BATTLE OF BATONE
By Conglomerate Squad

Winner: Year 10-12 Open Platform

CONSTRUCTION/DESTRUCTION

Proudly supported by

CREATIVE VICTORIA

PAX AUS

IGEA
interactive games & entertainment association

ACER

BATTLE OF BATONE

GDD

Conglomerate Squad – 2023

Contents

Planning

Organisation

Responsibility

For the most part we will work collaboratively, sharing our progress. However, everyone will be given a role to oversee and deadlines to manage. Each role is outlined below.

Programmer

Matty will take the role of programmer as he has experience with software development from entering last year. He will create the scripts needed for the game. He will code all the major game elements, the server hosting being most complex and vital.

Game Designer

Michael will be given the role of game designer as he loves designing games and made the original physical card game. His understanding of how the game works is vital to making the digital version.

Artist/Visual Designer

Isla is a natural artist making her an excellent choice for the visual designer. She will create most of the art assets, including the resources, the fortresses, action cards, and more. Isla will teach herself digital art using programs such as *Procreate*. She has never used digital art programs before, but we are confident she will learn quickly.

Sound & Musical Effects

Sonny will take control of the music and sound effects. He will create medieval themed music, perfect for the game. He will also help make, design, and edit the sound effects used in the game. Sonny has previous experience in sound effects (SFX) creation; however, he has never attempted to compose music.

Testing

Throughout the entire process, individual team members will continually test and improve the game. We will also test the game with external players. This will provide valuable feedback which will enable us to improve the game.

Submission Guidelines

We will ensure our game is acceptable for submission, by referring to competition rules and judging rubric. We will regularly check for any themes that do not comply with the rules and remove them if necessary.

Workflow

We want our game to be multiplayer, and we have chosen to have a local server multiplayer over other forms of co-op. We thought about having bots or AI players. However, to code bots that are smart enough for the game to work and be enjoyable, it would be an enormous undertaking.

We also considered taking turns on the same device, but we received outside feedback that this wouldn't work well as a playability factor. The game has to be on a PC for the Australian STEM Video Game Challenge (VGC) and getting up and swapping chairs to have a turn would not be a great way to hold a player's interest.

We have chosen local multiplayer over online servers. The reason being if we make our game an online multiplayer, then we would have to host a server for players to use and this would cost money. Whereas, players can host the game using their computer as the server, and other players on the same Wi-Fi can connect and play. This also means the host won't have to worry about random unwanted players joining the server because players must be on the same Wi-Fi network to link up. The style of game we have chosen depends entirely on the local server element, without it, the game cannot function. This has to be made first because if it fails, we will have to make a whole new game.

Next, we can create the visual and audio assets for use in the game. During this time, Mathieson will continue to code the rest of the game. Unfortunately, the applications for the audio and visual design are on the same device. This slows us down as we have to take turns using the *iPad* to work. But if everything flows smoothly, we will have the complete prototype ready for outside testing by June to July.

After we modify the game based on the feedback, we will record a tutorial video. This will demonstrate to players how to play and what to do in the game. We will also make a website for people to download the game, watch tutorials, and view the rules.

For more evidence of our planning in early stages, refer to the minutes of our team meetings in *Appendix A*.

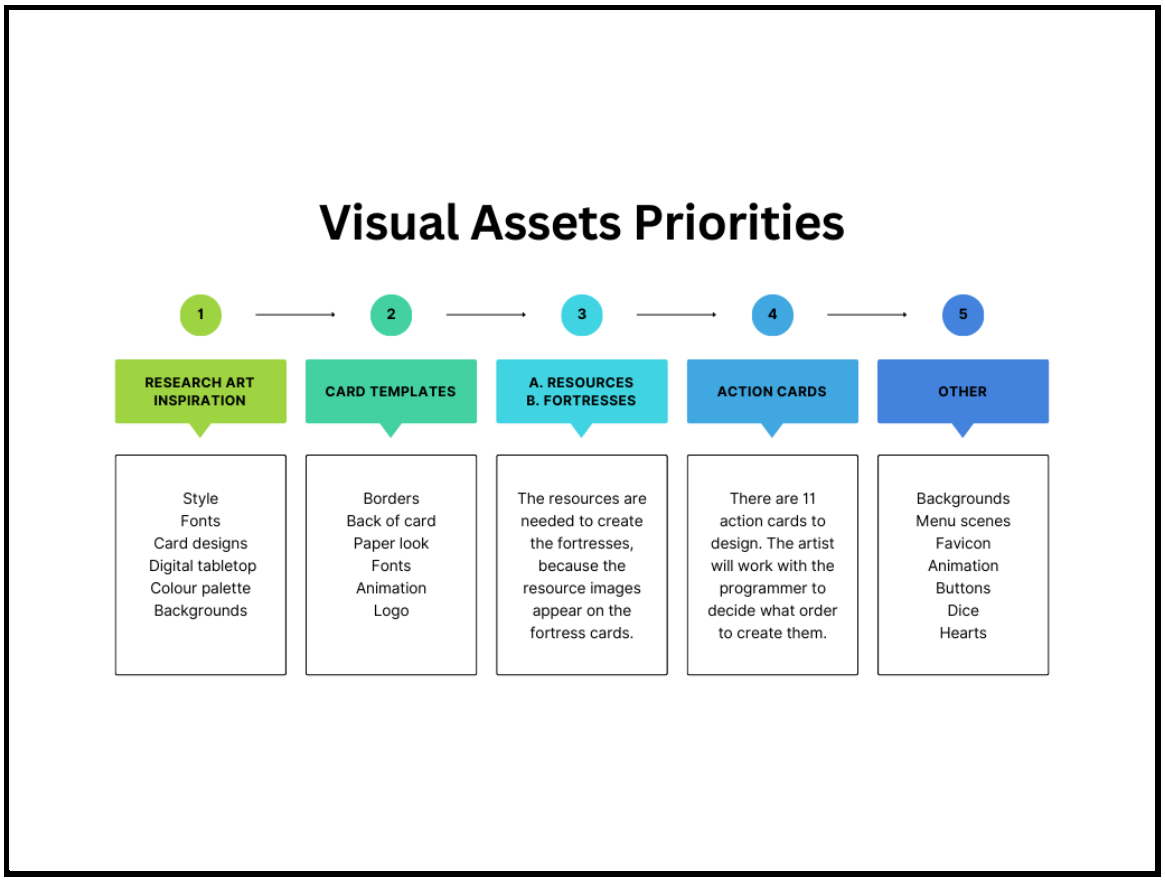


Figure 1: Diagram of the visual asset priorities

Timeline

We plan to have our full game ready for final testing by the 7th of July. We will allow for about a month of testing from outside testers, improving the game based on feedback. Once we are satisfied with the game, we will record our tutorial videos. An overview of our planned development process is outlined in the table below.

Game development process	Date to be finished
1. Test original card game to make decisions for conversion to digital card game.	28 February
2. Finalise server capabilities.	31 March
2. Program Prototype 1 and basic testing, art and SFX can be designed at the same time (animation can wait).	30 April
3. Add art images and SFX to the prototype.	30 April
4. Music inputted.	31 May
4. Prototype 2 and testing.	June-July
5. Final testing and debugging.	7 July
6. Tutorial Video recorded.	19 July
7. Upload game file to website or app platform, test.	31 July
8. Final GDD draft.	31 July
9. Game submission.	1 August

Table 1: This table shows what we prioritised first and the order of work.

Deadline

The competition submission window opens on the 24th of July and closes on the 7th of August 2023. Our aim is to finish the game by the 7th of July and the GDD before the 31st of July. These deadlines make allowances for any issues found in the final testing stage.

Inspiration and points of originality

Battle of Batone is both a physical and digital card game. There have been different inspirations for each. These inspirations are outlined below.

Card Games: Battle of Batone

Battle of Batone was originally a card game made 5 years ago by Michael with the help of his mother, Jillina, as a Christmas present for the family. While it is inspired by *Settlers of Catan*, Michael came up with the concept, rules, and art by himself. Battle of Batone contains a unique pack of homemade cards and a pair of dice.

Over the years the game received lots of positive feedback from friends and family alike. We wanted to re-enter the STEM VGC this year and we felt Battle of Batone fit the theme perfectly, so we have decided to convert it into a digital version. While we have to change some components from the card game to the digital version, the core elements will stay the same.

Board Games: *Settlers of Catan*

The main inspiration for Battle of Batone is the board game *Settlers of Catan.* Both *Settlers of Catan* and Battle of Batone both involve collecting, trading, building with resources as well as hindering or attacking other players. The aim of *Settlers of Catan* is to get a certain number of points by expanding your empire. However, the aim of Battle of Batone is to build your fortress up to destroy your opponents. This means *Settlers of Catan* focuses more on the expansion (settling) while Battle of Batone focuses more on attacking (battling). Whilst both games are tabletop games, *Settlers of Catan* is a board game containing a board, and game pieces in addition to cards and dice. On the other hand, the Battle of Batone is a card game with dice.

Video Games: *Solitaire, Settlers of Catan, Plants VS Zombies, Recore, Shadow of War,* and *Skyrim.*

Solitaire's gameplay, animation, sound, and special effects demonstrates to us how an online card game functions and engages its players.

Settlers of Catan is a board game that has an online version of it. We draw inspiration from how multiple players can play online together as well as the User Interface (UI) and User Experience (UX) it has as a digital board game e.g., buttons, page layouts, turn changing, sounds, and more.

Plants Vs Zombies is a game that influences our music, sound effect designs, and audio themes. The music throughout the arena and dark ages levels are the most influential for our game's music.

Recore, *Shadow of War*, and *Skyrim* will inspire our games loading screens and menus.

Game Inspiration for Battle of Batone		
Game	**Developers**	**Website**
Settlers of Catan	Klaus Teuber	www.catan.com/
Plants vs. Zombies 2	PopCap	www.ea.com/games/plants-vs-zombies/plants-vs-zombies-2?isLocalized=true
The Elder Scrolls V: Skyrim Special Edition	Bethesda Game Studios	elderscrolls.bethesda.net/en/skyrim
Middle earth: Shadow of War	Monolith Productions	www.shadowofwar.com/about/
ReCore: Definitive Edition	Comcept Inc., Armature Studio LLC	www.xbox.com/en-AU/play/games/recore-definitive-edition/9NBLGGH1Z6FQ
Microsoft Solitaire Collection	Xbox Game Studios	www.xbox.com/en-AU/games/store/microsoft-solitaire-collection/9wzdncrfhwd2

Table 2: Game inspiration

Originality

Battle of Batone has similar themes to settler games but has its own original gameplay and fantasy world. It incorporates various real and fantastical civilizations, each with their own fortresses, forming the mythical land of Batone in which the game is set. Gameplay is also highly unique with the game being broken into two stages as well as using both cards and dice.

There are many reasons why people would like to play Battle of Batone.

1. Battle of Batone is the best new and original digital card game. With the creativity of the new land that is Batone, the diverse civilizations infuse our game with imagination and excitement. Also, with our local multiplayer, friends and family can easily play together in friendly competitions.
2. Battle of Batone is a 3-6 player game catering for both small and large amounts of players.
3. Battle of Batone is suitable for most ages though it requires some skill in card games, as well as knowledge on how to operate a computer. However, with our genial style and general themes, the game's content will be appropriate and appealing to any who may play.
4. Battle of Batone has competition. Sabotage is a large element of the game creating lots of friendly player conflict.
5. Battle of Batone has a huge replayability value. Players can play the game over and over without losing interest. Role Playing Games (RPGs) and other story-based games offer little in the way of replayability, but Battle of Batone is fun to play multiple times.

Technical requirements

Game distribution

We had originally planned on trying to make a mobile game, but this was against the rules. So instead, we have decided to have it as a downloadable application. Players will download Battle of Batone directly from the website we will make for it.

Development Environment

We plan for our game to run on both *Windows* and *Mac* PC's. We will create the game using the *Godot* Engine, and then export one version for each. The laptops we are using have limited abilities and so *Godot* is one of our only options. However, *Godot* is great for us because we are familiar with it and our game will have local server hosting, which *Godot* has inbuilt functions and nodes for.

System requirements

The peripherals needed for Battle of Batone are a mouse, a keyboard, Wi-Fi, and for a better audio experience head/earphones or good quality speakers. The game will not start unless it has at least three players connected to the server including the host. It also has a maximum of six players and will not allow more than six to join the server.

Resourcing/Capability

Tools we will need might include the export requirements for *Windows* and *Mac*, along with the *Godot Engine* itself. Mathieson has been using our home network to test the game's server capabilities. He has been learning more about *Godot* and GDScript using the *Godot* Documentation, *GitHub*, *Reddit*, *Google, ChatGPT*, and *Bing AI*.

Technical requirements responsibilities

Our approach to taking responsibilities is collaborative; however, as we each have different skills, we allocated different technical responsibilities to the team member who thought they could best achieve the tasks. Our role allocation is explained further in the "Responsibility" section of the GDD on page ?.

Designing

Game Overview

Game title

Young Michael originally called the game Battle of Batone when he first made it five years ago. We liked the name, so we agreed to keep it for the digital version. We thought it fit our game, correctly portraying what our game is about, and its main objective. Batone is the name of the fantasy land in which the game is set. Battle indicates the main objective of the game; the different civilizations that make up Batone, fight for domination by destroying the others' fortresses. We also liked how it sounded, especially with the alliteration, the name flowed easily. Finally, it strongly relates to this year's theme of destruction.

