done, a player can select to leave a shop to return to the **In-Game Exploration State**.

11. Character Status Menu State

11.1 Displays an object for each character. When an object is selected, it displays information about the character. This information is broken up into different pages, which can be cycled through. Pages include base stats and skill check bonuses, temporary bonuses, class skills/spells (which may be on several pages, depending on the number of skills/spells the character has), and inventory.

11.1.1 Base stat page will display both the "normal" stat values and any bonuses or penalties to these stats that would be explained on other pages.

11.1.1.1 If the game is being run by a dungeon master, any temporary bonuses from blank skill check events may be altered or removed by selecting the listing, then choosing to alter or delete. If altered, the program will ask the user for input for the new value.

11.1.2 Skills or spells that can be used out of combat (such as a healing spell, or a spell which provides a temporary bonus) can be used by selecting their listing on their information page. Once selected, the program displays the potential targets of the skills or spells, and the player is asked to select a target, then confirm that they want to use the spell.

11.1.3 Items can be equipped, used, or shared with other characters by selecting their listing on the item information page. Much like skills and spells, any targets for equipping, sharing, or using items are listed, and the player is asked to confirm.

11.2 Once the user is done looking at their status, they can return to the previous state

of the application by selecting and confirming to go back.

12. In-Game Combat State

12.1 The **In-Game Combat State** begins by displaying a square grid overlaid onto a map of the space the battle is taking place in. The user is prompted to select tiles in the game state to place the player characters in one at time, with valid tiles being indicated to the user. The user is then prompted to confirm the correct placement of all characters; if not, the process begins from the beginning. Once the characters have been placed, the game shifts to turn based combat. If there is no Dungeon Master player, the AI controls the enemies and acts as the dungeon master, and waits for the player's input before responding with their actions.

12.2 During the players' turn, the player has the option of entering the status menu, the save menu, the settings menu, exiting the game, giving a movement command, giving an action command, and ending their turn.

12.3 Entering the **status** or **settings** menu changes the game state to those states, while exiting the game returns the user to the main menu.

12.4 Selecting a movement command causes the application to display the available grid spaces that the character can move to. Once the user has selected a grid space, they are prompted to confirm their action. If confirmed, the character moves there and is no longer able to make a move action that turn. The exception is if an advanced/expert user is acting as Dungeon Master, in which case they can make another move after confirming that they wish to break the established game rules. If the user changes their mind about wanting to move, a back object can be selected to return them to the main list.

12.5 Selecting an action brings up a list of all actions the character can take. Actions which cannot be taken because too many actions have already been made, as defined by the rules, are greyed out, and cannot be selected without confirmation from a player dungeon master, who has set to advanced or expert that they wish to violate the rules. Once an action has been selected, they must choose a target (if any) by selecting a tile with a legal target on the map, then confirm that they wish to perform the action. If there is no targeting required, the program merely checks for confirmation from the user.

12.6 At any point, the user can choose to end their turn, causing the program to move to the next turn.

12.7 Once combat ends (as a result of no enemies left on the map), the game returns

to the **In-Game Exploration State**, after informing the user of the results of battle in terms of money and experience points.

Task Objects:

Task objects served a similar purpose as state diagrams in helping to organize our thoughts and act as guidelines. While the state diagram focused on transitions and states, task objects helped us to better understand what kinds of data structures we would need.

Characters, which store statistics, skills, bonuses/penalties, items, and keywords
Bonuses/Penalties, which have descriptions, numerical values, skills, and keywords
Items, which have a weight, value, effects, and keywords
Instances, which have locations and permissions
Locations, which have events, permissions, and keywords
Events, which have transitions and items/bonuses

3.4 Evaluations of Prototypes

3.4.1 Usability Test

Once we designed and implemented the first complete draft of the interface, we tested how easy it was to use by having a group of volunteers attempt basic commands. The volunteers were selected based on individual interest and past experience with videogames. Some volunteers regularly played games within the same genre as ours, while others had not regularly played videogames for years.

The interface experiments were designed to test whether the sequence of inputs for various tasks matched what a user would expect those inputs to be. While the game normally begins with a brief set of instructions, these instructions might be skipped if one of the users is impatient, or the instructions may be forgotten. The sequence of inputs for basic tasks were designed so that a user who had skipped the instructions could quickly discover what the correct commands are through guessing and experimentation. This would aid a user in operating the program without having to restart the program if the tutorial is ignored or not remembered.

In order to test whether the program was as intuitive as we had hoped, volunteers were verbally instructed to perform mix of common tasks, with questions interspersed throughout. This allowed us to determine whether a user with no prior experience could perform those same tasks easily. The time it took each participant to perform the instructions was timed. These times were recorded, along with the answers each participant gave to the questions. Both types of data are listed below.

While we attempted to get as accurate a measurement on the usability of our interface as

possible, we were unable to achieve some goals during testing. All test subjects had at least some prior experience playing videogames; the difference between the two test groups is whether or not the subjects usually played videogames multiple times a week at any point within the last 5 years.

3.4.2 Playtests

After we finished adjusting our interface based on the results of our usability tests, we then proceeded to our final playtest. The playtest consisted of having four volunteers play through the instance designed by the project group. The volunteers were selected based on their willingness and prior experience with tactics RPGs. Choosing a group with previous experience allowed us to have an easier time looking for certain gameplay problems involving exploits in game rules. In particular, we wanted to check whether there were any strategies or tactics that would make the game impossible to lose, regardless of how well those strategies or tactics were executed or the relative strengths of the players and enemies. If the volunteers found a certain combination of skills that would allow them to defeat an enemy without taking damage, regardless of how powerful the enemy was, without the players having to use a large portion of their available mana, the overall effectiveness of the skills and the characters would have to be adjusted. While novice users would allow us to better test how easy it was to use the interface and understand the rules, we felt that experienced players would allow us to better test overall gameplay balance, and that major problems for novice players would still be minor problems for experienced players, which was something we could look for.

4 Game and Application Design

Once we had determined what the key characteristics for our application were, we could begin determining what it would look like. Our design process could be broken down into three core ideas. The first was the technical question of how we wanted the menus to work. The second was what patterns and styles would be necessary for art. The third was how would our rules be best adapted to a programmed game. Below, we explain the key principles and decisions we made for each aspect.

4.1 User Interface Design

When our game is being played, especially when the game is in a combat state, the device running our application will, in some situations, be expected to be passed around by a group of people, for use by one at a time. As a result, each user will spend significantly less time manipulating the battle interface for the app than playing the game, and when they do interact with the interface, it will be for short, well-spaced periods. This less-frequent use will mean that a longer amount of time spent playing the game will be required in order to become used to the interface. To compensate for this inherent weakness in that situation, the interface needs to be intuitive, with users quickly gaining a deep understanding of their options.

However, some groups of players will have one user acting as the Dungeon Master. This user will be spending the entire time using the application, and will be able to become quite proficient with it very quickly. These kinds of users will quickly grow frustrated with a user interface that will remain as hard to use after extended use as when they started. In order to avoid alienating these more frequent users (who are also likely to be the ones managing the gaming session, and without whom the game could not be played), the interface will need to allow for faster, potentially more complicated, inputs. These users will need to be able to repeatedly make selections from a large list of options in a quick and efficient manner.

In order to make sure that the application will be easy to use for infrequent users and efficient for more experienced users, the menu system for the battle has to be both easy to understand and use, while capable of fast and efficient selections. As a result, the two key characteristics of the interface are designed for consistent comprehension of options and ease of selection. The former will allow new users to understand what their options were, and the latter will reward expert users with the fastest possible completion times.

4.1.1 Menu Design

To handle consistent comprehension of options, a menu with five primary options, *attack*, *move*, *skill*, *item*, and *end turn*, was chosen. These specific main options are chosen to give players a reference frame for what kinds of actions can be taken. *Attack*, *move*, and *end turn* are designed

to be self-contained options, without submenus, while *skill* and *item* will each have a sub-menu containing a list of options for that category. This prevents the three simple (and very frequent) options from being obscured by the quantity of actions that will fall under the category of "*skill*" and "*item*".

In order to choose what menu options we would have, we studied the standards set by many videogames with menu-based combat. Our tabletop RPG shares many characteristics with Tactics RPG genre of games, specifically those with large skill-pools, allowing us to base many aspects of our menu on the conventions from that genre of game. This gives any user who has played Tactical RPGs an advantage by giving them familiar options and structures. Although users are not expected to have any prior experience playing videogames, the rise in ports of older console games (which tend to have a similar flow of battle) to tablets means that any potential user who might find our app will be likely to have also seen famous and highly praised games with similar structures. This presents a possibility for pre-existing assumptions about our interface, which will need to be addressed. The five main options included in our design are popular menu options for Tactical RPGs, and are very similar to what was used in the well-known Final Fantasy Tactics/Final Fantasy Tactics Advanced (abbreviated as FFT/FFTA) series of games[1&2]. Additionally, many games in the Tactical RPG genre have similar menu systems with slight variations on the available options, with the main difference being the inclusion of game-specific "gimmicks" for the players to use. This means users who had played the FFT/FFTA games would be able to draw direct parallels with the contents of each menu option, and users who have played any games similar to ours will be able to very easily infer those contents. More importantly, anyone familiar with the genre (and thus would have a reason to be interested in games similar to it) won't be penalized by having to relearn menu systems that break conventions that they, as users, have been trained to expect.

While there are menu systems for Tactical RPGs that greatly deviate from ours, including those made for PCs with a similar method of input, many of these games have a combat system with a different focus. These games tend to focus on more "modern" warfare based around firearms, and thus have a greater focus on relative positioning of characters or personal resource management such as heavily limited ammunition. These games tend to challenge the player to make good decisions *outside* of skill selection to make a limited variety of actions more effective (with more importance placed on line of sight or cover than which of two pistols is used to shoot). As a result, the UI for action selection in those games is highly simplified while displayed information is complex. In contrast, our game focuses on effective skill selection to make an action more effective. Thus, a standard for menus based more closely on console Tactical RPGs, with increased focus on displaying a variety of options, was considered more useful.

The different types of menu options we are using are designed to make use of the application easy for novice users. However, the potential selections in the battle menu do little to help experienced users operate the application quickly. Having fewer options does allow us to

[1] Square (1997). Final Fantasy Tactics [Video Game for Playstation Console]. Squaresoft.

[2] Square/Nintendo, (2003). Final Fantasy Tactics Advance (Video Game for Gameboy Advance Console], Square-Enix

increase the physical size of each option while taking up the same fraction of the screen, which would increase selection speed according to Fitts's law. However, the presence of sub-menus counteracts this benefit: forcing a user to make multiple smaller decisions is generally a slower process. This is shown by several different metrics. Hick's law states that each additional menu option slows a user down by a smaller amount than the previous option; this implies that adding an option to one larger list has less of an effect than adding an option to one of two smaller lists, and thus using fewer lists is faster (Raymond, 2005, section 4.3). Likewise, Fitts's Law would indicate that the total time to make two selections, each with either half the distance travelled or twice the margin of error (since each selection could have its size on the screen doubled while the total space for all options uses the same amount of space on the screen), will still take longer than one more difficult selection. To counteract this, a layout that experienced users can traverse quickly will be necessary.

In order to allow experienced users to quickly move through the battle UI, pie menus are used for the five main options, with "*skill*" and "*item*" expanding into further pie menus upon selection. In these pie menus, both sliding gestures (where the user does not lift up their finger and instead "drags through" the menu options) and single clicks are used as selection methods. Pie menus with sliding gestures allow for faster movements by allowing for a near-infinite margin of error: users only have to gesture in the correct direction, without having to worry (for the most part) about over- or under-shooting the selection. The risk of "missing" a menu option does reappear when the "*skill*" and "*item*" menus open up into their respective sub-menus (by gesturing far enough past the location of the main option to indicate a sub-option in the same relative position in the sub-menu). However, users can easily correct this accidental input as long as quickly-displayed feedback on the final selection is given.

In addition to the evidence presented by Fitts's law, higher efficiency is also implied by the Keystroke Model, which determines average task times by summing up the expected time requirements for each motion the user must make during the task. According to measurements taken when first establishing the Keystroke Model, moving a cursor over an option and clicking that option takes approximately 1.3 seconds for an average person, while drawing a straight line takes .9 seconds +.16 multiplied by the length (in cm) of the line (Card, Moran, & Newell, 1980). Assuming that the length of the line is non-negligible but as small as possible (1 cm), selection by sliding gestures in a pie menu takes approximately 20% less time than selection by clicking. While the Keyboard Model was measured using a mouse rather than a touch screen, similar speeds have been demonstrated in experiments with touch-based systems (Schulz, 2008), making this model applicable to our application.

When deciding on pie menus, we must also make sure to take into account some of the inherent problems with their use in our application. Pie menus are rarely used in skill-focused Tactical RPGs, meaning their use in our app will go against common conventions. However, as previously stated, skill-focused Tactical RPGs are also frequently made for consoles, which use 4-10 buttons and some combination of four directional buttons or a joystick. In contrast, our game

uses a touch screen for input. Attempts (such as the iOS port of Final Fantasy Tactics[3]) to use touch screens to navigate the list-box-style menus used in those console games have been met with complaints due to GUI elements sized for reading rather than selecting and awkward navigation systems designed around the use of physical buttons rather than touch controls (Final Fantasy Tactics: The War of the Lions for iPhone..Reviews, 2013). Admittedly, many complaints were aimed at the interface as it appeared on iPhones, which have smaller screens than the device we are intending our app for, reducing concern about how such a menu would scale to our intended device. However, no user feedback was provided about the use of the app on an average-sized tablet (as the alternative to an iPhone, an iPad, has an exceptionally large screen for a tablet), so any input method that ran the risk of functioning poorly with small screens needed to be avoided.

The common alternative for ports to devices with touch screen inputs is to feature static buttons representing the various inputs that would be used for consoles. We decided this was a poor design choice, as it forces arbitrary limitations upon the input system in order to make porting the game with the original layout easier. Since we were creating our own game, we decided that alternative inputs designed to make best use of a tablet's features was preferable to following conventions that were decided upon for reasons not applicable to our game. These decisions discouraged us from using list-boxes, and encouraged us to adapt the same information but presented in a different style of menu.