Game description

Battle of Batone is a 2D digital tabletop card game. Our game is about a battle fought over the fictitious land of Batone. The game is played in two stages. During stage one, players will gather resources and construct their fortress. Then in stage two, all the players are pitted against each other in an epic free-for-all. The objective of our game is to be the last player standing. Destroy other players' fortresses to eliminate them from the game.

Engagement

Our game is addictive, engaging, and filled with strategy. Battle of Batone has infinite replayability, due to the game being multiplayer, the ability to play with different numbers of players, the randomly assigned roles, and more. The diversity of our action cards also add variety, so players can aid themselves and hinder opponents in different ways. The dice mechanism means that the final outcome is left to chance, which relieves tension, and ultimately allows anyone to win. This stage is fun and thrilling, making many who play it laugh.

It is very fun to play, with an entertaining backstory. Players are easily able to imagine themselves as their roles in the world of Batone.

Battle of Batone has an enticing level of difficulty for the target age group. It may take a few turns for new players to catch on, but they quickly will be wanting to increase their skills. If they don't win the first time, it only serves to drive them on, wanting to challenge each other again. However, while strategy is a large element, the smartest player is not guaranteed to win. The luck of the cards and dice means all have a chance to win.

For future engagement our team has considered developing this game further. We have thought about adding Central Processing Unit (CPU) controlled characters, adding detailed animation, and perhaps even inputting a few more SFX. We even considered producing and selling the physical Battle of Batone card game.

Audience

Battle of Batone is for a general audience, recommended for 8 years and above. Our game appeals to a wide audience, not being limited to age or ability. It is a great family game, great to play with friends, and a good game for larger groups.

It is not generally recommended for younger children as it requires a certain attention span. Because our game contains text, labels, and buttons the player must be able to read. The current format contains a game log, which often updates, and younger children may fall behind in what is happening.

The rules and gameplay are complex and may confuse young children. Those who play need to have a decent sense of strategy and how card games work, as well as how to operate computers and digital gaming. Also, setting up and hosting servers may be difficult for young children, so we recommend having an adult or older child to set up the game for each player.

Characters/Roles

Battle of Batone is a digital card game rather than an RPG, meaning it's focus is not on a main character and storyline. But this doesn't mean it lacks characters/roles. Our game contains six roles: Knight, Wizard, Cowboy, Ninja, Pirate, and Astronaut. At the start of the game each player is assigned one.

Being a card game, the roles have no story, no dialogue, or any other roleplay features. The roles help players keep track of which fortress is theirs and whose turn it is. Typically, these characters wouldn't be seen together. However, Batone, being a fictional world created by a young Michael, is made up of six fantasy genres that blend together to give it an old-world feel. To appeal to all, we have also purposefully taken away the facial and body features of the roles, leaving only their hats, helmets, and masks.

Genre	Fortress	Characters/Roles
Medieval	Castle	Knight
Western	Saloon	Cowboy
Space	Rocket	Astronaut
Pirates	Pirate ship	Pirate
Martial Arts	Dojo	Ninja
Magical	Observatory	Wizard

Table 3: This table shows the genre and the relevant fortress and character.

Environment

Our game takes place on a tabletop, as it is a digital board game with card game mechanics. That style of game we chose doesn't have a 2D or 3D shown world, but it does have a setting. The events of Battle of Batone take place in the fictitious land of Batone. Batone takes warriors and their relevant fortresses from different genres, some historically based, to create an old-world feeling land.

Level design

Battle of Batone's environment varies slightly between the two stages. Each stage has a different objective which the environment is centred around. Stage one is about players constructing their fortress, so it's more peaceful and ordered. Stage two is where players fight and so its atmosphere is more tense, even the music changes into a rousing battle song.

Art and graphics

Our game is a tabletop game and so we wanted our art to reflect that. When deciding on a background we looked at a flat green colour like other digital card games. But we decided that the green looked too much like a casino and is usually used for playing card computer games, like *Solitaire* and *Hearts*. We considered a wood grain, but when our cards were made, the colours too closely resembled the background. We then changed it to a grey texture. Our cards are also inspired by medieval vintage cards, giving the feel of an olden time tabletop game.

Player interaction

Our environment is highly interactive, as it is a multiplayer strategy card game. Players interact with each other through trade, actions that affect others, and the dice battle. Our game does not include Non-Player Characters (NPC) or CPU controlled characters. In a future iteration of the game, we would explore including these.

User Interface

Players can interact with most of the UI elements in our game. For example, we have included a game log to keep track of actions because it was out of our scope to include realistic animation that shows the player what is going on. We have also set the UI around the edges of the screen, so as not to cover the play area, making it feel more like a real card game.

Audio design

We have created audio that supports our tabletop video game. For example, we have sounds for button clicking, shuffling, card placing, dice rolls, and more. We have also matched our music with our medieval inspired theme. We have different songs that also match the stages and their objectives.

Theme

Our game is closely linked to this year's theme, construction/destruction. While brainstorming ideas for our game, the theme was very much in the front of our minds. We wanted to make sure our game addressed the theme in a clear and obvious way to show that it wasn't an afterthought. We decided to challenge ourselves and include both elements of construction and destruction.

We thought about a card game Michael had made years ago called Battle of Batone. It was about the collection of resources to build fortresses, which were then used for battle. It was perfect, as it included both elements.

Construction: building your fortress using resources.

Destruction: players fight and destroy opponents' fortresses.

The link between this year's theme and the stages of our game are further explained in the Objectives/Goals section under Gameplay/Mechanics.

Our link to the theme is more conceptual than literal. In our game, players are not physically constructing or destroying objects. However, the digital card game mechanics imply the fortresses are being built and broken. For example, when a resource card is played it puts the player a step closer to completion of their fortress.

For further examples of construction and destruction themed elements, refer to the Visual and Audio Design section.

Construction and Destruction vocabulary		
• Build • Construct • Fortify • Resources	• Completed • Defend • Fortress • Destroy	• Eliminate • Dominate • Damaged • Battle

Figure 2: Vocabulary.

Gameplay/mechanics

Objectives/Goals

The aim of Battle of Batone is to fortify your fortress and destroy everyone else's. At the start of the game, players are assigned a role and relevant fortress. The game is played in two stages. In stage one, players collect and play resource cards to construct their fortress. The game contains a wide variety of action cards, designed to aid themselves or thwart opponents. On each player's turn they can firstly play resources onto their fortress. If they can't or choose not to do so, they then draw two cards. Finally, they have the option to play an action card. Their turn ends after they do so. Players may also offer trades at any time during their turn and accept trades on others turns. Play continues clockwise. Stage one finishes when one player plays all the needed resources, completing their fortress.

At the start of stage two, the player's fortress strength is equal to the number of resources on the fortress at the end of stage one. The player who finished their fortress becomes the attacker. They select an opponent they want to battle, and both roll a die. Whoever has the higher score damages the other's fortress and becomes the attacker. If a tie occurs, both reroll. Play continues until only one fortress is remaining. The player who owns it wins. A win screen is activated, showing all players clearly who won. Players are returned to the lobby where they can either quit back to the main menu or play again.

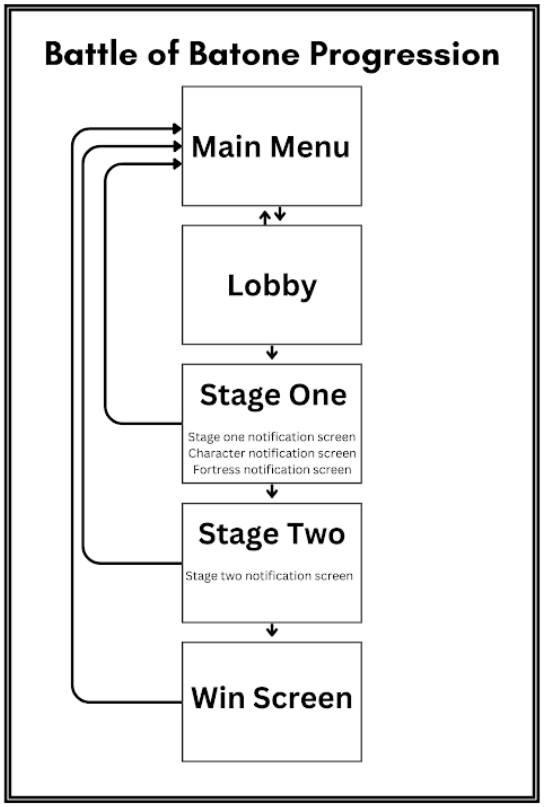


Figure 3: This chart shows gameplay and how it progresses.

Perspective

Our game is mostly 2D, with a top-down perspective of a tabletop card game. The one exception is stage two. The perspective remains the same, except for 3D dice that drop and roll. We know that some other digital board games have 3D dice and we thought it would be very cool to include in Battle of Batone.

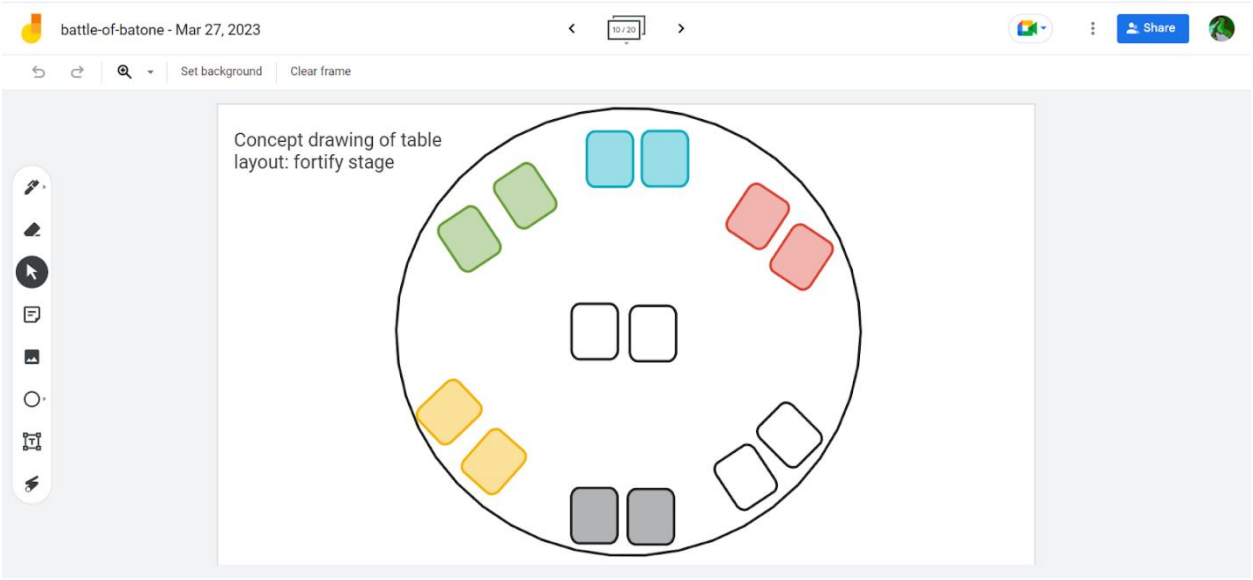


Figure 4: A concept drawing of the fortify stage.

Figure 5: The final product of the fortify stage.

Controls

Players will interact with our game physically using a mouse and keyboard. Most objects in our game will be clickable and there will be buttons and menus for the players to use. To interact with objects like cards in our game, a player will click on them, and options will appear dependent on the object or point in time of the game.

Instructions/Tutorials

While playing our game, players will have an option to open the rules page so they can read the rules at any time. They can also access the setup instructions from the main menu. We are planning to have a tutorial video available for players to watch if they are still unsure or learn in different ways.

Visual and Audio Design

Style

Our art style was heavily influenced by vintage medieval art and playing cards. We have used a medieval inspired art style which features lines to represent shading, a black line border, and a parchment texture card face. We believe it suits our game which includes some medieval inspired themes and images. We think this art style is very visually appealing and looks cool. It features See *Appendix B* for inspiration images.

Our art style represents the theme within a card game framework. We have lots of construction and destruction terminology, which are featured in the title, cards, and intermediate screens. During the game, players place resources onto their fortress and the images will colour in, representing they are building their fortress. In the next stage, the player's fortress card and the hearts on them, also change colour to visually show the player that their fortress is being destroyed.

We have chosen to adopt a medieval inspired style of music and audio to match our art theme. We will have a peaceful background song for the main screen, a village sort of music for the fortifying stage, and a rousing battle song for the battle stage.

We have closely linked our visuals to our audio. For example, our buttons have noises when clicked to let the player know it has done its action. Also, when cards are placed or flipped, then a noise is played to let players know what happened. And finally, our music fits with each stage, the ambience of the music matching the gameplay.

Our game's art and audio link to this year's theme. For example, when a player's fortress is damaged a destruction sound is played. Also, when a player's fortress is destroyed, a pop-up appears telling everyone that they were destroyed. The player's fortress card also goes dark, reinforcing the idea it was destroyed.

We didn't use any external assets. We wanted to build our game from scratch and have it all our own original content. We felt this would prove our competency and creativity.

Process

Music process

We chose Sonny as head of audio design. He had taken this role in last year's STEM VGC and had some experience in audio and editing. Sonny undertook an online learning course, watching tutorials to learn how to use our chosen music design platform/application, *GarageBand*. Sonny spent time creating our battle song using *Apple Loops*, and other various instruments. Mathieson helped make the music while he waited for Isla to draw images for the cards.

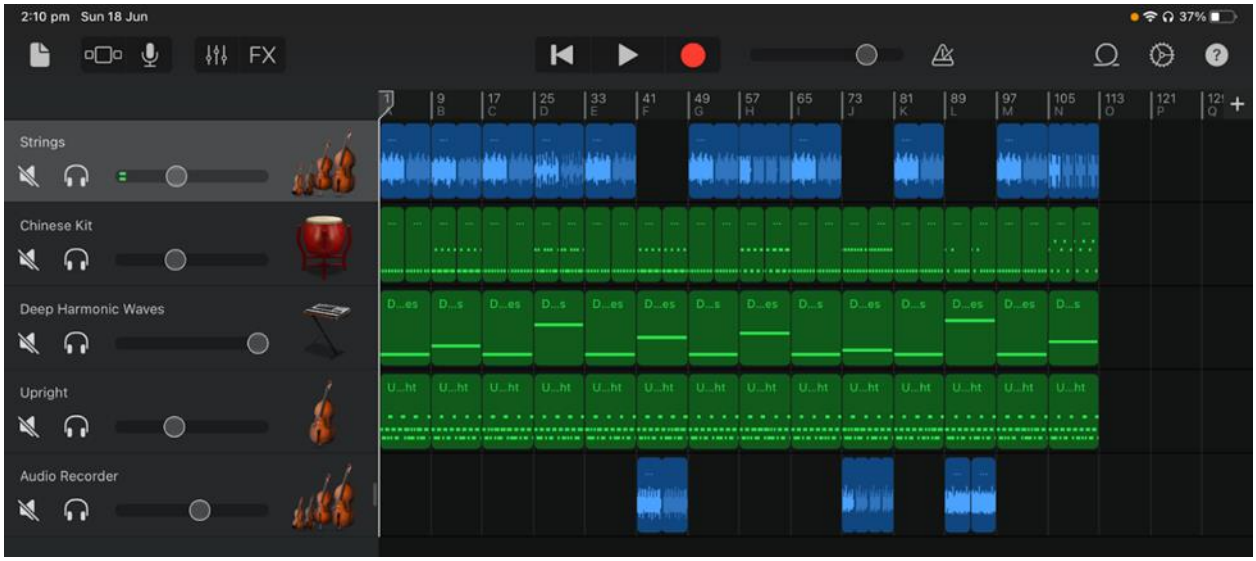


Figure 6: A screenshot of our battle song made using GarageBand for iPad.

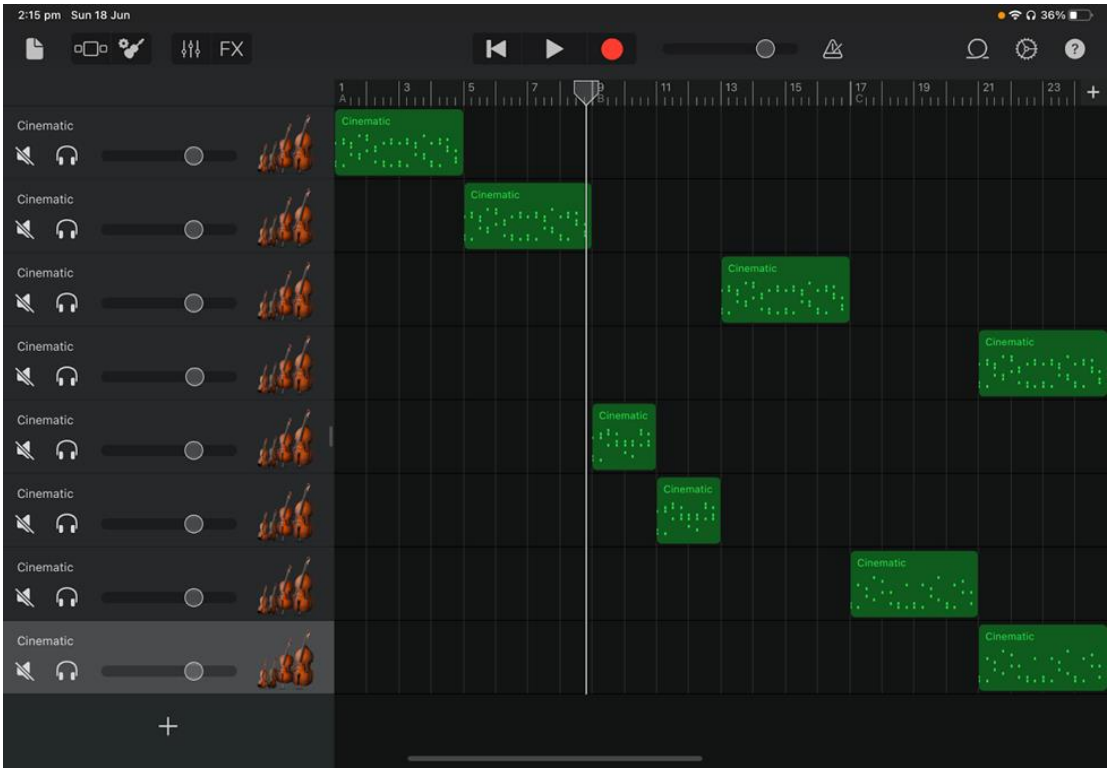


Figure 7: A screenshot of our menu song made in GarageBand.

Sound effects process

Initially, Sonny compiled a list of sound effects he thought our game would need. See *Appendix C* for full list of proposed SFX. But when we reviewed the list, we decided not to include many of the sounds as we thought too many sound effects would clog up our game audio. This made it easier for us, as we didn't have to record as many. Our team recorded most of the sound effects together, using a headset and the free audio programs *Audacity*, and *Sound Recorder*. Sonny then edited them in *Audacity* and *Microsoft Clipchamp*. He recorded and edited the rest by himself. The following table shows Battle of Batone's SFX and how we made them.

Sound	Creation
Card drawn, flipped, or placed	Made by drawing a single card off the deck.
Dice hit	Made by punching a couch and editing it in Audacity.
Shuffle	Made by riffling a deck of cards.
Win trumpet	Made by playing a brass ensemble on GarageBand.
Curse exhale	Made by recording Sonny's ghostly exhale and editing it in Audacity.
Destruction rumble	Made by dropping and clattering rocks together.

Table 4: Table of sounds used to create SFX.

Art process

Isla was given the leadership of the visual design, so she became our primary artist. We decided as a team against scanning the original images, as the drawings were made by Michael years ago, and were done in a child's style. We decided to adapt the art into a digital format. We purchased an *iPad* and Apple Pencil for drawing digital art. She used an app called *Procreate* to draw our images, adjusting them using feedback from teammates.

Isla based the images off the original Battle of Batone drawings because we liked the images and themes. However, we decided a few of the original drawings didn't fit our style, so we changed them. For example, we changed the General's Order image from a fired pistol to a medieval trumpet. We also changed the Thief from a western bandit to a hand stealing a diamond.

Figure 8: These pictures show the comparison of the original drawings to the new ones for the digital versions.

Card design process

When Isla started designing our cards, she created a template so that all our cards would be uniform. Also, to keep uniformity she selected one *Procreate* pen type and size. She chose the Ink Bleed pen to match vintage art styles. She then created the vintage paper texture for our card faces. She did this by using different pen strokes on *Procreate*. Lastly, she designed the back of the card. She made multiple drafts (see Appendix ?). The first draft looked too modern, the next drafts looked too much like casino playing cards, and so we decided to create the card back with the Battle of Batone logo. That logo was inspired by Michael's original logo including the sword as the T.

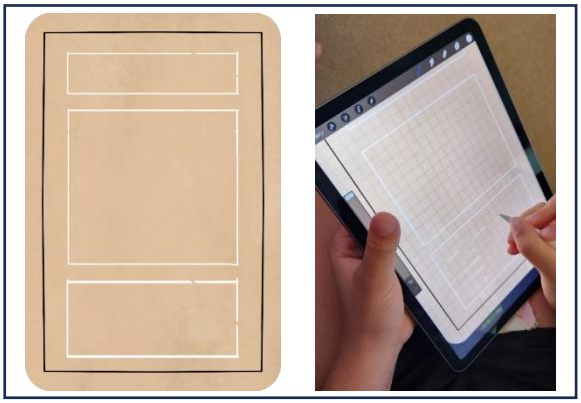


Figure 9: Card template design.

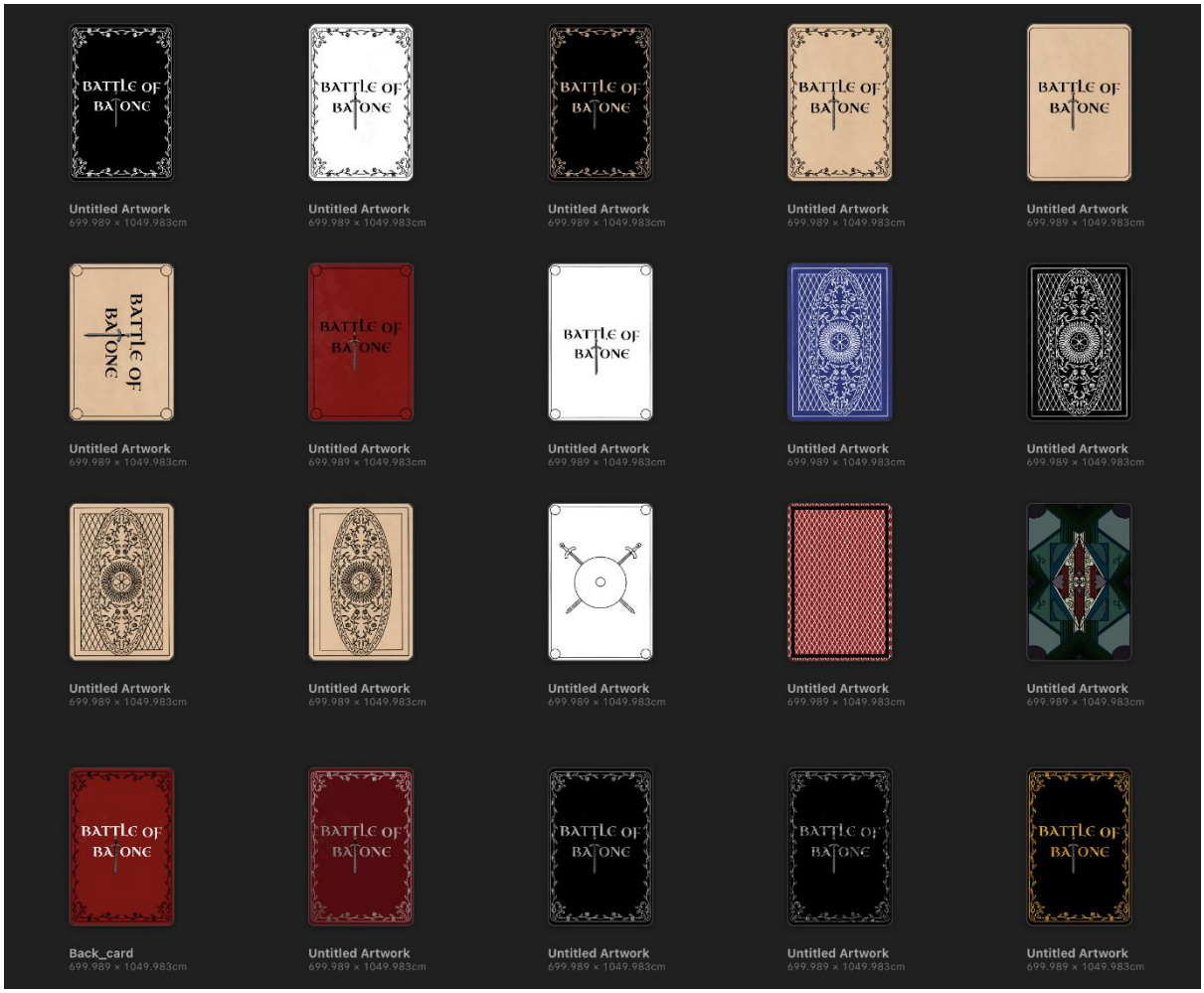


Figure 10: Card back drafts.

Resource, Fortress, and Action cards

Isla had to draw the resource images first as they were needed to then create the fortress pictures. The action cards came last because they affect gameplay, meaning the foundations of our code had to be in place before they were added. To see an art priorities list from later in the development process, see Appendix D.