Additionally, the use of extensive menus positioned on the edge of the screen or expansive submenus, as used in most PC-based Tactical RPGs seems infeasible as well. Edge-of-screen menus work on PCs because of the nature of mice: a user's cursor cannot be sent off of the screen, preventing the same kind of "overshooting" that a pie menu does. However, with touch screens, this overshooting can still happen, and the number of options available on those menus will obscure too much of the screen if enlarged to a size appropriate for touch interactions. Expansive submenus designed to fill up half the screen are generally designed solely for users who are expected to become experts in that game's interface, with the menu used to convey large amounts of detailed information. Because our game's in-combat system needs to display significantly less detail, and because of the likelihood of low-frequency users playing in a group setting, these complicated sub-menus will be hard to learn and will only be used for decisions in excess of what a player should expect to make during their turn.

Aside from concerns about conventions, the other weakness that needs to be accounted for is how difficult pie menus can be for novice users. While menu options can be executed quickly, the single motion nature of the pie menu puts more pressure on users who are unsure of their exact

[3] Square-Enix (2011). Final Fantasy Tactics: War of the Lions [Video Game for Apple iOS]. Tose/Square-Enix.

goal. To help alleviate this problem, our interface has multiple features to help users select the correct option.

The most immediate feature we have for aiding novice users is the inclusion of click-based selection. While a user can go as far into the menus as they want with sliding, upon release the current menu will stay selected, and the menu can be navigated by touching the desired sub-menu option once. This way, a user can slide through basic decisions (such as whether or not to use a skill) and then more easily take their time making a decision during a more complex decision (such as which skill to use), and move their hand to avoid obscuring potential options.

Another feature we originally planned for, but did not have enough time to implement, was a final confirmation for any decision that will cause the user to leave a menu system or make a non-reversible decision is accompanied by a confirmation window separated from the pie menu. While Fitts's law states that moving the confirmation away from the window will lengthen the time it takes to complete a task, it also prevents a user from accidentally making an irreversible decision by forcing the user to make a very deliberate input. In order to make sure that the confirmation window placement does not obscure the sequence of options to navigate through the menu, the confirmation window is always positioned near any new frames or graphics used to explain a menu option. A user unfamiliar with the app will see the confirmation buttons once they had finished reading about their selection, while a more experienced user will be able to memorize the location of the confirmation buttons. Through these methods, we are able to aid new users in correctly using the menu with minimal loss in efficiency.

4.2 Rule Set

Our initial rules were based on the assumption that our game would be able to function both as a Tactics RPG and a Tabletop RPG, and thus was heavily influenced by the project members' prior knowledge of Tabletop RPG rule systems, especially the d20 system popularized by Dungeons and Dragons. By the time we started to adapt the rules to work for our application, however, the Tabletop RPG aspect was dropped when we began to scale back our project. When designing a draft of the rules for our application, we wanted to avoid massive changes to an otherwise functioning rule set. In order to make as few changes as possible, we kept the following design goals in mind:

- Elements of chance could be handled through "dice rolls" using Java's native randomization functions, and thus didn't need to be changed
- Abstract ideas like "Bonus chance to hit on humanoid enemies" needed to be avoided, as the information presented to the player might not be able to clarify details (such as where the line between what was and wasn't "humanoid" should be drawn).
- Terrain penalties and bonuses needed to be avoided; while they would be easy to program into the game, they would be difficult for a non-programmer to specify through a text file,

a preferred component of our game engine

- Because we were making a single instance with specific characters, all character creation rules could largely be ignored. A developer could use other characters as a template, making their own judgment calls as to whether their particular instance needed

5 Implementation

Once all design decisions had been made, we began to actually construct the game. Below are the ways we created program based on our research and design goals. Our implementation has been broken up into four parts: Coding patterns used to create the game, conversion of art assets to match software requirements, adjustments to game rules to better fit the requirements and limitations of the code, and the development process of the actual instance used in the game.

5.1 Development of Application/Assets Used by Application

5.1.1 Code Structure

The project was developed exclusively for the Android operating systems, which affected the general structure of the code. In particular, Android's methods of handling Graphical User Interfaces (GUIs) caused the code to split into three groups: User Interface, Game State/Rules, and Data Storage. User Interface code created and managed any buttons or visuals on the screen, and took in any input from the user (such as click a button, or sliding an image). Code in the Game State/Rules section enforced the rules of the game and the transitions in the game. When a user triggered the "attack" button, for example, this part of the code would confirm that user indicated a valid target for an attack, that the user's character was close enough to that target to attack, and would perform the calculations based on the current state of the game to see if the attack was a success. Data Storage code was used to keep track of the various information that the Game State/Rules code would need to access, such as a character's statistics and inventories, what happens when a battle is won or lost, or what options a player could choose between when outside of battle. A rough layout of the code structure is shown in Figure 5.1

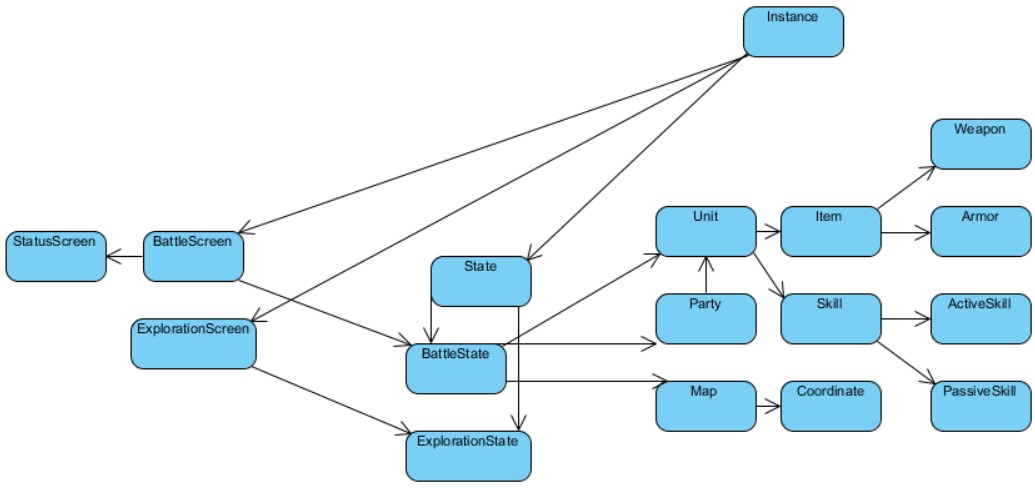

Figure 5.1: A diagram of the relationships between the classes. Arrows indicate that a class uses, stores, or is extended by the class being pointed to, with User Interface screens on the far left, Game State screens in the middle, and Data Storage on the right.

User Interface:

User Interface code consisted primarily of classes which would display a specific type of screen. These "Screen" classes would read in any images needed to display in the background or to use in menus, then generate those backgrounds and menus in specific patterns based on which state the class was meant to represent.

BattleScreen:

The BattleScreen class reads in file names for the images representing a map of the battlefield, the icons for the menus a user would use to select an action, character portraits that are both displayed on the map to indicate location and used in menus to help users associate information with a specific character, and other assorted images. The BattleScreen class then draws the map image with character portraits in corresponding locations, and draws any permanently visible information (such as each player's health or turn order). When the user indicated that they wanted to take an action, the BattleScreen class adds the menu buttons to the screen, with the appropriate images displayed for each button. Whenever a character performs an action that would affect the visuals, such as dealing damage or moving, the BattleScreen updates the visuals with the relevant information.

ExplorationScreen:

The ExplorationScreen class reads in file names to generate a background image and to generate a list of buttons, each representing an event. It also displays information to the players, such as the group's total gold. Once an event is selected, the ExplorationState passes that information on to a Game State class. Based on what the Game State class returns, the ExplorationState will either update visuals (if, for example, the party gains money), refresh all information (such as when moving from one room to another, causing the list of available events to change), or change to a different state (such as when an event starts a battle)

StatusScreen:

The StatusScreen classes reads in information about a selected character, then displays that information over two screens that a user can transition between. The StatusScreen class only updates to alternate between the two pages, and otherwise shows static text and images to help inform a player what the capabilities of the given player are.

Game State/Rules:

For most User Interface classes, there is an associated State class. Whenever the game transitioned from one type of screen to another, this would be represented in the code by changing from one State class to another. State classes would keep track of temporary information, such as file names for current images and currently available actions. Whenever the User interface needed to update what options or visuals needed to be presented to the user, they would be accessed

through the State classes. State classes would also confirm the legitimacy of an action by calling a Menu class, which would check that an option presented by a menu would not violate one of the game rules.

BattleState:

The BattleState class kept track of information directly relevant to a specific battle. This included file names for maps and menus, along with the state of each individual coordinate on the map, which characters were participating, the turn order, and the actions the current turn character could take. When a character moved or was damaged, the BattleState class would update this information, and the Screen class would check what these updates were.

CharacterMenu:

While the BattleState kept track of where units were and what actions had been taken in a turn, the CharacterMenu class checked that an available action did not violate a rule of the game, such as moving too many spaces or performing an action on the wrong type of target. In addition to confirming whether or not an action was legal, the CharacterMenu class would perform calculations to determine the success and result of an attack action.

In order to check the validity of the rules, the CharacterMenu implemented multiple algorithms. In order to quickly detect potential locations for a player to move to, a goal-less depth-limited search algorithm was used to quickly iterate through and mark all legal movement actions. Starting from the coordinate where the active character is positioned, the algorithm first marks each adjacent, legal coordinate so that the system knows to paint over those spaces on the screen. The algorithm then checks if those spaces have been visited before, and if so how long the path taken to reach that space was. If the current path is shorter, or the space has not been checked before, the search algorithm recursively calls itself starting from the checked coordinate, incrementing the distance already traveled by 1 from the previous call.

When confirming a movement action, an A* path-finding algorithm was used, which used a priority queue, sorted by the sum of the distance traveled to reach the coordinate and the minimum number of spaces needed to reach the goal. This guarantees that the shortest possible path is found.

When determining whether or not an attack is in range, a different set of algorithms are used from movement. The primary algorithm used is Bresenham's line algorithm, which is used to calculate line of sight. The algorithm determines whether there are any squares that are blocked off by a wall or another character that would overlap an imaginary line going from the player to the target location. It does so by first determining whether the "line" is longer along the x axis or the y axis. Then, for each unit it travels along the longer axis, it determines which space the corresponding segment of the line overlaps the most. If none of the checked spaces are blocked, then the algorithm confirms that the target space is a legal target.

ExplorationState:

The ExplorationState class keeps track of all events and transitions used by the

ExplorationScreen. It determines whether the conditions have been met to display an event, and passes the necessary information to the ExplorationScreen to properly add elements to the display. The ExplorationState class also applies the results of selected events.

Data Storage:

For every object in the game, whether it is a character or item, there is an associated Data Storage class. These classes hold universal variables, such as the monetary value of an item or a character's Strength statistic, and mark the presence of non-universal variables, such as a special skill associated with a particular item. Data storage is also used to keep track of specific transitions from one state to another when outside of combat.

Instance:

The Instance class holds all information on the progress of the game by keeping track of which State should be active, and which States can be transitioned to. It also constructs the States by reading in files; as a result it ends up constructing all classes except those used by the Party (as the player characters are the only information independent of a constructed play session aside from part of the interface graphics).

The Instance constructs the states based on the contents of Comma Separated Value (.csv) files. This allows users to write an instance by following a specific syntax without having to have prior programming experience. Because the Instance class constructs states, it needs to be able to also generate all information needed in those states, such as items characters may obtain, graphics to be used, and references to other data storage files. As a result, a user can control almost every aspect of the game by editing the file read in by the Instance class.

Map:

The Map class holds the layout of a map used by a BattleState class, along with image the BattleState passes to the BattleScreen in order to draw the map. The Map class also stores the current locations of characters in a battle.

Both the layout of terrain and the name of the image file used are read in through a .csv file. This means that a user can construct a map by editing the associated files.

Unit:

The Unit class holds a character's information, such as base stats (Strength, Dexterity, etc.), the number of spaces the character could move in a single turn, the maximum (and current) health and mana, which group they are part of (main character's group, an enemy group, a neutral group), what skills they have, what skills they are currently being affected by, the character's level and experience, the character's species and class, the amount of money they have, and what items they have. The Unit class also has code to update appropriate values when damage is dealt/healed, the unit levels up, or uses an item. The Units representing the players are stored in a Party class.

Item:

The Item class keeps track of the gold value of an item, and whether or not the owner of an item can use it to perform a special action. Items are also broken down into subclasses, such as weapons which also keep track of their range and damage, and armor which tracks the defensive bonuses it offers.

Skill:

The Skill class keeps track of information such as the skill level, range, and any associated costs. The subclasses ActiveSkill and PassiveSkill keep track of further information, such as any associated damage or extra effects associated with an activated skill, or the continuous bonuses from a PassiveSkill.

5.1.2 Interface Design

The menu interface for the BattleScreen is brought up by selecting the tile currently occupied by the turn character. The game detects where on the screen the user pressed, and compares this value to the portion of the map currently being displayed. If the game detects that the current character has been selected, it then brings up the menu, which consists primarily of ImageButtons, a built-in Android class. An ImageButton is a button with a modifiable picture that can respond to selection, movement after selection, and release.

The initial menu displayed consists of 6 image buttons, with one image button in the center and the other 5 forming a pentagon around the center button (refer to Figure 5.2). The buttons are positioned in such a way that they do not entirely overlap a grid space, and instead cover intersections whenever possible. This prevents the buttons from blending in with the map image or covering other important images.

Figure 5.2: Screenshot of the open initial menu in-game.

Three of the ImageButtons forming the pentagon have immediate effects on the state of the program when selected. When the End Turn button is selected, the menu immediately closes and the game sets the next character in the turn order as the current turn character. The Move and Attack options also close the menu when opened, but then highlight in red all coordinates in range, so that a user knows what legal moves they can make (See Figure 5.3). If any of these spaces are selected, and if the selected space matches the rules for the associated action, then the action is performed and the character moves to the empty space or attacks the occupied space. Otherwise, the highlight is removed and the action cannot be taken until it is selected in the menu again.