Background – Stage 1 and Stage 2

Sonny created our background image for stage one and two using a special pen in *Procreate*. We didn't like some of the first ones he made, so he used different pens, colours, and textures until we found one we liked. *For further explanation, see Arts and graphics in the Environment section.*

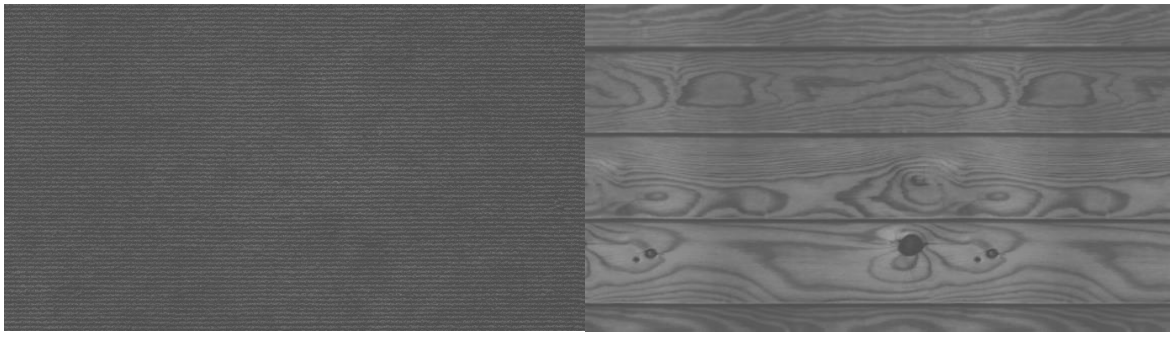


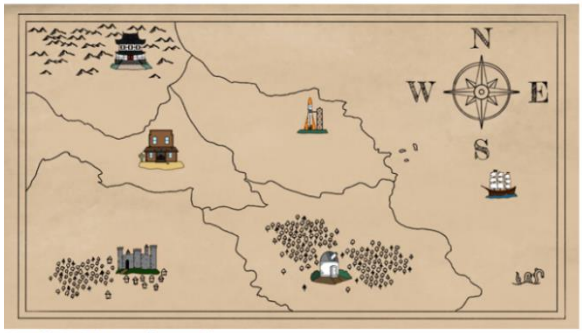
Figure 11: Current vs old table background.

Background – Main Menu and Lobby

We wanted to have a detailed landscape of Batone with some animation of moving trees, clouds, animals, and water for our main menu and lobby background. We decided this would take too long, so we settled upon the idea of a parchment map of Batone.

Draft 1: Too time consuming

Draft 2: Wrong dimensions



Draft 3: Not coloured

Final image: We used this in the game

Figure 12: Our main menu and lobby background drafts

Concepts

We used the drawings from the physical Battle of Batone as inspiration. We also drew some concept images of what we wanted the game to look like beforehand. We have included some examples below. For pictures of all the physical cards see *Appendix E.*

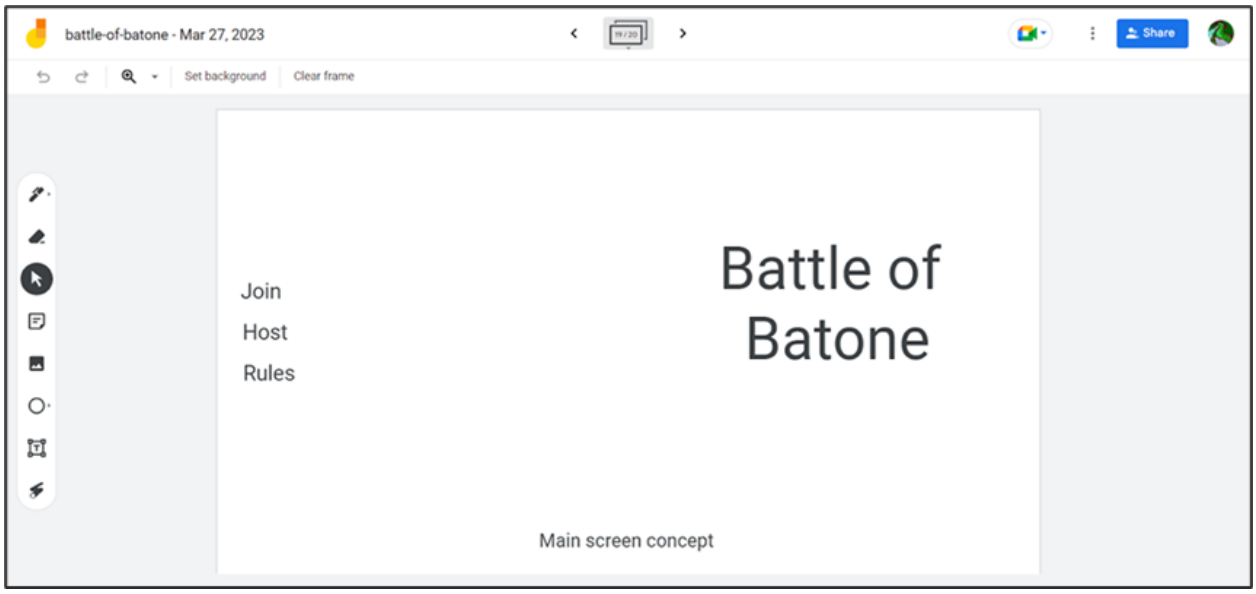


Figure 13: Concept drawing of the main menu screen.

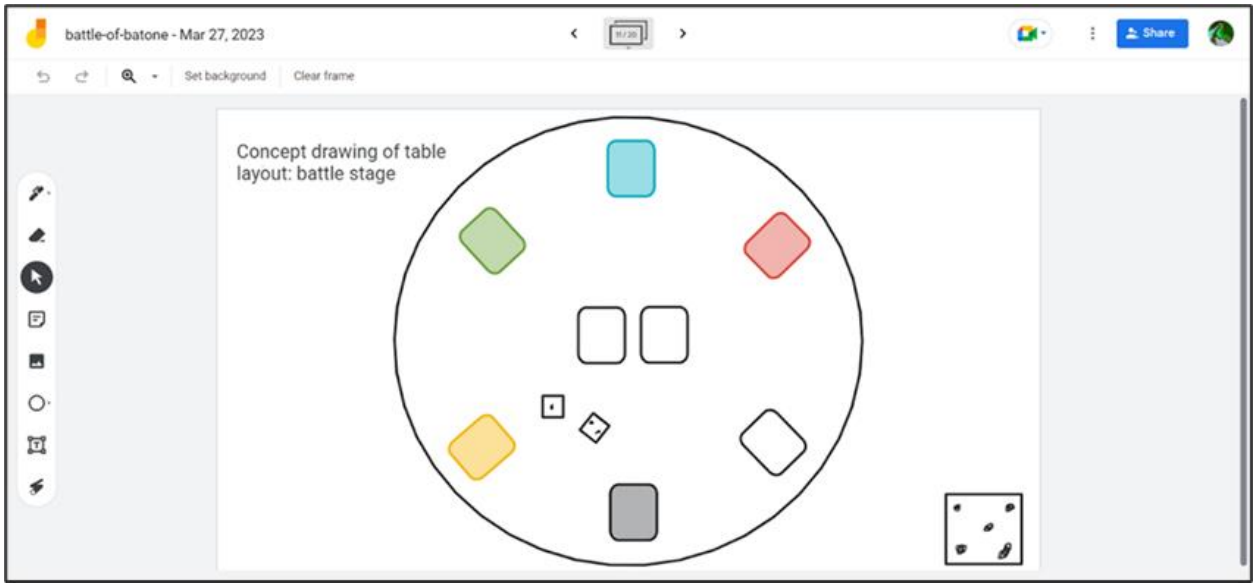


Figure 14: Concept drawing of the battle stage.

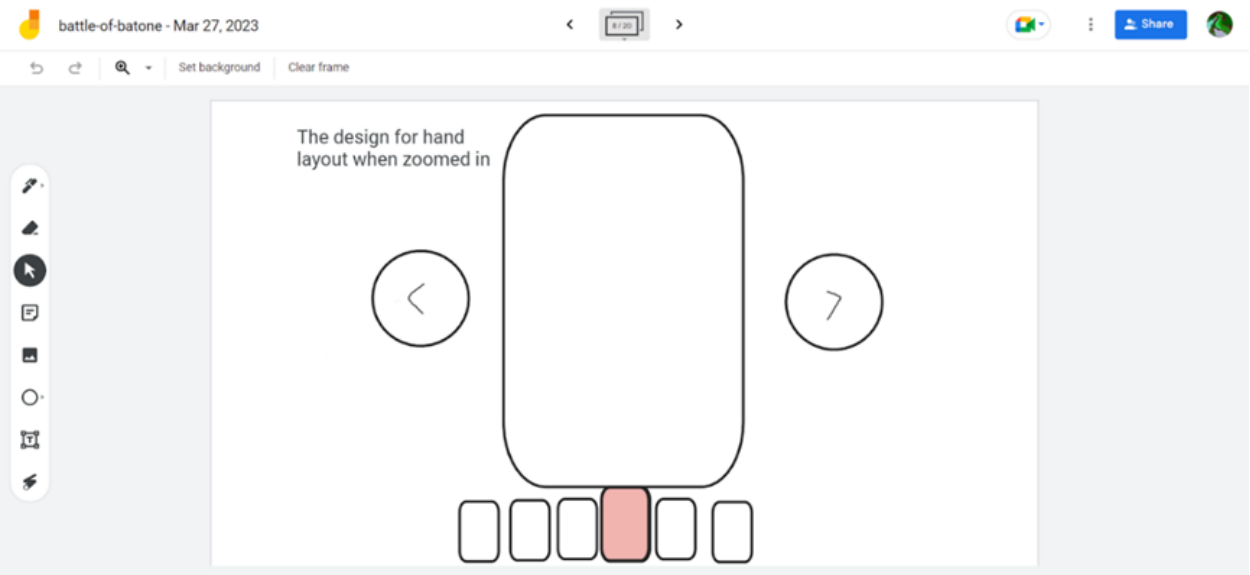


Figure 15: Concept drawing of the hand when clicked on.

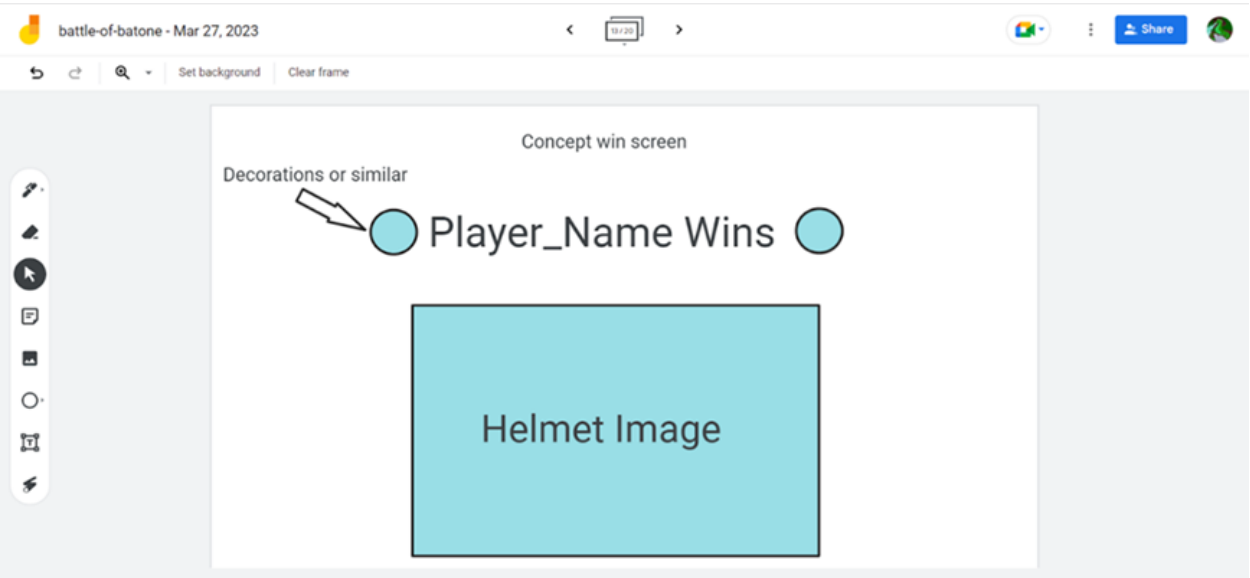


Figure 16: Concept drawing of the win screen.

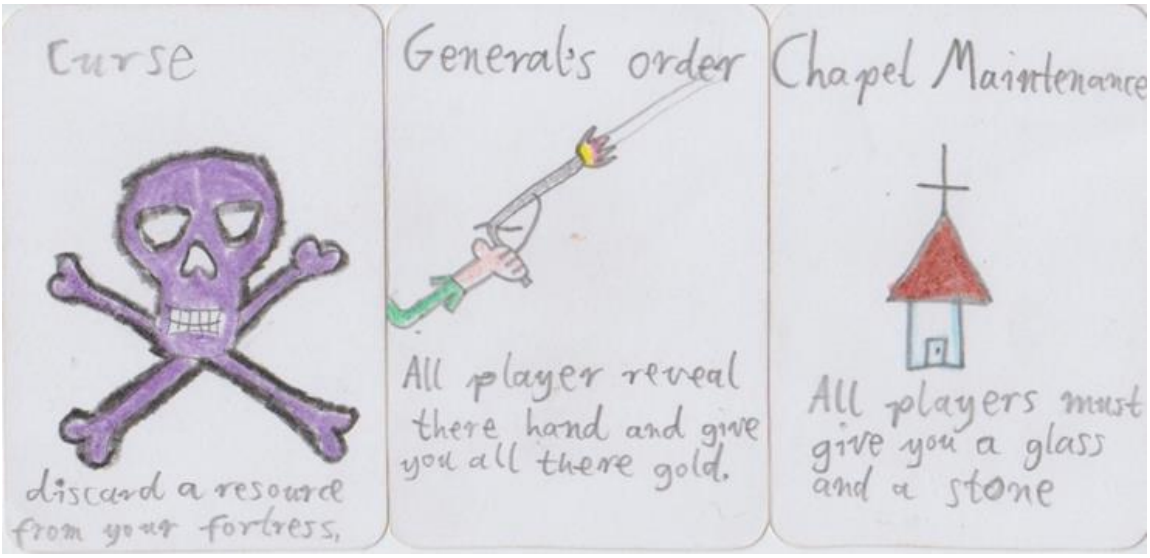


Figure 17: Some of the physical cards we used as inspiration.

Extra external inspirations

Later in the development process, we drew upon some extra sources for inspiration. For example, while writing the game rules we looked at how *Bang!* structured their rules. We also looked at many other games for inspiration on card layout and design. These included *Guillotine, Bang!, Lords of Waterdeep, Bohnanza, Catan, Exploding Kittens*, and *7 Wonders*.

Additional Game Inspiration for Battle of Batone		
Game	**Developers**	**Website**
Settlers of Catan	Klaus Teuber	www.catan.com/
Guillotine	Wizards of the Coast	
Bang!	Emiliano Sciarra	https://bang.dvgiochi.com/?lang=en
Bohnanza	Uwe Rosenberg	https://www.riograndegames.com/games/bohnanza/
Lords of Waterdeep	Wizards of the Coast	https://dnd.wizards.com/products/lords-waterdeep-board-game
7 Wonders	Antione Bauza	https://www.rprod.com/en/games/7-wonders
Exploding Kittens	The Oatmeal	https://www.explodingkittens.com/

Table 5: Additional game inspiration

Figure 18: An image of the cards we used as inspiration.

Animation

While designing, we thought it would be cool to animate some of our artworks. Animation is hard and time consuming, so we labelled it last priority. Isla and Sonny began experimenting with animation when they finished their main work. Isla animated the win screen and Sonny animated the background map for the main menu and lobby.

Fonts

Sonny took charge of finding fonts for our game. He looked through *Google Fonts* and came up with a list of fonts relevant to our game. The team went through the list together using the process of elimination to select two fonts, one for the headings and large titles, and one for the plain bodies of text. Our title font is *Uncial Antiqua*. We chose it because it is large and bold making good titles, paired with the sharp edges giving it the medieval look. Our basic text font is *Macondo*. It is clear to read when small and fits our medieval inspired style. Also, Isla used *Fredericka the Great* on our map as the letters on our compass.

This font was sourced from Google Fonts: *Uncial Antiqua* by *Astigmatic*

https://fonts.google.com/specimen/Uncial+Antiqua?query=unci used under the SIL Open Font Licence https://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&id=OFL

This font was sourced from Google Fonts: *Macondo* by *John Vargas Beltrán*

https://fonts.google.com/specimen/Macondo?query=Macon used under the SIL Open Font Licence https://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&id=OFL

This font was sourced from Google Fonts: *Fredericka the Great* by *Tart Workshop*

https://fonts.google.com/specimen/Fredericka+the+Great?query=fred used under the SIL Open Font Licence https://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&id=OFL

Located in google fonts:

- Bokor
- Metal Mania
- UnifrakturMaguntia
- UnifrakturCook
- Macondo
- Yatra One
- Fredericka the Great
- Averia Libre
- Pirata One
- Mirza
- Germania One
- Amarante
- Metamorphous
- Trade Winds
- Uncial Antiqua
- New Rocker
- Medieval Sharp
- Milonga

Title font options

- Milonga
- Uncial Antiqua✓
- Fredericka the Great

Description font options

- Averia Serif Libre
- Averia Libre
- Macondo✓

Figure 19: The list of fonts that our team picked from.

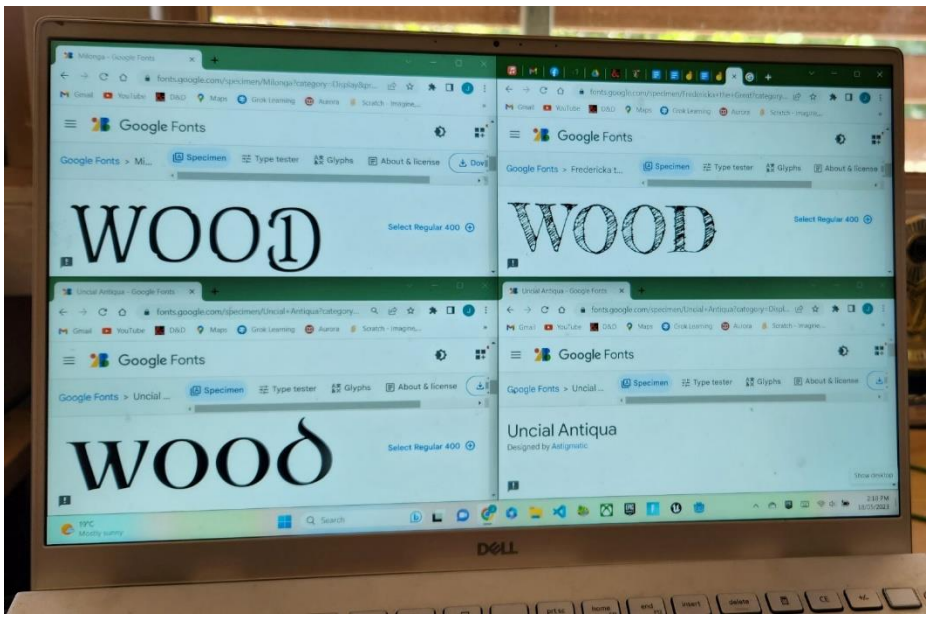


Figure 20: An image of our laptop while we were selecting fonts.

Game development tools

Tools used in the development process are listed below:

Tool	Use
Godot	Game development and scripting
Procreate	Art
iPad and Apple Pencil	Art
Canva	Visual and documentation
Adobe Color	Art
Audacity	Sound effects
Guitar	Music creation
Piano	Music creation
GarageBand	Music creation
Gmail	Collaboration and communication
Google Chat	Collaboration and communication
Google Docs	Collaboration and communication
Google Jamboard	Collaboration, communication, and planning
Google Drive	File sharing
Microsoft Word	GDD and documentation
Microsoft Excel	Planning and organisation
Google Fonts	Visual design
Wix	Website design

Figure 20: Game development tools

Reflecting

Testing, Fixing, and Project Execution

Testing

Over the years we have played Battle of Batone with friends and family and have always received praise and positive feedback for the game. While planning to make the digital version, we tested the original card game again with friends, looking for any issues and clarifying some rules.

Figure 21: Testing the physical version with friends to help us plan the digital version.

Gameplay and bug testing

Throughout the entire development process Mathieson bug tested the game so it would run smoothly. The type of video game we chose to develop was complex and took longer to complete than we expected. This left little time for outside testing of our game. Another challenge with testing was finding people with enough players and computers to test the game as it requires a minimum of three players.

We did a lot of in-house testing throughout the entire development process to ensure the game functioned as desired. We also had our dad test the game to get a view from someone outside of the development team. He enjoyed playing the game and gave us lots of constructive feedback. For example, when a player completed their fortress, it would jump straight to stage two with no explanation of what happened. Dad pointed this out and we developed a notification for the players to inform them of what happened.

We also had a friend help Mathieson test the game over *Google Meet* to identify compatibility and rendering issues. Using the information from the test, Mathieson learned that the game functions well on the *Linux* operating system (OS). He was also able to change the renderer to a lower powered one to allow older or weaker PCs to run the game with less lag. Unfortunately, *Apple Mac* computers were discovered to have many issues with game.

Fixing

Because we have been debugging the game throughout the development process, when it was testing time, the game had only a few minor issues and changes to be made. Below are some of the major problems our coder ran into during the programming.

Local Networked Multiplayer

This year we decided to create a multiplayer game, based on an old card game Michael had created years ago. We wanted to challenge ourselves this year and due to some early design choices and the competition guidelines, we landed on a local networked multiplayer game. Where players on the same network can play the game together on their own devices.

Mathieson, our programmer, began researching the possibilities of this type of game. He was able to create local networked multiplayer capabilities for the game, however, learning how to program this and debugging it took a large amount of time and he was not able to start programming the rest of the game until halfway through the competition.

An example of a major issue with this form of multiplayer was, once connected to a server if a player left the game, or the network was having issues and they were disconnected, none of the other peers on the server were notified of this disconnection. This meant the game continued as though that peer was still connected to the server, and the host peer would attempt to send signals to it that would crash the program, because it couldn't send information to an unknown peer. Mathieson fixed this issue by programming the server to close and the game to end if a peer disconnected during it, with a message to notify players that the server had been disconnected. We have also provided a note in the lobby to let players know that any quitting during the game will end it for all players.

```
peer.peer_disconnected.connect(
    func(peer_disconnected):
        rpc("remote_client_disconnected", peer_disconnected)
        client_disconnected(peer_disconnected)
        refuse_new_connections(peer_disconnected)
        if game_has_started == true:
            initiate_quit_sequence()
            var waiting = true
            while waiting != false:
                if get_tree().get_current_scene() == load("res://main_menu.tscn"):
                    rpc("rpc_quit_error_message")
                    quit_error_message()
                    waiting = false
                    return waiting
                else:
                    await get_tree().create_timer(0.1).timeout
                    waiting = true
                    return waiting
            return
)
```

Figure 22: Peer disconnected signal code used to solve the disconnection issue.

```
func _server_disconnected():
    OS.alert("Server disconnected.")
    multiplayer.server_disconnected.disconnect(_server_disconnected)
    get_tree().change_scene_to_file("res://main_menu.tscn")
    if peer != null:
        peer.close()
    variable_reset()
```

Figure 23: 'Server Disconnected' alert function.

Card Blanking

In Battle of Batone, if a player hasn't played any resources on their fortress, the resources will be displayed greyed out on the fortress card. If a player clicks on their fortress any resources not placed will be blanked over. Mathieson had a lot of issues trying to code this function. The main reason was the program couldn't just check whether the resource was in the list of resources played on the fortress or every resource of that type would be coloured when only a certain amount should have been. For example, the Dojo fortress requires two wood, two fabric, one steel, and two clay to be complete. At the beginning of the game, they would all have been blanked over. However, when the Ninja played a clay on their Dojo, both clays came up bright and colourful when only one clay was played. Mathieson's solution to this issue was to create arrays so that the program measured up these arrays and used them to blank only the correct amount of resources. For example, the function would check if an item was in an array or list and if that item was not already blanked out, if both were true it would blank that item.

```gdscript
func apply_blanking():
    if ServerManager.stage == 1:
        var blank_list = ServerManager.fortress_resources[ServerManager.get_my_fortress(ServerManager.ID)].duplicate()
        print(blank_list)
        var child_count = $CanvasLayer/FortressSubViewportContainer/SubViewport/ScrollContainer/HBoxContainer.get_child_count()
        for r in ServerManager.my_fortress_resources:
            blank_list.erase(r)
        for i in blank_list:
            print(i)
            print(blank_list)
            child_count = $CanvasLayer/FortressSubViewportContainer/SubViewport/ScrollContainer/HBoxContainer.get_child_count()
            var checker = false
            while checker == false:
                for c in $CanvasLayer/FortressSubViewportContainer/SubViewport/ScrollContainer/HBoxContainer.get_child_count():
                    var child = get_node("CanvasLayer/FortressSubViewportContainer/SubViewport/ScrollContainer/HBoxContainer/" + str(child_count - 1))
                    if child.texture == load("res://art/" + i + ".png"):
                        if child.modulate != Color("333333"):
                            child.modulate = Color("333333")
                            checker = true
                    child_count -= 1
    else:
        var count = 6
        var blanked_lives = 7 - ServerManager.my_fortress_resources.size()
        for life in blanked_lives:
            $CanvasLayer/FortressSubViewportContainer/SubViewport/ScrollContainer/HBoxContainer.get_child(count).modulate = Color("333333")
            count -= 1
```

Figure 24: 'Apply Blanking' function code to grey-out resources not yet played on the fortresses.

The Dice

Mathieson thought it would be a nice addition to have 3D dice roll on the screen when the "Roll" button was pressed in stage two. However, this ended up creating a lot of bugs and work for him. He created dice and made them roll on screen, but then had to use those dice to get the number the players would be viewing. and determine the outcome of that part of the game. The first issue was that *Godot* has no built-in function to tell if a 3D object is moving or not, so it was difficult to tell it when to check which side was facing up. He used *Godot's* **_process** function to determine whether the dices' position was the same as it was last **process_frame**, if it was, it meant the dice were still. However, using the process function created some issues with the code because it was run every **process_frame** so Mathieson had to make sure he had a boolean set up to check if it was already motionless or not. He later realised he should have been using the **_physics_process** function because the 3D dice are run with *Godot's* in-built physics, the code was changed to run with **_physics_process** instead.

The next issue was figuring out which side the player perceived as up and translating that to the code. At first Mathieson attempted to use the rotation of the faces on the 3D dice to determine which side was facing "up". The program was being run every frame so the number would occasionally get mixed up and the wrong number would be put into the game's calculations and the outcome of the battle would then be wrong. So, Mathieson disabled the rotation checker and added **raycasts** to the faces of the dice. He also created a "roof" object that only the **raycast** could interact with. If a **raycast** was hitting the "roof" then that **raycast's** parent face was the one facing up.

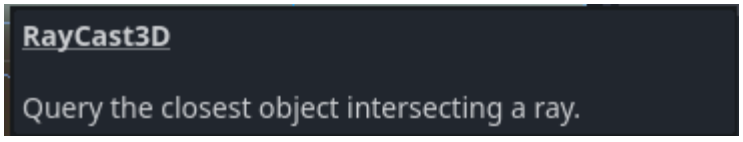


Figure 25: RayCast3D class description.