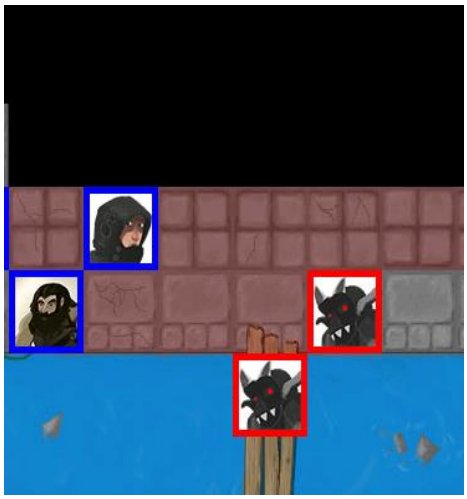

Figure 5.3: Example of the highlighted range for an attack. In addition to highlighting empty tiles, legal attack targets have their colored outline, which shows a player whether they are an ally or an enemy, brightened.

The other two buttons that form the pentagon display an inner menu when selected. This inner menu consists of up to 8 buttons; one button to return to the previous menu, and 1-7 buttons representing an item or skill to be used, depending on the menu (See Figure 5.4). When the inner buttons are selected, they behave in a similar manner to the Attack button, highlighting all coordinates in range and allowing the player to make an action against a legal target.

Figure 5.4: Example of an open sub-menu

When the center button is selected, the menu reacts to swiping motions made by the player. If the player drags their finger over one of the two buttons that form submenus, the submenus are immediately opened. If the player releases while they have dragged over one of the other three buttons, or a sub-menu button, the menu behaves the same as if the button were selected normally.

In order to close the menu, if it was opened by mistake, the user simply selects any location outside the menu.

If the menu isn't open, or if the user is currently selecting a coordinate as a target, the user can drag the map image around to look at different parts of the map. This functionality is disabled if the menu is open, and does not cancel an action when selecting a coordinate.

5.2 Development of Art Assets

When we started designing a tabletop game we were envisioning a blend between science fiction and fantasy art styles. We liked the idea behind framing the world within a technologically advanced society but still wanted the aesthetics behind a largely pre-electricity society. Specifically the game world was based on a large city, isolated on all sides by wastelands and forests. For story purposes the city was built from many different types of people, some of them futuristic robots while others were uncivilized barbarians. The game ended up having science fiction elements serving as the environmental framing device, while the fantasy influences characterized the game with monsters and characters. Although the city itself didn't make it in to the game it's presence can still be seen in the first area of the game, the waterway sewers, because we felt that masonry and plumbing was an important part of any reasonably advanced civilization.

Because the characters and monsters were initially going to be made from philosophical concepts a lot of the character design was going to involve iconic names like Plato or Socrates. We wanted to include a lot of less known names like authors and scientists, but since the game shifted to be more story specific we decided to wait on introducing these characters to the game world, as they would have just confused the narrative. They were going to be modernized in some ways, making them fit into a relatively advanced or simplified society. Specifically we were wondering how master artists would have reacted to flying machines, or how theologians would respond to a combination of religions.

When the project shifted from making an educational game to a platform for RPG's we removed a lot of these specific references, but some of the concepts still show up in the game mechanics and artwork (Figure 5.5). The Platonic Barbarian character, for example, has many different philosophy based skills, and the Mage is still a robot.

Figure 5.5: Early sketches

The enemy characters were the first concepts to become fleshed out within the game. Because we weren't thinking in terms of a specific story when we first started making the artwork it made sense to start making enemies to populate the game world. We thought that if we could establish the type of villains then we could design heroes that defeat them specifically. In the beginning these enemies were human-like (Figure 5.6), with the intention of making these enemies major sources of dialogue later when we developed the story. These were changed to be more beast-like since we thought it might visually confuse players if they were introduced to their characters and then immediately expected to fight other characters. We wanted to make it clear that the players were the heroes in the game world, and giving them time to become familiar with their character portraits might help enforce this thinking.

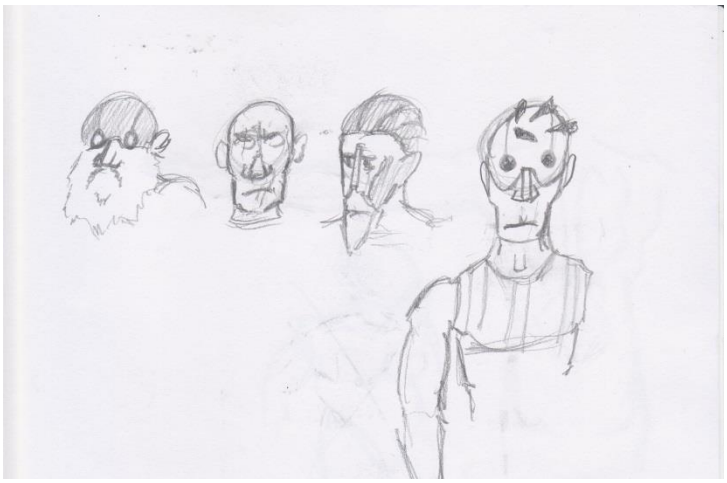

Figure 5.6: early human enemy concepts

The initial concepts of the enemies were multicolored. They were all unique elements of the world, and belonged to different environments and factions. In order to define a specific story the enemies were made to be more consistent in terms of colors. Our previous experiences with video games made us associate deep and glowing red colors with evil, so to make our enemies associate with otherworldly evils we tried to stick to mostly greyscale paintings with red eyes and backgrounds.

Figure 5.7: Final shadow priest used in games

The Worgen is the only true animal-like creature used in the final game.

Figure 5.8: early sketches of the Worgen and the Shadow Priest

Figure 5.9: Final version of the Worgen next to the original version of the firedog (unused)

The shadow monster, one of the enemies that didn't change over the course of the project, was originally thought of as a kind of demon rising from the shadows. We tried to use recognizable structures, like hands and arms, and then twist them, making them more angular and unfamiliar to make the players feel aversion and distrust. These were strange creatures that were trying to kill them, so making them inhuman was supposed to encourage them to fight the shadow monsters.

Figure 5.10: Shadow monster sketch

Figure 5.11: Final shadow monster

The first sketches were more for the benefit of coming up with concepts than anything else. They're based very much on the original tone of the game, which was to be set in a pseudo fantasy world.

The character concepts weren't finalized until making the actual final paintings. There were many concepts and ideas, but many of them didn't fit thematically with the rest of the game. Eventually we decided to focus on using pre-established archetypes of characters from our past

experience with games.

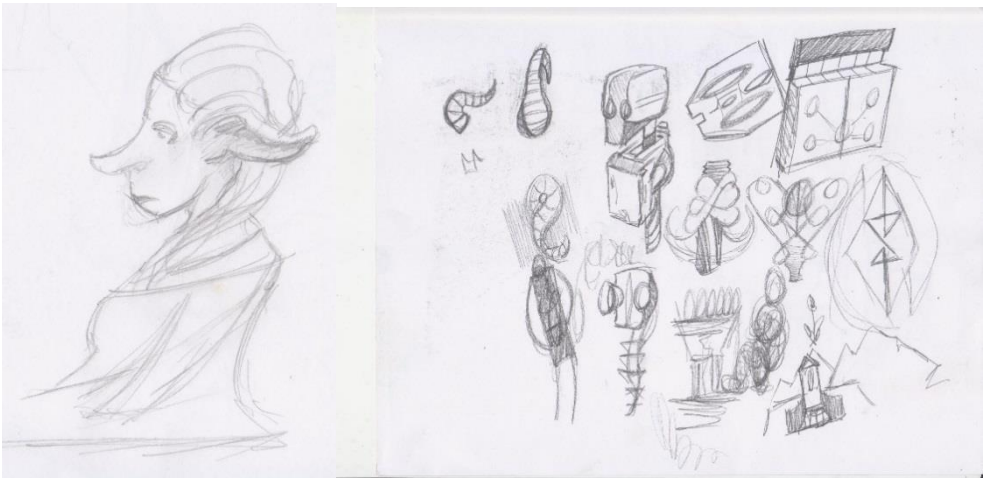

Figure 5.12: Early concept of characters and world elements

Mercenary:

The mercenary (Figure 5.13) was originally going to be a female gnome knight. The knight was supposed to act as the strongest of the character, eventually being replaced by the barbarian. Because visually the gnome was going to be stocky and muscular and we wanted to make sure each of the characters differed from each other visually, the character morphed to be more elven, taller, and slender. We eventually decided to make the character more like a rogue or an assassin, creating an archetypal contrast to the barbarian, and the mercenary was supposed to be more of a stealth attack character.

Figure 5.13 Rogue/Mercenary final painting

Barbarian:

The barbarian (Figure 5.14) did not change much over the course of the project. Since we were working in terms of archetypes we needed to deal with the concept of a barbarian. It's

difficult to replace the ingrained image of a barbarian, i.e. the hulking muscular almost naked savage figure, but our barbarian is more of a scholar than a brute. Our idea did grow out of the idea of a savage fighter, but we envisioned that this character came from a society of barbarians, and so his visual design reflects his past.

Figure 5.14 Barbarian final painting

Mage:

The mage (Figure 5.15) was meant to be part of an entire race of unique cybernetic people, each of them distinctive in their appearance. The Mage's final appearance is based on the idea of a mechanical human skeletal structure, sectioned into major body masses and adorned with a robe and a magical staff.

Figure 5.15 Mage final painting

The mage's robotic appearance is based on the initial idea of having a technologically

advanced society placed in a pre-electric world. The prototypical archetype of a mage is an old man with a long flowing beard and robes. We tried to get away from that image to try and counteract the physical prowess of the barbarian and mercenary, while still making the character appear mystical. Removing the facial features didn't actually change the human-like appearance of the character, and was meant to try and signify that this character was fundamentally different than any of the other characters.

Scribe:

The scribe (Figure 5.16) was the character added last, introduced initially as a non-player character that would serve as a guide to drive the story. He was meant to not actually be seen except in intermission or story segments of the game, and would only be used to describe some of the less interesting educational concepts. His first appearance in the game was in the form of a large portrait shrunk down to be an icon portrait, and used as a test character for the game's combat mechanics. Over time his own combat mechanics evolved to recreate him as a player character.

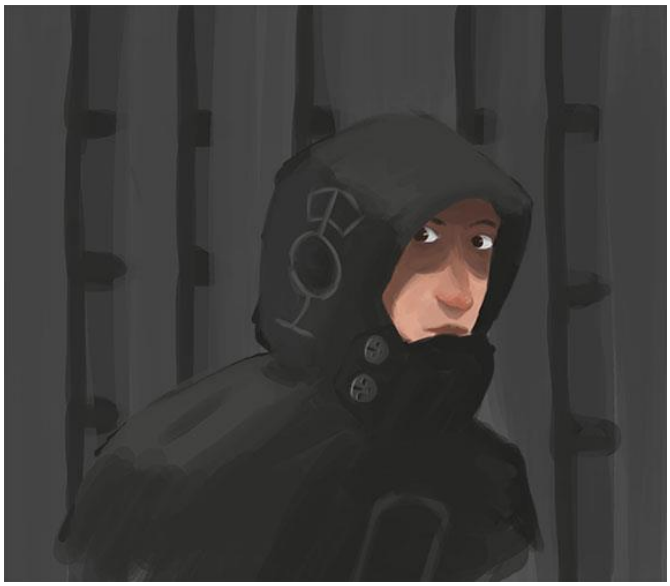

Figure 5.16 Scribe final painting

5.2.1 Game Environment

The game environments were one of the few concepts we had going into the design process. We didn't know exactly what we were going to do, but we knew we needed paintings reflecting the environments.

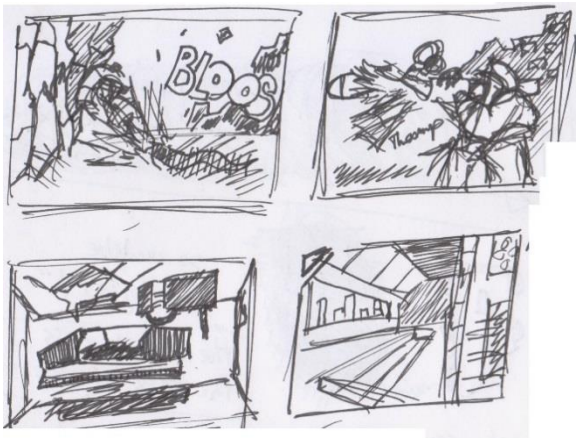

Figure 5.17: Environmental painting concepts

The paintings were meant to serve as buffer images for the next playable area in the game. Initially they were going to play a much larger part, and be much more complicated, but due to changing the focus of the project they were made without a lot of extra details.