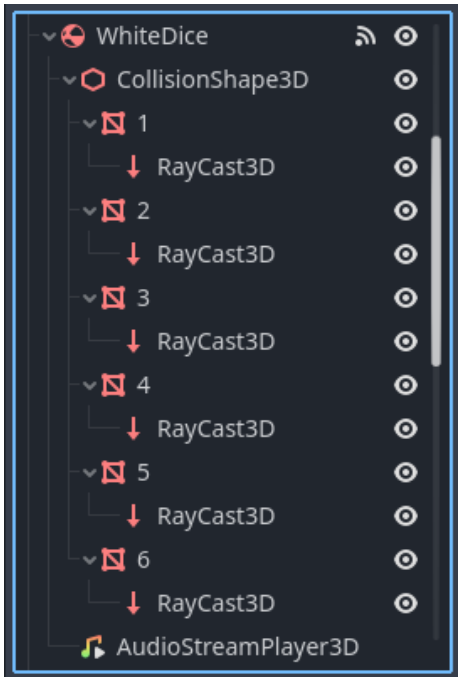


Figure 26: 3D dice node tree.

The final issue was that occasionally the dice would not land flat and the raycasts were not pointing straight up into the "roof". Sometimes the dice would get stuck on the walls, other dice, or land on an angle by themselves. Once again, there were no built-in functions to solve this, so to prevent the dice from getting stuck Mathieson programmed them to bounce off the walls by giving them extra velocity whenever they touched one.

```
46 func _on_white_dice_body_entered(body):
47     print(body)
48     @warning_ignore("integer_division")
49     var random_number = randi_range(1, 6) / 2
50     $WhiteDice/AudioStreamPlayer3D.play(random_number)
51     print($WhiteDice/AudioStreamPlayer3D.stream)
52     print($WhiteDice/AudioStreamPlayer3D.playing)
53     if body.get_collision_layer() == 4:
54         white_collision = true
55         print(body)
56         var white_position = $WhiteDice.position
57         await get_tree().physics_frame
58         if white_position == $WhiteDice.position:
59             if body == $Wall1:
60                 $WhiteDice.linear_velocity.z = -10
61                 print("b bounce")
62             if body == $Wall2:
63                 $WhiteDice.linear_velocity.z = 10
64                 print("b bounce")
65             if body == $Wall3:
66                 $WhiteDice.linear_velocity.x = 10
67                 print("b bounce")
68             if body == $Wall4:
69                 $WhiteDice.linear_velocity.x = -10
70                 print("b bounce")
```

Figure 27: Wall bounce script for dice.

Even though the dice can still potentially get stuck on each other, or land awkwardly by themselves, we decided it was improbable enough to be ignored. Our reasoning for this, each of these only ever occurred once and *Godot's* physics should stop these events from occurring most of the time.

Action Cards

During the early stages of the game Mathieson programmed all the action cards, however, he was unable to test them due to the artworks not being finished. In Battle of Batone's programming the cards are texture_buttons. The button would have the texture of the action it represents and when pressed would call the corresponding function. Not testing these functions while they were being programmed, brought up a lot of issues when the artworks were finished and implemented. Mathieson was unsure why the code wasn't working because the game would not crash, and no errors were printed in the debugger or output. After searching for the root of the issues, with unsuccessful results, he decided it would be a better use of time to reprogram the actions entirely. This was frustrating for Mathieson; however, he believes the actions were reprogrammed better and more efficiently the second time. See *Appendix G* for a comparison of old and new Chapel Maintenance action card scripts.

Mac OS

In the planning stage of our game, we decided that our game would run on both *Mac* and *Windows*. The game was coded on *Windows* and when tested on a *Mac* we found there were some major issues with it, e.g., menu bars don't exist on *Mac*. We decided at that point it wasn't worth trying to solve these issues and we are now only going to have the option of *Windows*. We also had trouble exporting the game for *Mac* too. This is something we would investigate if we had more time. After the competition we may continue to solve these errors and create a *Mac* and *Linux* friendly program, even researching the possibilities of mobile versions.

Art and visual design

We were unable to purchase our *iPad* for our art until April, so we were late in creating our assets. This set us back behind schedule. Also, *Procreate* was completely new for Isla. It was the first time she had ever used digital art, so she didn't know anything about it. She had to watch *YouTube* tutorials to learn how it worked. After Isla had learnt the main tools, she started designing and drawing the cards.

Isla wanted good detail on the images, so she made the canvas very big. We later realised this made the files sizes too big which created challenges importing/exporting and load times. We were forced to compress them, otherwise the game would be too large, and no one would want to download it. Mathieson also had to adjust the image dimensions.

One problem that stemmed from this, is that when we compressed our images, the colours changed. We didn't know what was causing this problem. Luckily it only severely affected one card, the Tax collector. It changed our gems to brown, so we had to change the colours on *Procreate*. Our blue and red had to be brighter, but the red didn't work. We had to try multiple times until we finally got it right. Once Isla had compressed the updated images, she downloaded the compressed versions and sent them to Mathieson.

Another problem was our animation. If we exported animations from *Procreate* they either lost their quality or were not functional. We had to send the animations frame by frame to Mathieson and he constructed them within *Godot* which took time.

Project Execution

Version control, file sharing, and backups

We decided to use *Google Drive* as our file sharing platform. We have used *Google Drive* before, so it was easy for all members of the team to use. It also acted like a backup location, storing copies of all our work.

Our game is complete for the competition, but if we had unlimited time and resources, we would improve a few things. We would include animations of the cards being played and shuffled to show players what is occurring visually. There was also a minor glitch that occurred occasionally, where at the beginning of the game a player might receive an empty trade that didn't really exist. Ideally, we would like to fix this, but we made the decision not to because it occurred so rarely, it didn't affect the gameplay or functionality, and we believed the time could be better spent on other areas.

One of the largest usages of time was learning about and programming the local networked multiplayer capabilities. We believe that it has really paid off, because the game is multiplayer and can be played across devices. A similar success was the game log. It took a long time to correct and program each individual log, however it allows the players to know what is occurring in the game which is key to its playability.

We are confident in our game development, design, and programming skills. Next time we would work on our time and project management. Early in the planning process Mathieson created a document of naming conventions for the *Google Drive*. See *Appendix F* for naming conventions image. However, for some of the younger, less experienced members of the team it was difficult to adhere to these naming conventions and this resulted in some difficulty locating files in the drive.

Limitations

While we were deciding on game mechanics, we originally thought about making basic CPUs as computer players. People with no one to join them, could then play the game themselves against the bots. But we decided this was too hard, as we would have to account for every single scenario and give the appropriate response. We selected the other option to have multiple real people playing the game against each other. Real players are smarter than any basic CPUs we could make, meaning they play more realistically which is in our favour. Having real players only also adds the social aspect of the game. To do this we had to set up the local server mechanisms.

Sometimes our team had miscommunication. At times, team members forgot to look at the task list, or did not set deadlines for task completion. This meant waiting for tasks to be done or taking longer to complete than should have.

We stopped scheduling and recording our meetings towards the end of the project. Not properly scheduling meetings wasn't too much of an issue because we would often talk and share thoughts about the game as well as it's progression to each other. However not recording them or taking notes about our conversations occasionally led to information or ideas being forgotten.

We have learnt from our challenges and in the future, we will set more specific tasks and deadlines, keep our documents tidier and more consistent, hold more organised meetings, be more accountable in completing tasks, as well as recording any ideas or information.

Programming

Ideally, we would have more team members spending time coding or learning code. Mathieson was the only team member doing any coding on the game. This meant that when it came to a problem in

the code, nobody else was able to give him much assistance. It also meant that he spent most of his time and focus on coding the game, working very hard to get it to where it is now.

An example of where extra coders would have been ideal was when Mathieson was creating the server. It was extremely difficult to learn and create. Because he had to construct it by himself, it took him roughly halfway through the submission timeline before he moved on to programming other aspects of the game.

While Mathieson had to do code the entire game, we still managed to develop a quality submission for this year's Australian Stem videogame competition. We all worked on parts of the submission that we were good at or interested in. This worked well because we could develop our skills in these areas or develop each part of our submission done efficiently with quality results.

In the future we would like to have all the team members sharing roles more. This is so we can learn from each other, share ideas, and have more variety in our tasks.

Self-development

In future projects we would like to have more realistic expectations of ourselves and each other. We would sometimes want to do things our own way, not want to admit we were wrong, or listen to other members ideas. We wasted time on unnecessary aspects of the game trying to make it perfect. We also sometimes got frustrated and stressed when things weren't up to the standard we desired. However, because of these expectations we pushed extra hard to create the absolute best submission we could.

Original assets

We could have incorporated third party SFX and music or free images in our game, but we really wanted to show our skills and originality by creating as many of the game assets ourselves. We only used free-to-use fonts for our game, its rules, and the GDD.

In the original game, each role not only had a unique fortress but also a unique weapon as well which was on the other side of the fortress card making it double sided. In stage one, you would have the fortress face up but when stage 2 began you would flip it over to reveal your weapon. We intentionally decide to omit the weapon for the digital version. While we felt having the weapons would be a great way to convey destruction, we thought that it took away from the construction and destruction of the fortress itself and put less emphasis it because players spend time building in stage 1.

Website hosting

We made a website for Battle of Batone using *Wix*. We chose *Wix* because it allowed us to quickly create a professional looking website for free. On the website is the download link. We decided to have Battle of Batone as a digital game rather than a browser game. We chose this option mostly for safety reasons. Because the game runs on the players' own servers, we thought it would eliminate the risk of players allowing web access to their internet. The other reason is we thought it would decrease game lag.

Appendices

Appendix A: Minutes of team meetings

8 February 2023

Australian STEM Video Game Challenge team meeting

Agenda Item/Topic	Discussion/Outcomes	Action By	Due Date
1. GDD Planning section	What platform will you use to build your game? *Godot* 2D because we are familiar with coding on this platform already; Unreal Engine is powerful, good software, renders well. We decided to keep everything 2D but consider some simple 2D animation like bubbling potions or cannon firing. **Motion**: That we use *Godot* Engine in 2D with simple 2D animation. **Motion**: Sonny to draft a GDD template for the Planning section	Sonny	13 Feb 2023
2. Ideas brainstorm	• Battle of Batone – card game • DNA • Beavers dam – build, humans, weather and creatures destroy • Robot Wars • Speed Run – like temple run, build platforms, and break barriers • Survivor type game • Potions • Ants in a pantry build to get to food, destroy food and pantry • Coding, decoding • Farm theme – growing crops and pests destroying crops • Tycoon game (Mike refuses to build such a game). Motion: That we try Battle of Batone card game		
3. Theme	Confirmed the theme: Construction and Destruction. Rules state that you can cover either or both topics		

4. Computer software and hardware capabilities	Check computer capabilities on laptops and Mum's computer. There is some concern that the computers are slow or not good enough. Might be ok just for *Godot*. Motion: Do a check on upgrades etc	Mathieson	
5. Gameplay – how to show lives in battle phase	• keep cards as lives • flip cards to show hearts (some felt this didn't keep to a traditional card game) • trade cards in for lives Motion: To be decided after card game testing		
6. Gameplay – trading resources	There was discussion about trading resources and how that would work in the video game compared to the card game. Motion: To be decided after card game testing		
7. *Google Drive* and file naming	The group decided there is no need to learn GitHub as we are all in the same location and files are easy to share. Mathieson proposed file naming system for all to follow e.g., laser_sound_0.1 Motion: Mathieson to create a new *Google Drive* with folders and a reference sheet doc for naming files	Mathieson	13 Feb 2023
8. Artwork	Discussion about the backs of the cards and what they might look like. We talked about whether we needed new artwork or refine the ones we have. Most agreed what we have was ok. But not set in stone. Motion: Isla to practise drawing of the cards and look at digital solutions for creating artwork	Isla	
9. Snacks	Motion: Michael to organise snacks for the next meeting	Michael	13 Feb 2023
10. Next meeting	13 February 2023		

13 February 2023

Australian STEM Video Game Challenge team meeting

Agenda Item/Topic	Discussion/Outcomes	Action By	Due Date
1. Game chosen	Spelling - Batone To decide about number of players and AI Motion: build a digital version of Battle of Batone card game.		

2. Trading	Need to decide about trading phases and rules		
3. Game Play	• number of resources in deck based on players and fortress • AI or take turns - Yes, No or both, online playability • trading • players - capped at 6 players • more actions - consensus was no more actions • test to see if wood, clay, fabric is harder to get		
4. GDD Planning section	Organisation Motion - Sonny to write this section	Sonny	
Roles	• Game Designer - Michael • Art/Visual Designer - Isla • Programmer - Mathieson • Storyteller - NA • Sound and Music Effects - Sonny • Tester - everyone plus outside testers		
Submission guidelines	Use Satellite Jump as a guide for wording for GDD about submission guidelines		
Workflow	In what order will you develop the components of your game? Step 1 - test card game to make decisions for conversion to digital card game. Step 2 - program Prototype 1 and basic testing, art and FX can be designed at the same time (animation can wait) Step 3 - add art images and FX to prototype Step 4 - Prototype 2 and testing Step 5 - Final testing and debugging Step 6 - Upload game file to website or app platform, test Step 7 - Final GDD draft Step 8 - Game submission		
5. GDD Planning section	Inspiration and points of originality • Provide references Motion: group to research and make comment Motion: Assigned to Michael to draft for GDD because he designed the card game	Everyone Michael	
6. GDD Planning section	Technical requirements Motion: Mathieson to fill out as he is the programmer and has most knowledge	Mathieson	
7. Key Dates	Submission window: Mon 24 July - Mon 7 August Motion: plan to have testing done by 7th July to get the game in on time.		