Figure 5.18: Bar exposition "foyer" screen

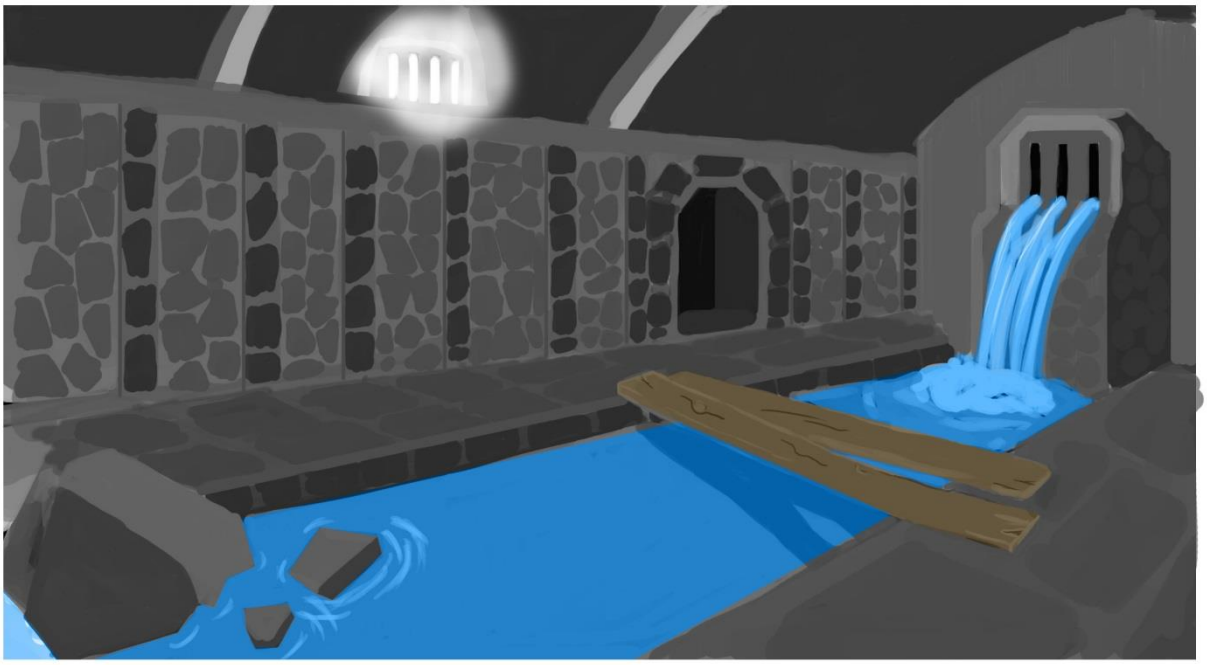

Figure 5.19: Waterway exposition "foyer" screen

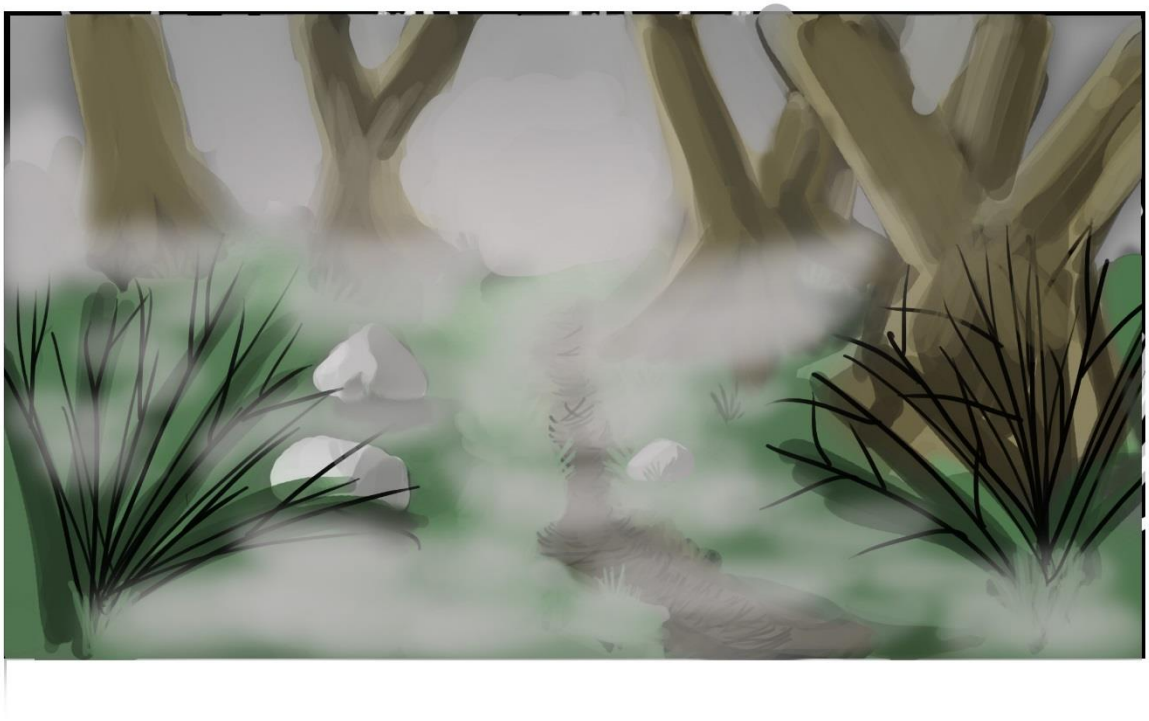

Figure 5.20: Forest exposition "foyer" screen

We designed the environment paintings at the same time as the tile-set. There was a lot of focus on creating a tile-set that fulfilled the basic needs of the story.

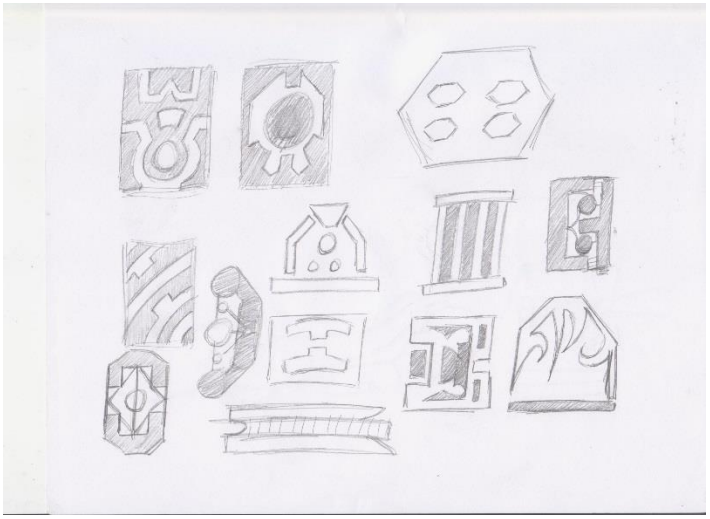

Figure 5.21: Early concepts of tile-set

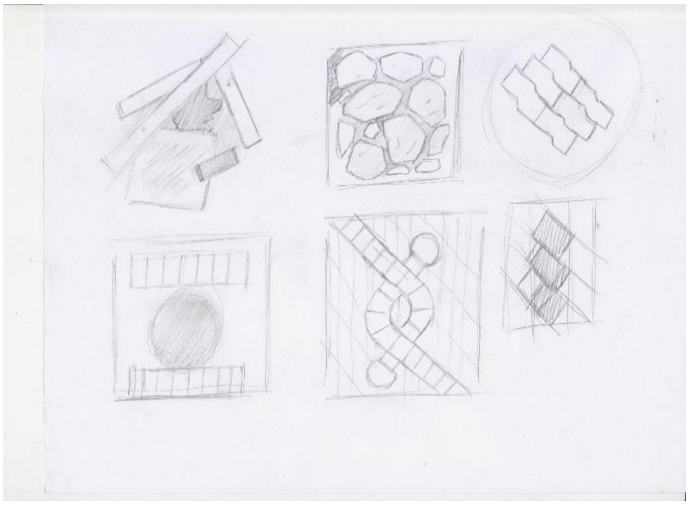

Figure 5.22: Concepts of tile-sets

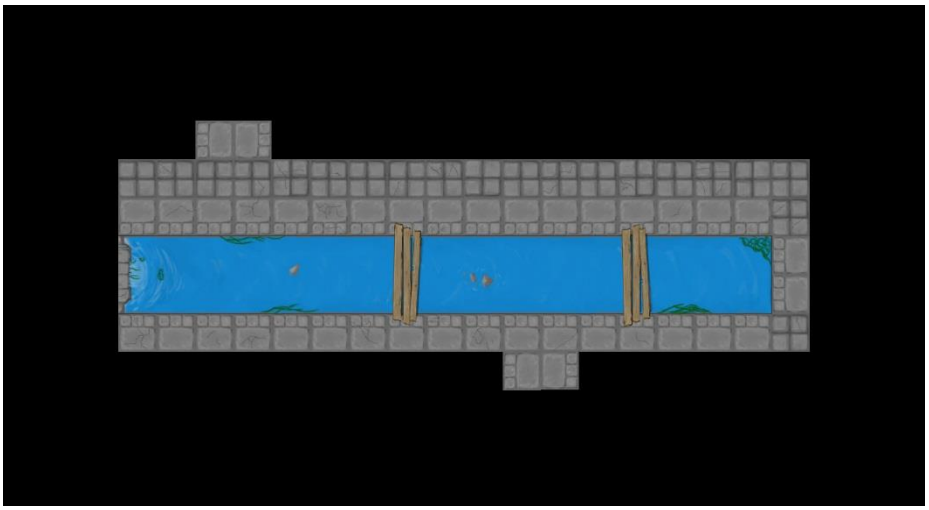

Figure 5.23: Example of tile-set

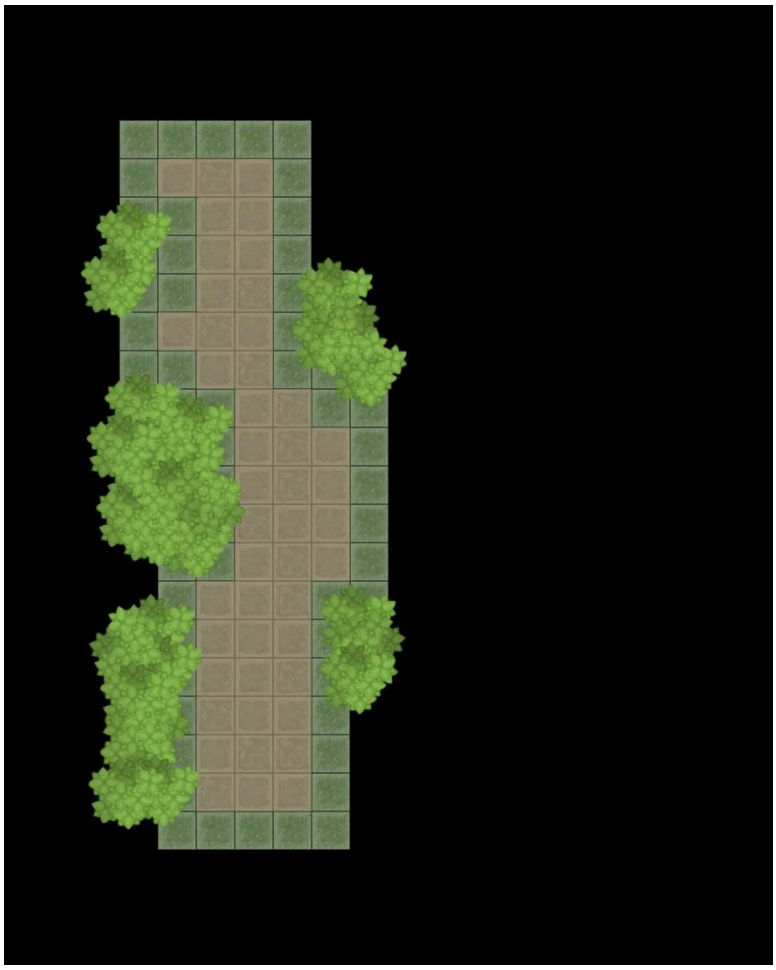

Figure 5.24: Example of tile-set

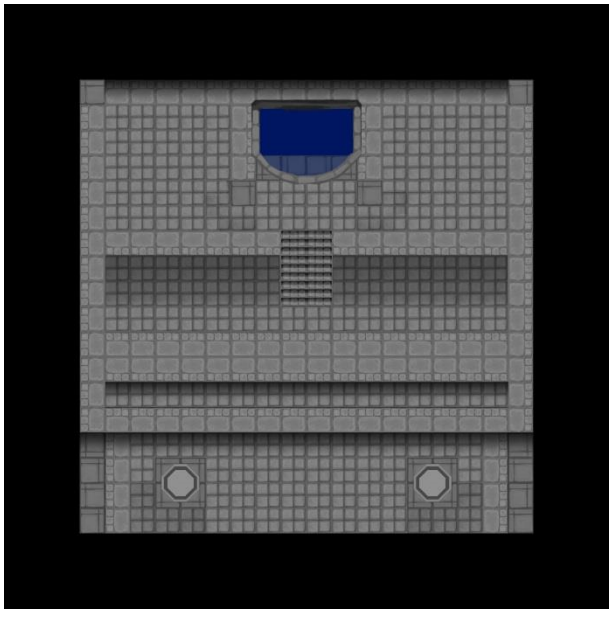

Figure 5.25: Example of tile-set

5.2.2 Game Icons

Creating game icons is similar to creating a typeface. The icons need to be visually appealing while still being easily identifiable to a wide range of people. We tried to avoid making the icons too complex to avoid players misinterpreting them, and when we tried to make them too simple we found they didn't fit visually with the rest of the game. While the style of the game, and by extension the user interface elements, can vary the actual icons still need to be congruent with each other. Really detailed pictures for an attack icon but only a black and white logo for an inventory screen might break a player's immersion, and also make it more difficult for them to play the game.

To break down the individual elements of a game UI from an art perspective, it starts with an intent. The icon needs to represent something, in our game this is summarized by calling them actions.

The icons that we need for a tabletop role-playing game are dependent on the game's mechanics. We have actions available to attack so we need an icon to reflect that. We need to keep in mind whether the players need to differentiate between the attack and defend icons regularly, or if that interaction is rare enough that we can make them visually similar. There are no basic rules to determining what icons will work for every game, and although there are a lot of standards for things like typography only some of them can be reliably used in designing a video game. Specifically we were working on a small scale, so making extremely complicated icons would make them difficult to distinguish.

At the start of the project, to get simple icons ready help us flesh out the gameplay, we started making pictures based on the following short list of actions:

Attack

Magic attack

Move

Defend

Inventory

Information

Cancel

The actual usage and format of the icons didn't become clear until after we started incorporating them into the game, so at the beginning of the project we explored different forms of each of the icons (Figure 5.26). The inventory, for example, started out as either a bag, or a chest, or some sort of backpack.

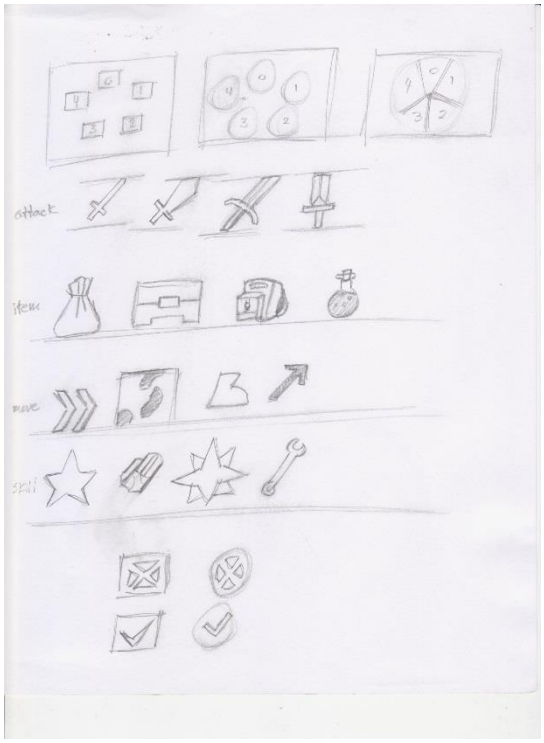

Figure 10 Early concepts for icons

Our icons are meant to be almost entirely neutral, not necessarily trying to entice the player into using that action, but still entertaining and informative enough that they will notice it. In our game the icons are actually very small, so to try and keep the visual properties and complexity in balance we need to try and make use of all the space available, but also limit the actual pictures to simple color ranges.

The first draft of icons was not very polished. They were made with a purely fantasy environment in mind, and went through several drafts of revisions. Our thinking was that due to gaming conventions it would be easy for players to distinguish the basic types of icons, and that the action icons would be distinct enough that the players would learn what skill did what simple from the image.

The first versions of the icons (Figure 5.27) were simple, and were meant to be part placeholder, part blueprint for the final icons.

Figure 5.27: early concepts

The second version of the icons (Figure 5.28) was made to provide a more clear direction for what we needed. Since we weren't decided on what the aesthetic style of the user interface was going to be specifically, these icons are bright blue to make them stand out against the game environment.

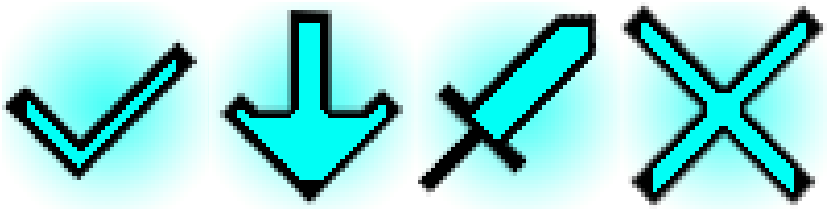

Figure 5.28: Second revision of action icons

One of the problems we encountered was that while most users could distinguish between each of the icons their use wasn't immediately clear. Either the game needed to be modified to contain a more extensive tutorial in which the user was forced to use every option possible in the game or the icons needed to be made excessively clear. Our solutions were either restricting the number of icons in the game and reducing the gameplay or adding text to each of the icons stating its explicit use. We chose creating text in pixel format to suit the very small icons. Some of the names had to be simplified in order to fit the descriptions, but eventually all the icons in the game were labeled with what they needed to convey.

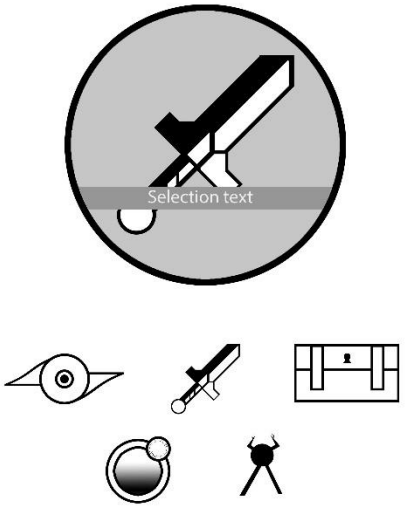

Figure 5.29: Early revisions of icons

The Icons also went through another revision (Figure 5.30). The blue backgrounds changed to red because there was an abundance of water in the game and we felt the blue icons weren't visible enough. The icons also because more illustrated, to fit better with the overall style of the game.

Figure 5.30: Final version of icons used in game

A red border was also added to further increase visibility of the icons in the game. Since the icons are so small, and adding a border in the application would change the layout of the icons, we made the red borders a 2 to 3 pixel wide box on the actual image itself. Although the icons already had a border, that border was grey, which made it difficult for some users to distinguish what was a piece of the tile-set and what was an icon.

The first real set of action skill icons (Figure 5.31) were for the Barbarian character. This presented a new challenge in that the Barbarians skills are of mostly ethereal or conceptual basis. We tried to approach each icon with the mindset of making the icon fit the intended use.

Figure 5.31: barbarian skills icons

We then had to expand on these skill icons to include the other characters. These proved to be quite a bit more concrete, ranging from magic spells that could be rendered for their effect to actual actions like stabbing or using a specific weapon.

Below are the mercenary/rogues skills.

Figure 5.32: mercenary skill icons

The mercenary's action icons are each meant to display the physical act of stabbing or moving. The use of the color green on some of the images is to represent poison because it's a very subversive color. It's bright, unnatural, and with high enough contrast with the rest of the icons it suggests there is a particular use for this skill. The stealth skill is purposefully very dark, meant to compliment the feeling of needing to use stealth-based combat.

Below are the Scribes abilities, which are rooted in a loose magical system where the scribe will draw the symbol in the air with his hands and then make the intended effect happen

through force of will.

Figure 5.33: Scribe skill icons

The scribe wasn't originally meant to be a playable character, and so his skills are more generic than the mage, barbarian, and mercenary's actions. Each of the scribes skills are also meant to reflect the method in which the spells are cast, by moving his hands in the air.
Lastly, the Mage abilities, which are all somewhat gravity related.

Figure 5.34: Mage skill icons

Each icon incorporates the color blue in some way. The reasoning behind this was that blue is a mystical color, and the Mage is harnessing some otherworldly ability. The mental shove ability was again limited to simply showing a hand.

5.2.3 3D models and printing

We also started working on a physical analogue to go with the application in the form of 3D printed models that players could use to manipulate and keep engaged in the game even when it wasn't their turn.

Initially this meant we were only going to create 3D models of the characters (Figures 5.35, 5.36), but as the project progressed we started evaluating different methods to create physical game tiles that could be arranged by the players into the form of the game maps.

Figure 5.35: 3D model of character to be printed

Figure 5.36: Printed model

To create a set of tiles to be arranged on a table, we went through several different revisions of designs. The first set of tiles was interlocking:

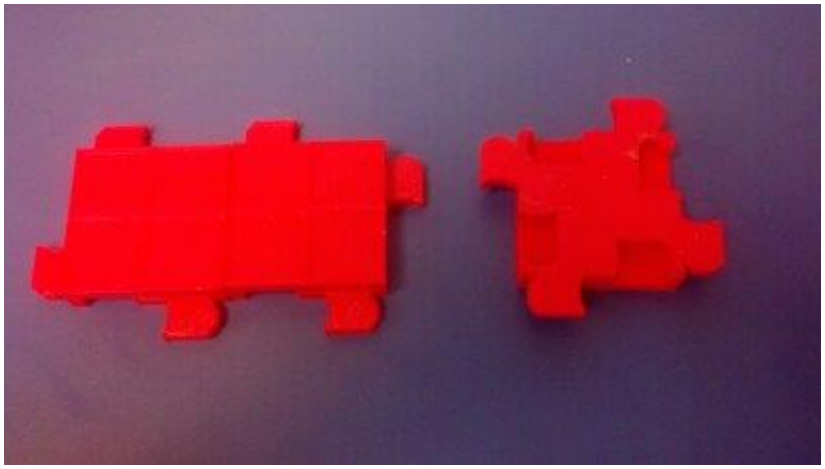

Figure 5.37: Interlocking tiles

Our second set of revisions was also interlocking, but we tried to add more of a vertical element, creating walls and doorframes that could be arranged:

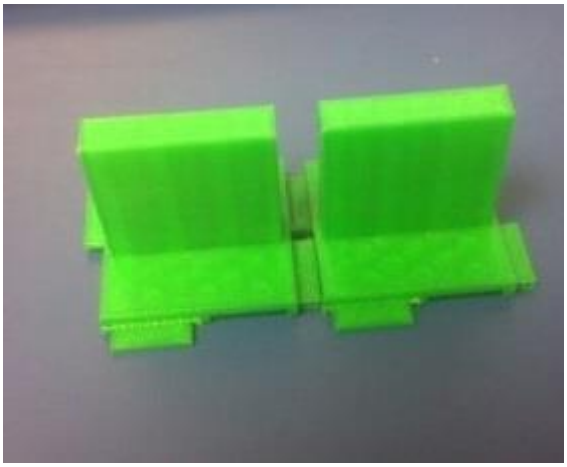

Figure 5.38: Interlocking wall tiles

Eventually we decided to drop the interlocking features, and focus on making the tiles easily printable. We made two different types, a floor tile and a wall tile that is significantly shorter than previous versions, because we felt that players only needed confirmation of walls, not physical walls.

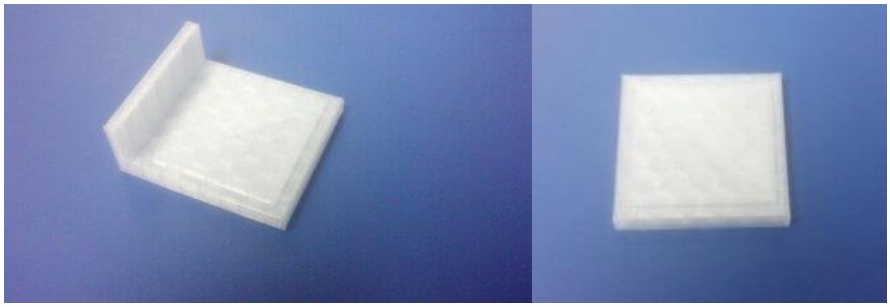

Figure 5.39: printed tiles

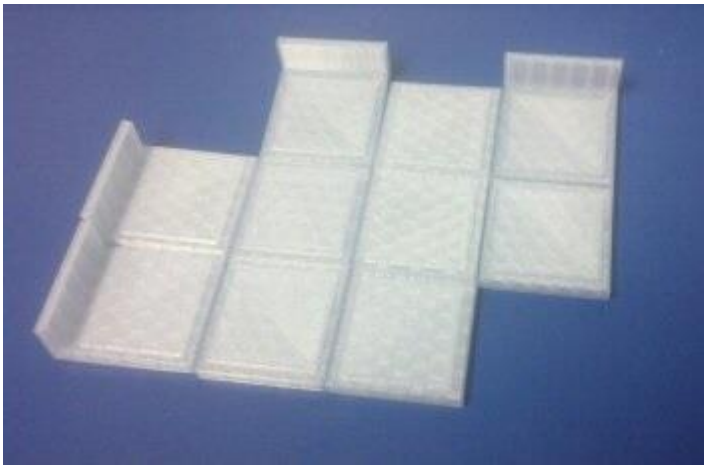

Figure 5.40: Example of several tiles arranged to make a board

5.3 Constructing an Instance

Creating a game using our engine is, in simplified terms, a matter of writing a text file to be read by the program and providing the properly named images to be loaded into the game. In order to create a *good* instance, however, a user needs to keep in mind good principles of game design. Our instance is meant to be an example of the potential of our engine, meaning that our game needed to be small-scale (and as a result required a simple design). However, when we created our sample game, we still needed to keep in mind those good principles of game design, which allowed us to craft an outline for our game.

5.3.1 Guiding Principles

- A sample game needs to be short, showing off a few potential aspects of the engine without relying on complexity and unique events to attract players.
- Sample games should be easy to play through; a challenging sample will be unable to show all desired features to a low-skill player. If there is a difficult section, it needs to be at the very end of the game, to ensure that the rest of the game gets seen.
- The players should not start with the weakest form of characters. If a character gets upgrades or unlocks actions such as spells, they should have access to some of these upgrades or actions from the start. This allows the player to see what standard gameplay will be like, as the player will likely have access several skills for the majority of the game.

5.3.2 Difficulty Curve/Pacing

The following section explains the thought process behind each battle in our sample game.

- **First Battle:** This battle should be the easiest, to allow the player time to adjust to how the game works. There should be a small number of enemies who effectively act as punching bags; this will allow the users to determine how effective certain tactics are without putting themselves in any real danger. Players should expect to take a very small amount of damage, allowing them to get an idea of the level of threat posed by enemies without being put at any real risk
- **Second Battle:** This battle should feature a stronger single enemy, who is about as easy to defeat as the multiple weaker enemies. This allows the player to test how the tactics learned in the previous battle work against a different type of enemy. This larger enemy should be capable of defeating the players, but only if the players perform at the worst possible level. Even below-average skill players should have no trouble defeating this enemy.
    - Players are given new equipment as a reward after this battle. This establishes that characters will grow in power over the course of the game, creating the expectation of further upgrades in players. This expectation encourages players to keep playing
    - Players should be healed after this battle, so that they aren't punished for any risks they took while learning about the game. This also allows us to balance the next set of battles with the assumption that players across all games are on equal footing at this point.
- **Third and Fourth Battle**: These battles should feature stronger enemies than the first battle, with noticeable strengths and weaknesses. This encourages players use tactics based on these strengths and weaknesses, rather than picking the first tactic the player thinks of. This encouragement demonstrates to the player that the game rewards players who are better at adjusting their tactics, which is an important part of Tactics RPGs. The battles should still be incredibly hard to lose by mistake, but players should expect to take non-negligible damage in order to further reward players for optimal play, even when at no risk of losing.
- **Fifth Battle:** This battle should act as a boss fight, and be the first battle to actually pose a noticeable challenge to the players. Enemies should make use of the strengths and weaknesses demonstrated in the previous battle, and should either appear in noticeably higher numbers or pose a greater threat individually to the players. If a player is consistently playing poorly, there should be a chance of defeat, but the odds should still be in the player's favor, even if the player has low skill.
    - Players are once again healed and given equipment after this battle, in preparation for the final battle. This prevents a struggling player from reaching the final battle, and being unable to complete it because they started out in a near-dead state.
- **Final Battle:** This battle should be the only battle that has a reasonable chance of beating a low-skill player. This allows a player with average or above-average skill to demonstrate

the advantages of good tactics. At least one enemy should pose a high individual threat. This means that a player who has discovered ways of crippling powerful enemies can be rewarded, as many of the other battles have featured larger numbers of weaker enemies which those tactics would be less effective against.