8. Testing for 6 players	Plan a time with the M******* family to test 6 players. In the meantime test for 4 players Motion: Sonny to set up a date with the M******* family	Sonny	
9. Next meeting	20 February 2023		

20 February 2023

Australian STEM Video Game Challenge team meeting

Agenda Item/Topic	Discussion/Outcomes	Action By	Due Date
1. Number of players	We will have 3-6 players.		
2. Resource Availability	The game will feature only the amount of resources required by the fortress cards selected.		
3. Art	Different art styles and platforms will be explored and discussed. A new *Google Drive* folder will be created for people to add ideas of art styles into. We are still waiting to get a tablet for digital art production.	Sonny and Isla	
4. Game platform and online play.	Local online play and mobile game viability will be investigated and further decisions will be made.	Matty	
5. Curse Card	Still needs to be discussed		
6. AI	Whether to have computer players in the game or not. We have decided against it for now.		

Appendix B: Inspiration images and references

References

Journal, W. S., Atlantic, T., Magazine, W., Times, T. N. Y., Post, T. W., & Times, F. (n.d.). Welcome to the World of Catan. CATAN. https://www.catan.com/

PopCap. (n.d.). Plants vs. Zombies 2. https://www.ea.com/games/plants-vs-zombies/plants-vs-zombies-2?isLocalized=true

Bethesda Game Studios. (n.d.-b). The elder scrolls: Skyrim. Elder Scrolls. https://elderscrolls.bethesda.net/en/skyrim

Monolith Productions. (n.d.-a). Nothing will be forgotten. Shadow of War. https://www.shadowofwar.com/about/

Comcept Inc., Armature Studio LLC. (n.d.). Play Recore: Definitive edition. Xbox Cloud Gaming (Beta) on Xbox.com. https://www.xbox.com/en-AU/play/games/recore-definitive-edition/9NBLGGH1Z6FQ

Microsoft. Xbox. (n.d.). https://www.xbox.com/en-AU/games/store/microsoft-solitaire-collection/9wzdncrfhwd2

Figure A1: Medieval playing cards designed by Peter Flötner, c.1545. Sourced from https://www.wopc.co.uk/germany/peter-flotner,-c.1545

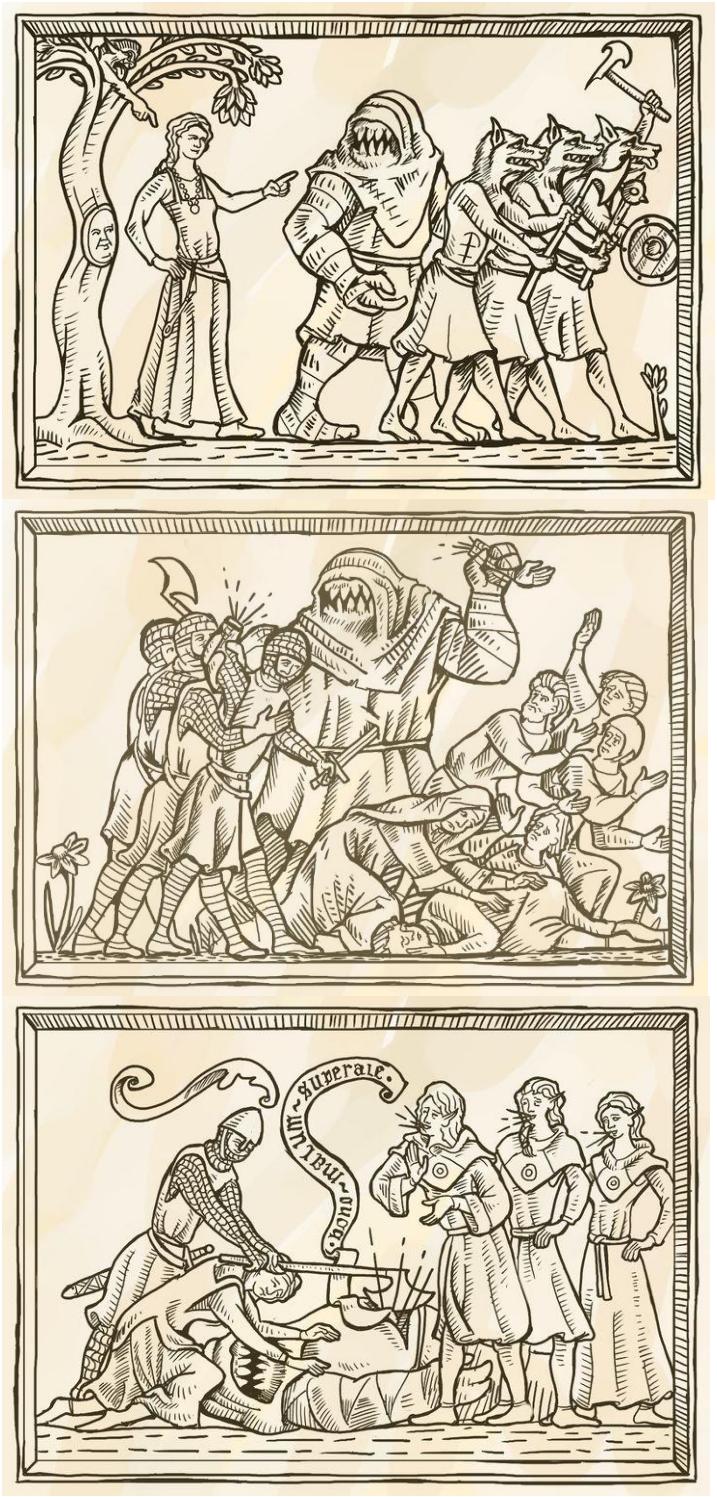


Figure A2, A3, A4: Medieval style artworks. Sourced from https://www.pinterest.com.au/pin/457537643384397793/

Appendix C: Full SFX list

MUSIC	Noise	Status
Main Menu	Tranquil background music	Confirmed
Stage 1	Medieval peaceful music	Confirmed
Stage 2	Rousing battle music	Confirmed
SOUND EFFECTS	Noise	Status
General Sounds		
Card drawn or flipped	Card flip	Confirmed
Card placed	Card flip sound	Confirmed
Pause	click	Confirmed Deemed unnecessary
Dice roll	Dice clatter	Confirmed
Shuffle	Shuffle	Confirmed
Resource played	Sawing hammer and nail	Deemed Unnecessary
Life lost	Impact of a kind	Deemed unnecessary
Stage 1 Sounds		
Intro	Sawing and wood noise	Deemed unnecessary
Bases:		
Saloon	Bar fight noise?	Deemed unnecessary
Pirate ship	Ship creaking	Deemed unnecessary
Observatory	Glass beaker clinking	Deemed unnecessary
Castle	Trumpet fanfare	Deemed unnecessary

Dojo	Martials art noises	Deemed unnecessary
Rocket	Rocket rumble	Deemed unnecessary
Resources:		
Base sound	Construction noise	Deemed unnecessary
Gold	gleam?	Deemed unnecessary
Steel	Steel rasp	Deemed unnecessary
Wood	Wood hit	Deemed unnecessary
Stone	Rocks clatter	Deemed unnecessary
Clay	Mud splat	Deemed unnecessary
Fabric	Fabric rumple	Deemed unnecessary
Glass	Glass smash	Deemed unnecessary
Actions:		
Time Freezer Potion	Time slow	Deemed unnecessary
General's Order	Gunshot	Deemed unnecessary
Kidnapped	Muffled protests	Deemed unnecessary
Poor Harvest	Dry plants crunching	Deemed unnecessary
Flood	Water rushing	Deemed unnecessary
Chapel Maintenance	Hammering and construction	Deemed unnecessary
Thief	Ratty chuckle	Deemed unnecessary
Shield	Shield fortify	Deemed unnecessary
Extra Pay	Coins clinking	Deemed unnecessary

Tax Collector	Pounding on door	Deemed unnecessary
Special:		
Curse	Evil exhale	Deemed unnecessary
Stage 2 Sounds		
Intro	Fighting sounds, sword clash and shields general battlefield noises	Deemed unnecessary
Weapons:		
*Mac*e	*Mac*e on shield clash	Deemed unnecessary
Katana	Katana air slash	Deemed unnecessary
Gun	Old gun fired	Deemed unnecessary
Bomb	Explosion	Deemed unnecessary
Wand	Spell sound	Deemed unnecessary
Cannon	Cannon fired	Deemed unnecessary
Win sound	Positive trumpet fanfare	Confirmed
Lose sound	Negative trumpet fanfare	In progress Deemed unnecessary

Table B1: The list of sound effects and their status.

Appendix D: Art priorities

Visuals	Image	Outstanding	Priority	Actioned
Game Components:				
Battle of Batone Logo	Correct pen thickness;		5th	Done
Icon / Favicon	Sword from the logo inside a circle with border		5th	Done
Back of the card			6th	Done
Glow		VERY IMPORTANT	1st	Done
Main Menu	Map of Batone	Not usable (the dimensions are incorrect); Need 1152 x 648 px	9th	Unfinished
Stage 1	Table background	Current one ok if you zoom in	1st	
	Stage one banner	Test Font? Scroll tile or banner by Isla		
	Scroll tiles	Not started - need 9 tiles (one image cut into 9 pieces)		
Stage 2	Table background	Same as stage two		
	Stage two banner			
	Scroll tiles	Same as stage one		
Dice	12 dice images (1-6 black/white) 2D face / needs to be congruent	Not started	7th	Done
Life	Heart card	what that looks like	8th	Done
Fortresses:		Border of cards		
Saloon	Saloon	Check all spelling / text sizing	3rd	Done
Pirate ship	Pirate ship	Check all spelling / text sizing	3rd	Done
Observatory	Observatory	Check all spelling / text sizing	3rd	Done
Castle	Castle	Check all spelling / text sizing	3rd	Done
Dojo	Dojo	Check all spelling / text sizing	3rd	Done
Rocket	Rocket	Check all spelling / text sizing	3rd	Done
Resources:				
Gold	Gold	Border and card colour discussion	4th	Done
Steel	Steel	Border and card colour discussion	4th	Done
Wood	Wood	Border and card colour discussion	4th	Done
Stone	Stone	Border and card colour discussion	4th	Done
Clay	Clay	Border and card colour discussion	4th	Done
Fabric	Fabric	Border and card colour discussion	4th	Done
Glass	Glass	Border and card colour discussion	4th	Done
Actions:				
Time Freezer Potion	Potion bottle	label good; text heading and body	2nd	Done
General's Order	Trumpet	attachments for flag; text heading and body	1st	Done
Kidnapped	Missing poster	Text heading and body	2nd	Done
Poor Harvest	Grasshopper on wheat	Text heading and body	2nd	Done
Flood	Wave	Text heading and body	2nd	Done
Chapel Maintenance	Chapel	Text heading and body	2nd	Done
Thief	Hand taking jewel	Text heading and body	2nd	Done
Shield	4	a bit plain; Text heading and body	1st	Done
Extra Pay	Coin	Text heading and body	2nd	Done
Tax Collector	Bag and coin	Needs more wealth and fix cup size (too small)	1st	Done
Curse	Skull	Decide between purple or white (White); Text heading and body		Done
STAGE 2 Weapons:	Not to use for time constraints and it detracts from this year's theme			
Mace				
Katana				
Gun				
Bomb				
Wand				
Cannon				
Win		Crown??		
Lose	No lose screen or image			
Fonts	Write in all fonts in the GDD and license them			Done
Roles/Characters:				
Wizard	Hat or weapon	Hat		Done
Pirate	Hat or weapon	Hat		Done
Ninja	Hat or weapon	Hat		Done
Cowboy	Hat or weapon	Hat		Done
Knight	Hat or weapon	Hat		Done
Astronaut	Hat or weapon	Hat		Done
Text on Action Cards	Font same; Left or centred? Must start at same point		All the way	Done

Table D1: Art priorities checklist

Appendix E: Original vs New Battle of Batone cards

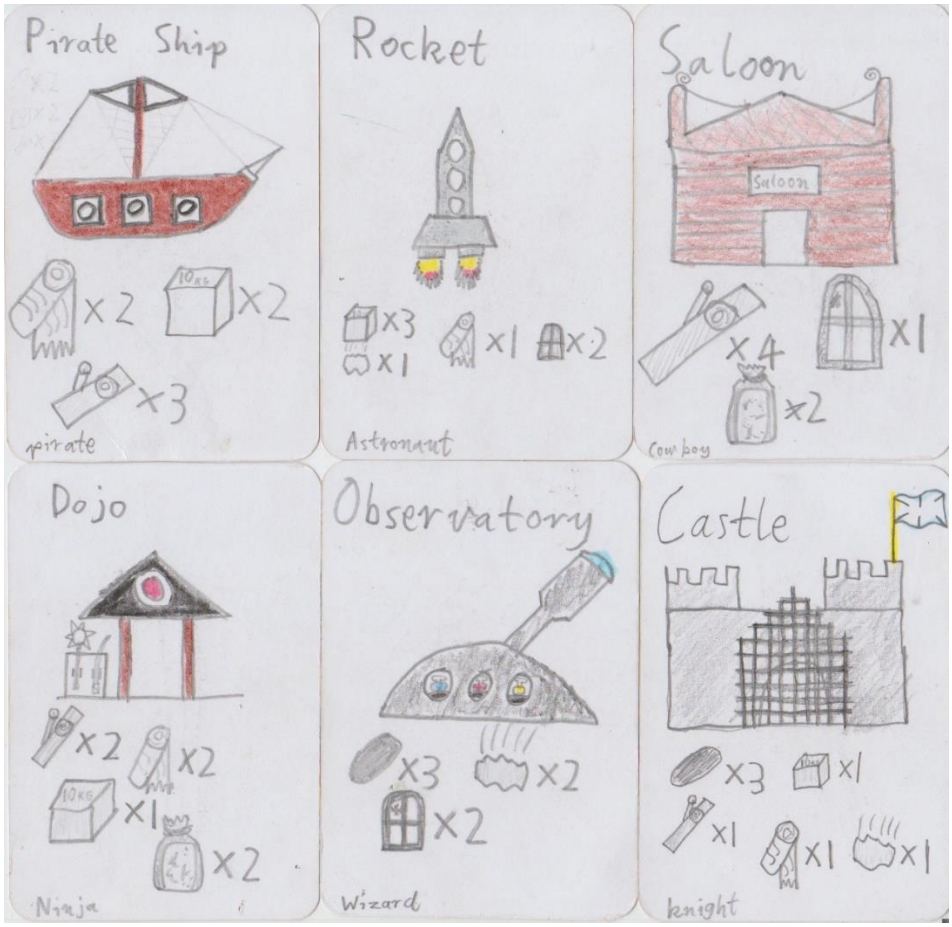


Figure E1: These were the original fortresses drawings from the physical version of Battle of Batone.