6 Evaluation

Once our menu systems and game were implemented, we then proceeded to test their effectiveness and make adjustments as necessary. Below, we describe the process used to test our games, the direct results of those tests, and any changes made to our application as a result of those tests.

6.1 Usability Test

Below are the results from our usability tests. See Section 3.4.1 to review how the usability test was conducted.

Table 6.1: Average Time Taken per Task in Seconds and Accuracy of Answers for Usability Tests

Question/Task	**Average Time/ Success Rate**	**Average Time (Gamer)**	**Average Time(Non-Gamer)**
Open the Turn Menu	4.94*	2.93	7.69*
Attack a Unit	2.66	3.07	2.06
Identify Ally/Enemy	80%	67%	100%
Estimate Change in Health (Accurate Statements)	100%	100%	100%
Use “Strike” Skill on Unit	10.9*	6.18	18.00*
Estimate Change in Health	100%	100%	100%
Estimate Change in Mana	100%	100%	100%
Move Character to the Right	2.716	3.04	2.51
Determine Next Turn Character	100%	100%	100%
End Turn	2.86	3.28	2.23
Move Map	100%	100%	100%

Times marked with an asterisk (*) are likely inaccurate due the presence of a single statistical outlier in a small data set

6.1.1: Interpretation of Data

The next few sections explain how we interpreted the data, and what we did (or would have done if we had more time/manpower) based on the result of each question.

Open the Turn Menu:

Many users were able to figure out how to open the turn character's menu immediately, while some tried to tap on the pictures of the character used to display health and turn order first. Once a user properly opened the menu, they had no problem doing so as necessary for other tasks, and every user was able to figure out how to open the menu without further prompts. While, on average, non-gamers took longer opening the menu, this is largely because of a statistical outlier, and aside from this data point the average times between the groups were very similar.

Problems: Multiple identical images of the character are present on the screen, which lead to confusion during first use.

Solution: When designing the tutorial screens, instructions state which image to click, and the accompanying image displays the character on the map, and does not include any images of the side menu. Users were able to determine proper inputs on their own without the tutorial, so in-game changes do not appear to be necessary.

Attack a Unit:

This tested whether the user could identify and properly select the intended icon, whether they intuitively understood that target selection was done via tapping once the menu closed, and whether the character images were clear enough that distinguishing characteristics could be identified at small-scale. All users were able to complete this task fairly quickly, with little difference between gamers and non-gamers. This task was, however, made slightly easier than intended because the unit was described. Users did not need to be able to understand visual cues during target selection (e.g., associating the red overlay with attack range, associating colored borders around characters with viable targets, etc.) since they were instructed to select a specific target without having to identify whether this would be a legal move.

Problems: No notable problems.

Identify Ally/Enemy:

Once users had finished attacking the barbarian, they were asked whether the selected unit (who was an allied unit) was an ally or an enemy. When attacking, units in range of the attack have a blue highlight around them if they are allies, and a red highlight around them if they are enemies. While some users were able to confidently answer the question, others were unsure, and at least one subject stated that they had to blindly guess in order to answer the question. Some of the users that *were* able to determine that the barbarian was an ally used visual cues other than the highlights, specifically by inferring that the four characters shown on the side menu (which include the turn character and the barbarian) were the player characters, and thus allied. This may have been a result of the ease of the attacking task as explained above, since users would immediately select the targeted unit without trying to comprehend the information being displayed during targeting. Regardless, this question demonstrated that information about which side a character is on needs to be displayed outside of targeting.

Problems: The intended method of differentiating allies and enemies is ineffective,

especially when not targeting. The problem is worsened if the user does not regularly play videogames, and cannot rely on established RPG conventions to differentiate allies and enemies using other methods.

Solutions: Add a weaker version of the highlighting to icons regardless of whether an attack or skill command has been selected, and make icons more consistent between allies/enemies. In addition, emphasize the existence and meaning of the indicators in the tutorial.

Estimate Change in Health

While there was no marking to represent a full health bar, users were able to compare the missing health with the size of the mana bar below it, which had an equal maximum length. All users, regardless of past gaming experience, had consistent estimates for the health lost, implying that they were able to associate the red bar on the side menu with the health of the unit displayed above it.

Problems: While no users had difficulty determining the percentage of health a character had, users may have trouble determining these ratios if the mana bar is not full.

***Solutions*:** We have pre-emptively adjusted the health bar to better match the length of the character portraits, in order to have a consistent reference point

Using Strike Skill on Target:

This task generally took significantly longer than simply attacking the unit, which could be to the result of having to search through an additional menu. Users also sometimes mentioned that they were unsure whether or not they successfully used the skill; those users also noted the lack of feedback for attacking, but found it particularly odd with skill use.

In one extreme case, a user had significant difficulty selecting a skill. This is in part because of their selection technique: the user was trying to select the menu option, which was on the upper left hand quadrant of the screen, by stretching a hand supporting the tablet from the opposite corner. This indicates that menu Solutions buttons may need to be larger to aid detection

Problems: Longer selection process than most other tasks, small button sizes (uncommon but significant occurrence), lack of feedback.

Solutions: Use clearer icons to aid identification; increase button sizes; add a text box that updates upon attacking, skill use, or turn end to provide visual feedback.

Estimate Change in Health:

Users did not have any problems reading the change in health after the use of a skill.

Estimate Change in Mana:

Users were able to correctly identify that the blue bar underneath the health bar represented mana, and that it did not change. No user tried to guess a random number, or stated that they were unsure, implying that users recognized that mana was being displayed, and that users did not feel the need to guess blindly.

***Problems*:** None noted.

Move Character to the Right:

Only one user did not do this task incorrectly, as they accidentally selected the square to the left of the one they intended. While the user made an error during input, they were still able to identify what the correct input was, and attempt it.

Problems: Users would occasionally fail to be precise enough in their inputs. The precision the program required lead to uncommon but significant errors.

Solution: Require confirmation of selection (Would have been implemented, but project ran short on time).

Determine Next Turn Character:

Users were asked which character they thought had the next turn. While the turn order was different for each user, all users were able to determine who went next based on the turn order indicator on the side menu.

***Problems*:** None noted

End Turn:

Users were asked to end the current turn. No users had trouble performing this task, although one user noted a concern that ending the turn was too easy, and might be done by mistake.

***Problems*:** Possible to accidentally end turn pre-maturely (While multiple testers were concerned with this possibility, this never occurred during testing)

***Solution*:** Require confirmation of selection (Would have been implemented, but project ran short on time).

Move Map:

Users were asked to scroll the map as far to the right as they could. All users were able to do this easily, and many of them determined the proper inputs through unprompted experimentation between tasks or before the first task was explained.

6.1.2 Post-Experiment Comments

After the experiment was done, users were asked if they particularly liked or disliked anything about the interface. Below are a list of pros and cons given by users:

Pros:

- Functions smoothly.
- Easy to use.

Cons:

- End turn icon was less clear than the other icons.

- Leaving the application was too easy (specifically using the back button built into android devices).
- Hard to determine the statistics of non-turn units (including information the player is meant to have access to).
- Lack of health indicators for enemies.
    - While the lack of health indicators was originally intentional to match conventions of Dungeons & Dragons, we later changed the game to match the conventions of Tactics RPGs.
- Not enough feedback on the map itself.
- Health bar might be better if green (based on alternative conventions).

6.1.3: Adjustments

Once initial changes to the program were made, the app was shown to one of the subjects who had raised several valid concerns during initial tests, and who was the source of some outlier data. When given the updated app, the subject felt that all needed changes had been made, aside from a small suggestion to include icon images in the status screen.

Final List of Changes Made to the Program based on All Suggestions:

- Alignment-based highlights around each character's icon on the map were added, even when no one is targeting.
- Textual feedback was overlaid on the map. This text was updated after attacks (and included whether they hit or missed), skill use, and ending a turn.
- Slight adjustments were made to the status screens.
- Players were allowed to check status of allies and enemies regardless of whose turn it was.

6.2 Playtests to Determine Effectiveness

Below are the results from our usability tests. See Section 3.4 to review how the usability test was conducted.

The playtest lasted approximately two hours, ending before the subjects reached the end of the game. While the playtests helped us learn numerous things about how to make the game better, a few general concepts stood out:

- Since this is intended to be played as a social game, expectations for the game need to take into account inefficiencies from multiple people playing. In particular, a group game like this needs to be made easier to account for multiple, sometimes conflicting, tactics being

used. It also needs to be made shorter due to the down time that occurs as each player processes an action another player made.

- As it currently stands, the game drags on due to the mechanics involved. Players want to use skills as often as possible, especially when enemies have high health and high evasion rates. As a result, players would run out of mana well before a fight would end. Even if they won the fight, that fight took significantly longer than players wanted it to.
- More information needs to be easily available to a player. Displaying certain skill information, such as damage or mana costs, only on the skill screen is inconvenient for players. The players are more likely to guess their way through the game than taking the time they need to learn what their characters can do.

The information gained during the playtest helped us to establish a list of things we would do to improve the game, both to make the interface easier to use and to make the game more enjoyable.

6.2.1 Format of the Playtest

Four volunteers were given a tablet with the game installed, and asked to play the game, with one of the project members acting as the enemies. The game began with a tutorial, which one of the players read to the group out loud in full, without skipping pages. The instance that the volunteers were given had almost no exploration aspects and consisted almost entirely of combat, with a paragraph of text explaining the plot between battles. The players were told before the playtest started that they would each need to choose a character. They were reminded of this a second time right before they started their first battle, as only half the players had selected a character during the tutorial. While it was explained that the tablet would need to be used by every player, the players themselves decided the manner in which they shared the tablet or read information on it.

Each player took turns for their character, occasionally asking about details of the game. When appropriate, these questions were answered by a project member, either directly or by pointing the player towards an in-game reference tool. In addition to answering questions, the project group noted any comments the players made in regards to game balance, interface concerns, or problems with understanding the game. In addition to recording those comments, some of the volunteers were asked a series of questions about the game after the playtest ended. While individual volunteers were asked the questions, other volunteers were nearby and would offer input related to the other players' answers.

6.2.2 Specific Responses from Playtest

- Players wanted information on the accuracy and rate of success calculations for regular attacks.
- When exiting a menu, players wanted the game to provide feedback as to why the menu was exited. This was in part caused by an unintended functionality of the game, which

would allow players to check the range of a skill when the player was out of mana and unable to use that skill. Attempting to use the skill would cause the program to behave as though an invalid tile was selected, which is an intended method of closing the menu.

- Players wanted to move a few spaces, perform another action, and then move again with a reduced maximum distance (Characters are only allowed to make one movement action per turn for game balance reasons).
- Players were unaware that there was a system in place to perform multiple offensive actions each turn based on the "length" of those actions. This warrants a clearer explanation during the tutorial.
- Players disliked the low success rate of regular attacks and the high mana cost of skills. An over-reliance on regular attacks with a 50% success rate frustrated players, since half the time a player who was out of mana wouldn't be able to perform a significant action. The low success rate also made the game run significantly longer than intended.

6.2.3 Adjustments Based on Responses

- The regular attack success rate calculations were changed in order to make hitting a target significantly more likely. A character with default stats (no bonuses or penalties) attacking another character with default stats now has a 75% chance of hitting, rather than a 50% chance.
- The menu interface was debugged and adjusted, so that it would be clearer to players whether they could perform an action.
- A section was added to the tutorial explaining action "lengths", so that players better understood why they could only use certain skills before or after attacking.
- The layout of status menus was adjusted to help readability.

6.2.4 Future Adjustments Beyond Immediate Scope of Project

- Add confirmation windows to certain actions
- Further improve scaling in the menu interface between advices

7 Results

Based on our conclusions across evaluations, we are able to make informed judgments about how successful our game was. Below, we go into detail about the large-scale successes and failures based on our interpretations of collected data.

7.1 Usability Evaluations

Based on evaluations during the initial usability tests and playtests, our game has good usability in terms of task performance, but weak usability in terms of clarity of information and readability. Throughout testing, users were able to quickly and easily perform the tasks expected of them, with minimal error or confusion. Situations in which users had little prior experience with games or were only able to use the application infrequently (such as during the playtest) had little negative effect on the user's ability to complete tasks. Users also repeatedly stated that they felt that the interface was straightforward and easy to understand. The main criticisms with our interface were about how information was displayed. Subtle design hints would frequently go unnoticed unless they were specifically pointed out or drastically altered, and detailed explanations in-game were often either ignored or found to be confusing.

Further improvements to the usability of the app should be focused on clearer layouts. In particular, the status screen should likely be redesigned from scratch, to allow more space for detailed information. As it currently stands, the information is displayed as large amounts of black text on a plain white background, which is discouraging for users who want to quickly find the information they want. Status screens should be designed under the assumption that the successes of the menu interface can be carried over, and that less information needs to be displayed at once. If menus or buttons could be added to alter the layout of the status screen dynamically, this would free the design from many crippling space constraints, and allow for a better-formatted description of a character's skills and statistics.

7.2 Asset Evaluation

Feedback acquired during playtests about art assets was mostly positive. Players could identify what was and was not an icon, could distinguish the icons from the tile-set, and tell the players apart from the enemies. Players also responded positively to the different coloration of enemies and player characters, especially taking note of the use of grey-scale on the enemies. What players did have difficulty with was reading the text on the icons. This could be due to size or visibility, but there is no clean solution to making the icons more readable. The text can't be made larger because of pixel width constrictions, and making the entire icon larger would cause problems with the application.

7.3 Gameplay Evaluation

Based on feedback during our playtest, we were able to demonstrate proof-of-concept (that our game rules had the potential for deep, meaningful choices during gameplay), but that our example game would need significant tweaking in order to be considered a solid standalone game. This conclusion is reasonable given that our project focused more on establishing a valid and reusable game engine than the game itself, but it also means that there is significant room for improvement. In order to generate a good standalone game as a representation of what our game engine can do, we would need to do more frequent, iterative gameplay testing with multiple styles of play. While the easiest solution would be to repeatedly balance our game around the frequent use of skills that inflict status penalties on enemies in order to reduce damage taken while increasing mana efficiency in regards to damage, this would likely cause our game to run into the same problems we had during our playtests: players did not want to use the tactics we had in mind, and the developers' and players' goals would run counter to each other. To deliver on a good game in the future, our playtests would need to incorporate ways to measure what the players' desired tactics are.

7.4 Conclusion

Overall, we believe our game and game engine was successful in its initial goal: to create a game capable of meaningful narratives and decisions, with an easy-to-understand interface that works on mobile devices. While it is possible to improve on our end result, the game we developed has demonstrated good elements of design.

8 References

Briclot, A., Koch, M., & Moris, J. (2013). *The Art of Remember Me*. : Dark Horse.

C. Chrishna-Pillay (2003) Personal communication.

Card, S. K., Moran, T. P., & Newell, A. (1980). The keystroke-level model for user performance time with interactive systems. *Communications of the ACM*, *23*(7), 396-410. doi:10.1145/358886.358895

Diaz, A. (2009, January 27). Dresden Codak » Archive » Advanced Dungeons & Discourse. *Dresden Codak RSS*. Retrieved May 1, 2014, from http://dresdencodak.com/2009/01/27/advanced-dungeons-and-discourse/

Dungeons & Dragons Roleplaying Game Official Home Page. (n.d.). *Dungeons & Dragons Roleplaying Game Official Home Page*. Retrieved May 1, 2014, from http://www.wizards.com/dnd/

Final Fantasy Tactics: The War of the Lions (iOS). *Metacritic*. Retrieved December 2, 2013 http://www.metacritic.com/game/ios/final-fantasy-tactics-the-war-of-the-lions

Garrett, J. J. (2011). *The elements of user experience: user-centered design for the Web and beyond* (2nd ed.). Berkeley, CA: New Riders.

Gurney, J. (n.d.). Gurney Journey. *Gurney Journey*. Retrieved May 1, 2014, from http://gurneyjourney.blogspot.com/

Krug, S. (2006). *Don't make me think!: a common sense approach to Web usability* (2nd ed.). Berkeley, Calif: New Riders Pub..

Le, K., & Yamada, M. (2005). *The skillful huntsman: visual development of a Grimm tale at Art Center College of Design*. Culver City, CA: Design Studio Press.

Raymond, E. 2005. *The Art of UNIX Programming*. [e-book] Pearson. Available through: http://www.catb.org/esr/writings/ http://www.catb.org/esr/writings/taoup/html/ [Accessed: 18 Nov 2013].

Schulz, T. (2008). Using the Keystroke-Level Model to Evaluate Mobile Phones. In *proceedings of IRIS31-The 31st Information Systems Research Seminar in Scandinavia* (pp. 10-14).

Stone, D. L., Jarrett, C., Woodroffe, M., & Minocha, S. (2005). *User interface design and evaluation*. Amsterdam: Elsevier :.

Sugarbaker, M. (2012, January 14). What is a role-playing game?. . Retrieved May 1, 2014, from http://www.gibberish.com/archives/2012/01/14/what-is-a-role-playing-game/

Supergiant Games | Bastion. (n.d.). *Supergiant Games Bastion Comments*. Retrieved May 1, 2014, from http://supergiantgames.com/index.php/media/

The Hero's Journey Outline. (n.d.). *hero's journey*. Retrieved May 1, 2014, from http://www.thewritersjourney.com/hero's_journey.htm

9 Appendices

Appendix A: Rule Set Draft

Note: Our original draft of the rules featured blue highlights intended to be used in a hyperlinking system. This hyperlinking system was eventually scrapped as being overly complicated and hard to implement, with little benefit to the player.

Core Rule Set

Character Stats:

Health Points

- Represents how much damage a character can take before they die.
- Abbreviated as HP, often written as current hp/max hp (61/103 means that the character has 103 max hp and has taken 42 points of damage, meaning that if they take 61 more points of damage they are killed)

Mana Points

- Represents how many magical skills or spells a character can use before resting/recovering
- Abbreviated as MP, often written as current mp/max mp

Will Points

- Represents self-confidence and mental health, running out causes the character to have a mental breakdown and either stop acting or stop acting rationally.
- Abbreviated as WP, often written as current wp/max wp.
- A character can have a negative number of will points, requiring more will points to be recovered before the player can control the character again.
- Characters will automatically begin recovering will points after an extended period of time has passed (a character will regain 10 Will points after 10 turns in **combat**, or 5 minutes of in-game time (time spent by the characters, not the players themselves) outside of combat.

Another character may try to reason with the character during combat, which counts as a **small action**, in order to restore 10 Will points.

- Penalties for running out of Will Points should be chosen at random (random selection might be determined by rolling a die with a number of sides divisible by 4, so using a 20 sided die a Dungeon Master may decide to use 1-5, 6-10, 11-15, and 16-20 to represent different penalties). The following are some examples, but a Dungeon Master may feel free to create additional penalties.
    - The character cannot act until they recover Will points.
    - The character will run in a random direction until they recover Will points.
    - The character will **attack** whoever is closest to them (with the Dungeon Master deciding on how to move if multiple paths lead to the same target, and breaking ties if two targets are equally distant). This will cause combat to begin if the player loses will points while outside of combat.
    - The character will use a random **skill** (the skill should be chosen by the Dungeon Master, either by dice roll or an arbitrary method if the number of skills a character has is not convenient). If the skill has to target a character or area, the character uses the skill on the nearest target in a randomly chosen compass direction, or if it has to target an area, then the character uses it on the first enemy in a randomly chosen compass direction or at the maximum range of the skill (if no targets are in range).

Movement Speed

- Represents how far a character can move over a short period of time. This is not the character's maximum running speed, but instead how much distance a character can *easily* cover without tiring themselves out, while still maintaining control. As a result, shorter characters who are more nimble may have a lower base movement speed as a result of gait size, but their nimbleness may give bonuses to movement speed in the form of skills
- Movement speed is measured in spaces on a grid. A typical value for movement speed may be 6 (tiles that the character can travel per turn).

Base stats

STR (strength)

- Represents physical strength, endurance, constitution.

- Strength is the statistic that quantifies physical ability, including how much the player can carry in terms of **weight**, how many **health points** they have, how much **damage** they do/receive, and how likely they are to hit with blunt weapons.
    - Also used for skill checks for activities where weight would be a detriment, such as climbing, swimming, jumping over obstacles, etc.

AGI (agility)

- Represents coordination and reflexes.
- Agility is the statistic that quantifies physical dexterity, including how likely the character is to hit with slashing/stabbing weapons or physical projectiles, and how likely they are to dodge attacks.
    - Also covers skill checks focused on fine motor skills or tactile sensitivity, such as lock-picking, pickpocketing, checking for traps, etc.

INT (intelligence)

- Represents logical ability and factual knowledge.
- Intelligence is the statistic that quantifies a character's ability to understand information, including how many spells or skills the player may learn, as well as accuracy for magical or abnormal projectiles (such as boulders pushed down a hill).
    - Also covers skill checks for identification of objects, understanding of machinery, logical puzzles, etc.

WIS (wisdom)

- Represents lateral thinking , awareness, and decision making
- Wisdom is the statistic that quantifies how well the character can make judgment calls, including how well a character can analyze enemies (and learn their stats, attributes, or tactics mid-combat), how likely they are to succeed at affecting an opponent with illusionary skills or spells, and ignoring sleight-of-hand or diversion skills
    - Also covers skill checks based on observations, such as spotting enemies or items, and evaluating the intentions or honesty of someone with whom they are talking.

SPR (spirit)

- Represents mental constitution

    - Spirit is the statistic that quantifies determination, and determines how many **mana points** a character has, and how likely they are at resisting psychological skills such as those causing fear or panic.
        - Also covers skill checks based on concentration, and any bonuses or penalties are added to any skill check on a prolonged action, such as adding on to strength if a weight is being carried over a long distance. Spirit-based bonuses or penalties are added to a skill check that is causes noticeable discomfort, such as adding on to agility bonuses and penalties if picking a lock that is covered in barbs.

- Each point of a base stat above 10 adds +1 to relevant skill checks, and every point below 10 subtracts 1 from relevant skill checks

Character Creation:

- When creating a character, each player chooses one **race**, one **class**, and one **class modifier**.
- Each of these selections will have an impact on the **skills** the player character has
- In addition, each player is given 24 points to invest in **base stats**, with the character's base **strength**, **agility**, **intellect**, **wisdom**, and **spirit** each equal to the number of points invested in the particular stat + 8.
- Each character starts with 100 **health points**. If they have more than 10 points in the strength stat, they have an additional + 10 health points for each point of strength above 10 (for a total of 10 * the strength stat)
- Each character starts with 100 **mana points**. If they have more than 10 points in the spirit stat, they have an additional + 10 mana points for each point of spirit above 10 (for a total of 10 * the spirit stat)
- Each character starts off with 100 **will points**. If they have more than 10 points in the spirit stat, they have an additional + 10 will points for each point of spirit above 10 (for a total of 10 * the spirit stat)
- Each character also starts off with **weapons** and **armor** based on their **race** and **class**.
- Each character starts at **level** 1, with 0 **experience points.**

Character Growth:

- Characters gain additional skills and points to invest in base stats by **leveling up**. Leveling up occurs when a character has accumulated 100*(their current level^2) **experience points** (so level two requires 100*(1^2) = 100, level 3 requires 100 * (2^2) = 400, etc). Experience points are acquired through completing major tasks at the discretion of the dungeon master, and through defeating enemies in **combat**. The greater the challenge the task or enemy poses, the greater the reward in experience points. There is no maximum to a character's level.

Leveling up:

- Once a character has leveled up, they earn 2 more points to invest in **base stats**, and can level up one **skill**. In most cases, leveling up a skill increases the effect of that skill in proportion to the skill's level (a skill which does damage might do an additional "x" damage per skill level

Exploration Mechanics

- Players move between zones determined by the **Dungeon Master**. Zone sizes are arbitrary, and should be separated by divides in what the players can see and what they can do. Rooms or large obstacles are good markers for zones. Zones themselves do not have and defined spatial characteristics, and mostly serve as mental "break points".
- In each zone, players can interact with the environment in any way a **Dungeon Master** will allow. The Dungeon Master is encouraged to allow any course of action the players can accurately explain and defend. If an action is possible but absurd, it is up to the Dungeon Master to decide whether or not to humor the request.
    - Most of these interactions will involve a **skill check**. To succeed at a skill check, the sum of whatever bonuses a player has to any relevant skills, and the result of rolling a 20 sided die, need to be equal or higher to whatever number the Dungeon Master feels would be appropriate. Seeing a coin on a table may require the sum be 5 or higher, while swimming through rapids may require a sum greater than 20
    - Examples of actions or skill checks a player may need or wish to take:
        - Swimming
        - Climbing
        - Lock-picking
        - Pickpocketing
        - Disarming Traps

        - Spotting Enemies
        - Listening
        - Searching for items/traps
        - Moving undetected
        - Reading another character's intentions
        - Understanding a cultural reference
    - By default, there is no penalty for failing a skill check, although a Dungeon Master is free to determine a consequence for a failed skill check (failing to disarm a trap sets it off, failing to pickpocket by a large amount causes the victim to notice, etc.).

Combat Mechanics

- Combat occurs whenever enemy character(s) notice the players and choose to fight (rather than run or talk) or vice versa, at which point the map becomes a grid of 4-sided **tiles**, fitted to the layout of the room. The players start combat by choosing one of the starting tiles that are within an area chosen by the **Dungeon Master**, while enemies start in tiles pre-selected by the Dungeon Master.
- Combat is turn based, with turn order being determined by **bonuses** to reflex-related **skill checks**. By default, this would be whoever has the highest **agility**, but the Dungeon Master may take into account skills that give bonuses to the character's reflexes, or a situational bonus from ambushing the enemy, as they see fit.
- Each turn, a player character can move and perform actions.
    - A character can move along a path of free tiles of length equal to or less than their **movement speed**. Characters cannot move through enemies, allies, or obstacles, unless a **skill** dictates otherwise.
    - There are three types of actions: instant actions, small actions, and large actions.
        - Instant actions include saying a single word or short phrase or making a gesture, very quickly reloading a ranged weapon (such as grabbing an arrow or magically adding new bullets to a gun), some **skills**, or any action which would normally be a small action but through a skill or temporary bonus is now faster (such as changing weapons if one has a quick-drawing skill). Instant actions can be performed as many times as one wants within a turn, but if a player attempts to do an absurd combination of instant actions, the

Dungeon Master may feel free to stop them (one cannot say a short phrase 10 times in one turn, this would likely be considered a small action instead).

- Small actions include drawing or sheathing a typical weapon under normal conditions, saying a sentence or two to another character, flipping a switch, throwing an object, and some **skills**. As a rule, a small action would either take a second or two of physical action, or take 5-8 seconds without requiring physical action or intense concentration. Small actions can be performed once a turn in addition to a large action or a second small action
- Large skills include attacking with a normal weapon under normal conditions, casting a spell, or any action which requires sustained movement for more than 5-8 seconds. Large actions can be performed once a turn

- Whenever a player attacks or uses certain damaging **spells**, the enemy character has a chance to block or dodge that attack or spell. In order for the attacking character to succeed, the sum of their bonuses towards hitting the opponent and the result of rolling a 20 sided die must be higher than the opponent's bonuses towards dodging. If the hit lands, the enemy takes **damage** based on the sum of any bonuses the attacking character has towards dealing damage and a dice roll, with the type and number of dice dependent on the **weapon** or skill used to attack. Some skills may reduce the amount of damage a character may take, but by default there are no bonuses towards damage reduction.
- If a character runs out of **health**, is rendered unconscious, or unable to move, their turns are skipped until they regain consciousness or control, or until they are revived. Once revived, their turns continue in the same order as normal. A character who has been killed or knocked out and is no longer standing upright doubles the amount of movement speed a character needs to pass through that tile.

- Combat ends when one side has no functioning members left (all members of that side have died, been rendered unconscious, are unable to move, have fled the battlefield, etc.). In the case of multiple sides the battle ends when all remaining sides are allies.
- When combat ends, the game returns to the **exploration** gameplay

Item Stats

- Each item has a weight (in pounds) and a monetary value (in bronze, silver, and gold coins with a 10000:100:1 ratio in value). Items may have other statistics in addition to these default ones and any unique to their type of item.

- Each item also has a classification: Weapon, Armor, Potion, Tools, and Other

Weapon Types:

- One handed weapons can be wielded in one or two hands (with two hands increasing any bonuses from **strength** when using the weapon by 1.5 times and decreasing any penalties from a low strength score when using the weapon by half). If wielded with one hand the other hand can hold another item
    - If the other item is another one handed *weapon*, as opposed to a shield or tool, the user has any bonuses from agility cut in half and any penalties from agility multiplied by 1.5, but may attack twice in the same action.
- Two handed weapons require both hands, and any penalties or bonuses associated with the weapon involving strength are doubled
- One handed projectiles can only be wielded with one hand
    - A second one handed projectile can be used in the off-hand, with twice as many projectiles fired per action, but reloading two one-handed weapons at the same time requires an action one size larger (an instant action becomes a small action, etc.), unless the reload requires a large action(s), in which case it doubles the number of large actions.
- Two handed projectiles require both hands

- Weapons have a damage output, two switching speeds (a draw speed, for drawing the weapon, and a storing speed, for putting it away). If it's a projectile weapon it will also have an ammo capacity and a reload speed, and if it's a melee weapon it has a classification of blunt or sharp. Some weapons may have other bonuses associated with them.

Armor Types:

- Helmets include any headwear, and only one can be worn at a time. Some helmets may incur penalties for skills that require clear sight or hearing.
- Gauntlets are any armor that cover the hands, including gloves. Two can be worn at a time, and they generally come in pairs. Gauntlets also have a damage output for unarmed strikes. Some gauntlets incur penalties for skills that require precise hand movements, such as lock-picking.
- Chest armor includes any armor that covers the upper body. One set can be worn at a time. Chest armor may incur penalties for stealth-based checks.

- Leg armor includes any armor that covers the lower body. One set can be worn at a time. Chest armor may incur penalties for stealth-based checks and checks involving acrobatic skills.
- Boots include any armor that covers the feet. One pair can be worn at a time. Boots may incur penalties for stealth-based checks and checks involving acrobatic skills.

Potions:

- Potions are stored liquids that cause effects when used on something. Potions have a status effect, methods of application, and possible targets of application.
    - Status effects are what the potions do. Examples include healing the target, poisoning the target, temporarily increasing a base stat of a target, temporarily adding a bonus to a skill of a target, changing something's color, dissolving something, or causing it to explode.
    - Methods of application are what conditions cause the potion to start working. Examples are drinking, touching, and inhaling.
    - Targets of application are what the potion can work on. Examples include people, metal, plants, and open air.

Tools:

- Tools are items that give bonuses for skill checks when they are used. Tools have bonus types and conditions of use. A character needs to establish that they are using the tool, as the tools do not give bonuses automatically
    - Bonus types are what bonuses the tool gives, and how large the bonus is. Bonuses can be for specific skills such as lock-picking or climbing, or can be general descriptions for when they would apply to non-generic checks (such as for breaking certain types of objects).
    - Conditions are requirements for the tools to be used. A tool might require two free hands (meaning that the user can't be holding a weapon at the same time), or may require that the area be quiet, or require access to a level surface.

Other:

- The "other" category includes any objects that do not directly fit into the other 4 categories. Household objects, food, supplies, and books are some of the many examples. Other objects have no universal stats besides cost and weight.

Skills

- Skill rules are determined on a case-by-case basis according to the skill description. Skills frequently violate normal gameplay rules, so if a skill contradicts a rule then assume the skill is correct. Likewise, if a skill specifically violates another skill, ignore the skill being violated.

Appendix B: User Preferences Survey

Player Preferences for a Tabletop Role Playing Game Application

This survey will be used to gather information on what kinds of preferences different kinds of players or potential players of Tabletop Role Playing Games (Tabletop RPGs) might have. The results of the survey will be used to aid in the design of an application that assists players in managing a game session of a tabletop RPG. All information gathered from this survey will be kept confidential and anonymous.

The following questions are about your experience with certain types of roleplaying games. If you are unfamiliar with what a question is talking about, answer no.

Are you familiar with the name Dungeons & Dragons, or the terms Tabletop Role-Playing Game (Tabletop RPG) or Live Action Role Playing (LARP) Yes No

Are you familiar with any of the common terminology (d20, attribute abbreviations like Str Int and Dex, +(x) skill bonus/penalties, instance, Dungeon Master, etc.) Yes No

Have you ever played Dungeons & Dragons or another Tabletop RPG, or participated in LARPing? Yes No

Have you been in a group that has participated in more than 3 sessions of playing one of these games? Yes No

Have you ever organized and/or ran 3 or more sessions with roughly the same group of people? Yes No

For each question, score how important the feature is on a scale of 1 to 7, with 1 being completely unimportant and 7 being very important

When playing an RPG for the first time, each player usually either creates a character, or is given a character. Each character has distinguishing statistics that affect how they are played; a character with high strength might be better at fighting their way through enemies head on, while a character with high intelligence might be able to solve a puzzle or lure enemies into traps. The following questions are about deciding on a character before the game starts:

Being able to create your own character	1	2	3	4	5	6	7
Knowing the specific or situational strengths and weaknesses of your character	1	2	3	4	5	6	7
Being able to finish character creation quickly	1	2	3	4	5	6	7
Having the option to choose a character that has already been made for you	1	2	3	4	5	6	7

Often times, Roleplaying games have someone running them as a “Dungeon Master”(DM), who keeps

track of enemies, obstacles, traps, and rewards for everyone else playing. For each of the questions, assume there is an app on a tablet that can help the DM keep track of this information and the various rules, but the DM is still expected to interact with both the players and the app:

Letting the DM make a judgment call to ignore the rules 1 2 3 4 5 6 7
Having the game keep track of what characters are allowed to do, and any calculations based on character statistics 1 2 3 4 5 6 7
Having a storyline or level already made for the DM, so they don't have to create their own from scratch 1 2 3 4 5 6 7
Having instructions/rules available so the DM can read them before the game starts 1 2 3 4 5 6 7
Having explanations of the rules available to the DM, even if the game already handles the calculations 1 2 3 4 5 6 7
Making sure that it's easy for a DM who doesn't want to read the rules to just start running the game 1 2 3 4 5 6 7

These questions are about using the aforementioned app to play without having a player act as the DM, and having the program tell the players relevant information instead
Being able to play a game *with friends* with the app doing everything the DM would 1 2 3 4 5 6 7
Being able to play a game *alone* with the app doing everything the DM would 1 2 3 4 5 6 7
Having graphics beyond a simple grid map and simple shapes for characters 1 2 3 4 5 6 7

Often times, a DM will choose a set of rules based on who is playing; if the DM is playing with beginners, they might use a simplified rule set, while experienced players may want to use the full set of rules to increase the challenge, or to allow for more complicated or situational rules. The players are usually informed of what set of rules the DM is planning to use, and may have them available in a small pamphlet or a large book, depending on how complex the rules are. The following questions assume the app to assist the DM is still being used, and that the instructions are available on the app. Having to stop mid-game to check the rules would prevent the app from being used, slowing down the game, although all rules could be checked before the game starts. The questions are about how the rules should be presented, and what kind of rules should be used.

Being able to use a full, complex set of rules, even if it makes the app slightly harder to use 1 2 3 4 5 6 7
Having the instructions include the full set of rules, even if it makes them slightly harder to read 1 2 3 4 5 6 7
Having simplified rules available, even if it changes how the game is played 1 2 3 4 5 6 7
Having the game do calculations based on the simplified rules, even if it can do the calculations based on the complex rules without extra input from the DM 1 2 3 4 5 6 7

The rest of the questions assume the player either has significant background knowledge of tabletop

RPGS, or has at least a small amount of actual experience playing them. If you do not have the background knowledge or experience, feel free to skip the rest of the survey

The following questions are about whether or not the app mentioned in previous sections should control movement in order to better calculate things like exact distance from one character to another, line of sight, area of effect, etc.

Having the app calculate movement with more precision than the DM indicates

1 2 3 4 5 6 7

Being able to override or ignore the precise movement referred to in the previous question

1 2 3 4 5 6 7

Having cover/line-of-sight calculated for the DM 1 2 3 4 5 6 7

Being able to ignore obstacles when calculating cover/line-of-sight 1 2 3 4 5 6 7

The following questions are for experienced users who have some experience or understanding of running a campaign or instance, and assume that the app has a built in level editor for creating an instance.

Being able to playtest an instance on your own to test how hard it is 1 2 3 4 5 6 7

Being able to playtest to check for problems with floor layouts (unintended obstacles for cover or accessing an object/area) 1 2 3 4 5 6 7

Being able to playtest to see how the rules handle specific skill interactions 1 2 3 4 5 6 7

Appendix C: Platonian Barbarian Draft

Note: This was intended to be a draft of a printed sheet a player would have on hand to act as a reference if they wished to play our game without the use of a tablet. While our game is no longer designed around the potential for pen-and-paper play, and several concepts presented in the draft were not included in the final game, we wanted to include this draft as a reference to our original plans.

*Convention: **x**d**y** is the result of rolling a **y** sided die **x** times (so 2d20 is the sum of two rolls of a 20 sided die)*

Human Platonian Barbarian

- Gear:

- Battle-axe
  - Blunt *(Apply bonuses from strength when trying to hit an enemy)*
  - 2-handed
  - Base Damage: 3d10 *(this weapon does damage equal to the sum of three rolls of a 10 sided die, before bonuses)*
  - Bonus: Heavy Blade: This weapon can be considered either sharp or blunt when making a skill check *(cutting a rope, breaking a rock)*
  - Attack speed: Large action
  - Draw speed: Small action
  - Store speed: Small action
  - Weight: 10 lb
  - Value: 5 gold pieces

- Long Bow
  - 2 handed
  - Projectile
  - 1 shot per reload, instant reload, 20 reloads with default storage
  - Range: 8 Spaces, 12 spaces if aiming *(no movement or any short actions performed that turn)*
  - Base Damage: 1d20 *(this weapon does damage equal to the sum of three rolls of a 10 sided die, before bonuses)*
  - Attack speed: Large action
  - Draw speed: Small action
  - Store speed: Small action
  - Weight: 1 lb
  - Value: 2 gold pieces

- Helmet:
    Leather Cap
    Armor Bonus: 1
    No penalties
    Weight: 1 lb
    Value: 1 silver piece
- Gauntlets:
    Steel Gauntlets x2
    Armor bonus: 2 each
    Damage: 2d6
    Penalty: -2 to checks involving precise hand movements (picking a lock) unless unequipped
    Weight: 2 lb each
    Value: 50 silver pieces each
- Upper Body:
    Chainmail
    Armor Bonus: 4
    Penalty: -1 to stealth checks
    Penalty: Takes 2 turns in combat to equip/unequip
    Weight: 5 lb
    Value: 2 gold pieces
- Lower Body:
    Chainmail Pants
    Armor Bonus: 4
    Penalty: -1 to stealth checks
    Weight: 4 lb
    Value: 1 gold piece
- Shoes:
    Leather Boots
    Armor Bonus: 2
    Penalty: None
    Weight: 1 lb
    Value: 20 silver pieces

- Class skills:

- Oration specialty:
    The Platonian barbarian has no formal education, and instead has devoted himself to understanding and expressing an idealized essence of perfection. As a result, the Platonian barbarian is illiterate, but compensates with strong verbal communication skills
    Bonus: +2 to verbal skills
    Penalty: Automatically fails skill checks based on reading

- Essence of Valor:

  The Platonian Barbarian inspires himself and nearby allies with a rousing speech on the nature of bravery and heroics

  Small Action:

  Costs 5 Mana a turn

  All allies within 10 spaces get +2 per skill level to both hit chance and damage, and +2 per skill level to skill checks involving willpower

- Essence of Good:

  The Platonian Barbarian shames nearby enemies with a lecture on the nature of good and evil

  Small Action:

  Costs 5 Mana a turn

  All enemies within 10 spaces get -1 per skill level to both their hit chance and damage, and -2 per skill level to skill checks involving willpower

  Only works if enemies are performing actions that they could view as unethical *(If they enemy is trying to rob and murder someone it would work, but if they are trying to defend their homes or territory it might not. Final decision is up to the Dungeon Master)*

- Essense of Power:

  The Platonian Barbarian channels within himself the concept of the ideal fighter, allowing him to ignore pain, move faster, and strike harder

  Small Action on first turn, lasts up to 5 turns, cannot be stopped until 3rd turn

  Costs 15 mana, + 5 for each turn beyond third

  Character ignores 4 damage a hit, deals an additional 4 damage a hit, has +2 to armor, and +3 to strength and agility checks

  -2 penalty to intelligence checks, cannot choose to perform intelligence-based skill checks (penalty applies if the Platonian barbarian is *forced* to make the skill check)

- Strike Form

  The Platonian Barbarian channels a desire to match an intangible perfection of form to allow himself to hit things better

  Large Action:

  Costs 10 Mana

  Regular attack, but with a (+2 +1 per skill level) bonus to hit chance *(at skill level 2, the bonus is 2+2 =a bonus of+4)*, and deals an additional (9 + skill level) damage if successful *(at skill level 2, the bonus damage is 9+2 =11 extra damage)*

Appendix D: Shadow Enemy Draft

Note: This was originally intended to be a reference sheet in the same style as the Platonian Barbarian character sheet.
Convention: xdy means roll a die with y sides x times, and take the resulting sum. 2d6 would mean roll a six sided die twice

Shadow Enemy

Health: 60
Mana: 40
Will: 40

Strength: 6
Dexterity: 6
Intelligence: 3
Willpower: 8
Spirit: 4

Attacks: Grappling: -2 to hit, 1d10 damage

Armor: base check for enemy hits is 10 (the sum of the attack bonus and the player's roll must be at least 10 in order to hit)

Skills:

Shadowmelt: Enemy can physically blend with shadows, temporarily reducing how large it is
Has +3 armor in dimly lit areas (caves, forests at night, etc.)