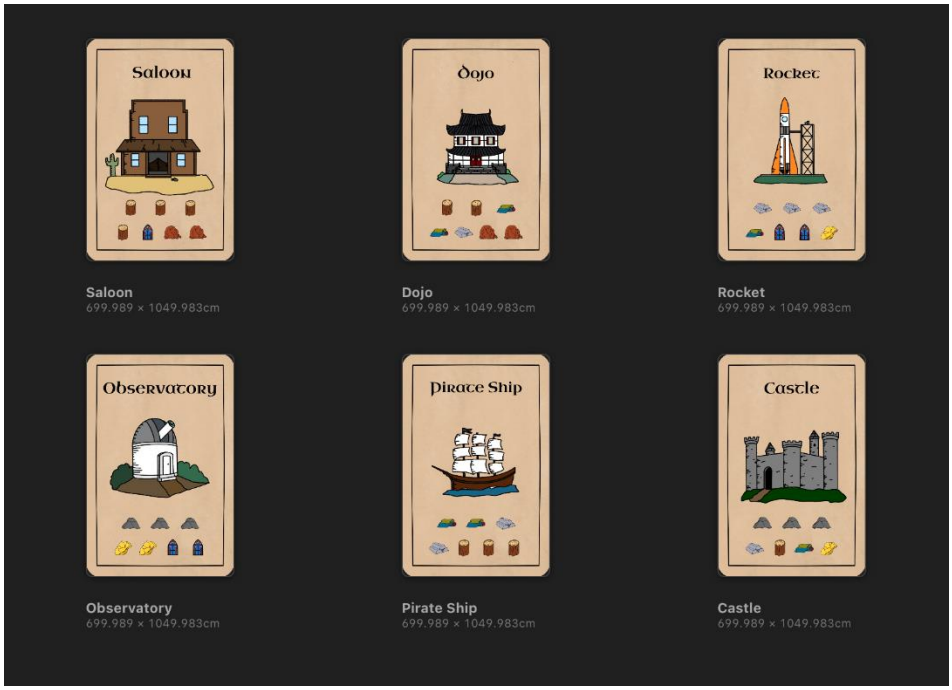


Figure E2: New fortress images.

Figure E3: Original resource illustrations.

Figure E4: New resource cards.

Figure E5: These are the original action cards we used to design our digital versions.

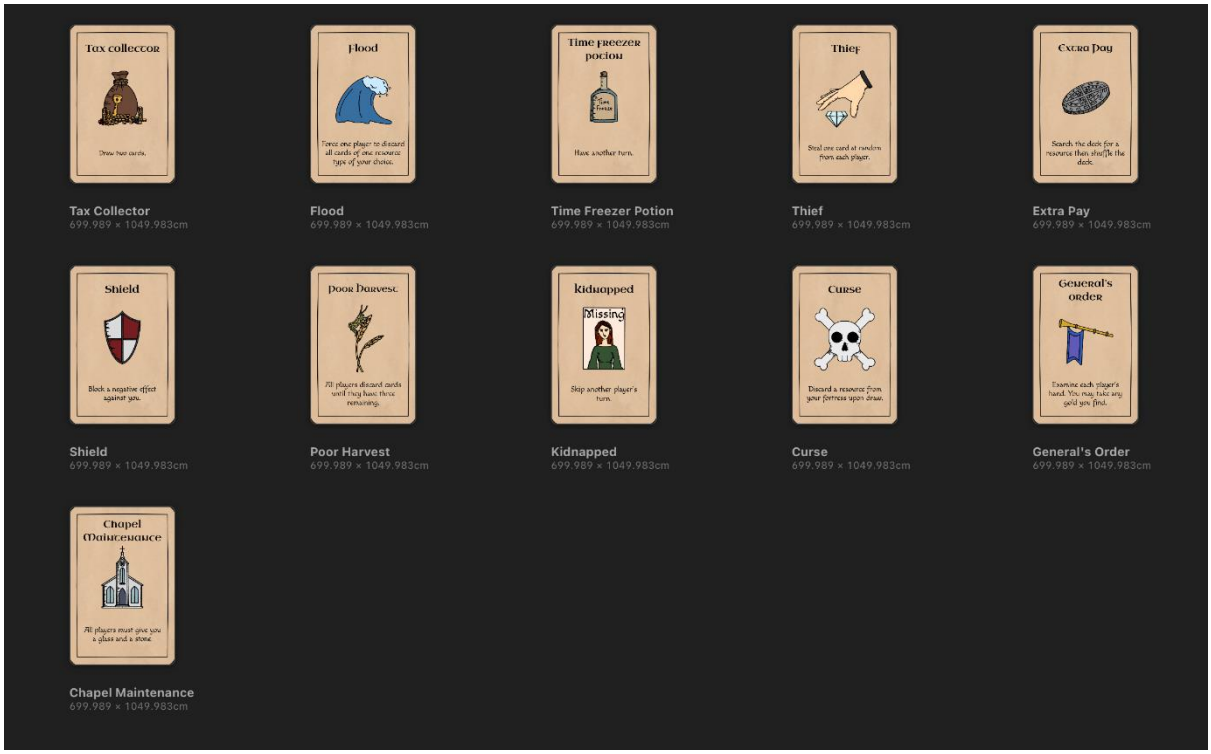


Figure E6: New action cards.

Appendix F: Naming conventions

Naming Conventions

File Names:

item_format_version
laser_sound_3.6

Version Control:

All numbers representing the version of the file must be written with one decimal place, no more no less. For example, 0.0, 0.5, 2.0, 36.9.

Folder Names:

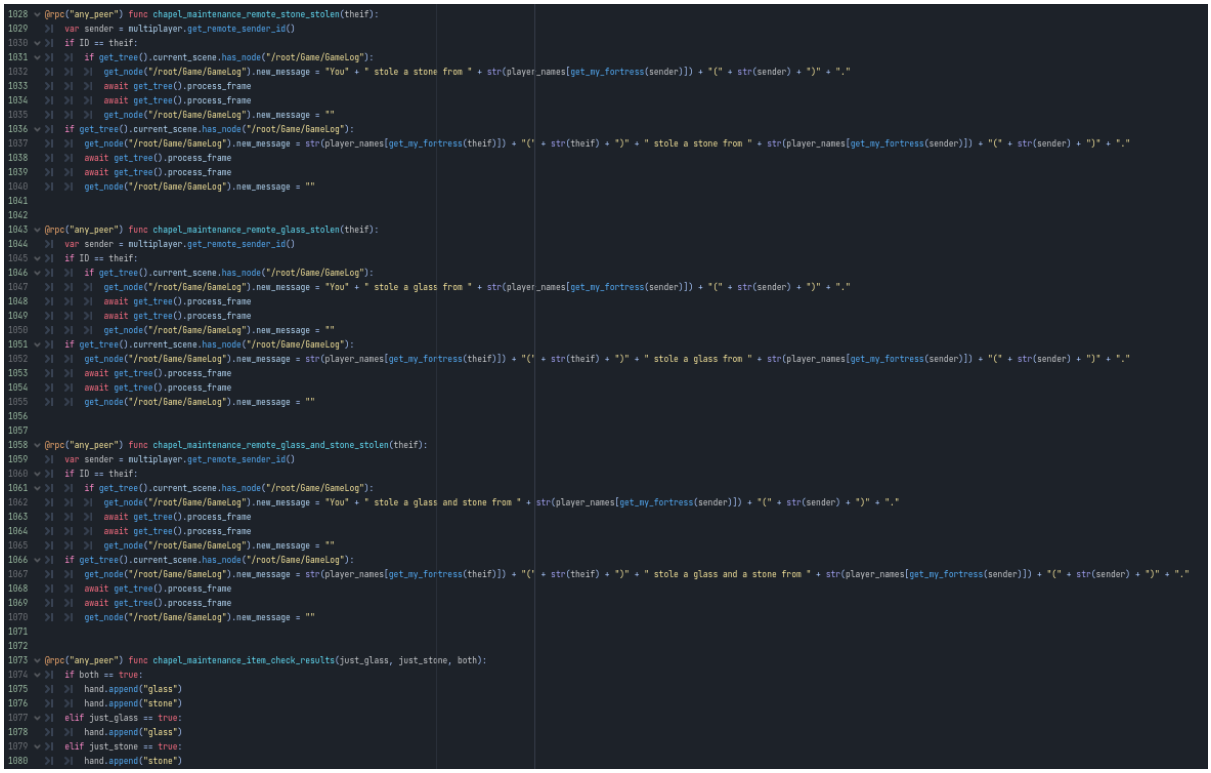
All words must begin with a capital, with a space between each word. For example, Laser Sounds, or Flame Images.

Appendix G: Scripts

```gdscript
func chapel_maintenance(_my_fortress, button_name, _action_card):
	hand.remove_at(int(button_name))
	discard.append("chapel_maintenance")
	rpc("update_discard", discard)
	if get_tree().current_scene.has_node("/root/Game/GameLog"):
		get_node("/root/Game/GameLog").new_message = "You played a Chapel Maintenance."
		await get_tree().process_frame
		await get_tree().process_frame
		get_node("/root/Game/GameLog").new_message = ""
		rpc("chapel_maintenance_played")
		await get_tree().process_frame
		await get_tree().process_frame
		for c in connected_peer_ids:
			if c == ID:
				pass
			else:
				rpc_id(c, "remote_shield_check")
				await get_tree().process_frame
				await get_tree().process_frame
				if shield_check != null:
					if shield_check == true:
						rpc_id(c, "remote_play_shield", "Chapel Maintenance")
						await shielded
						if shield_played == true:
							log_shield(c)
							shield_check = null
						else:
							rpc_id(c, "chapel_maintenance_item_check")
							shield_check = null
					else:
						rpc_id(c, "chapel_maintenance_item_check")
						shield_check = null
```

```gdscript
@rpc("any_peer") func chapel_maintenance_item_check():
	var sender = multiplayer.get_remote_sender_id()

	# Glass and stone.

	if hand.find("glass") and hand.find("stone"):
		hand.erase("glass")
		hand.erase("stone")
		if get_tree().current_scene.has_node("/root/Game/GameLog"):
			get_node("/root/Game/GameLog").new_message = str(player_names[get_my_fortress(sender)]) + "(" + str(sender) + ")" + " stole a glass and a stone from you."
			await get_tree().process_frame
			await get_tree().process_frame
			get_node("/root/Game/GameLog").new_message = ""
		rpc("chapel_maintenance_remote_glass_and_stone_stolen", sender)
		rpc_id(sender, "chapel_maintenance_item_check_results", false, false, true)

	# Glass.

	elif hand.find("glass"):
		hand.erase("glass")
		if get_tree().current_scene.has_node("/root/Game/GameLog"):
			get_node("/root/Game/GameLog").new_message = str(player_names[get_my_fortress(sender)]) + "(" + str(sender) + ")" + " stole a glass from you."
			await get_tree().process_frame
			await get_tree().process_frame
			get_node("/root/Game/GameLog").new_message = ""
		rpc("chapel_maintenance_remote_glass_stolen", sender)
		rpc_id(sender, "chapel_maintenance_item_check_results", true, false, false)

	# Stone.

	elif hand.find("stone"):
		hand.erase("stone")
		if get_tree().current_scene.has_node("/root/Game/GameLog"):
			get_node("/root/Game/GameLog").new_message = str(player_names[get_my_fortress(sender)]) + "(" + str(sender) + ")" + " stole a stone from you."
			await get_tree().process_frame
			await get_tree().process_frame
			get_node("/root/Game/GameLog").new_message = ""
		rpc("chapel_maintenance_remote_stone_stolen", sender)
		rpc_id(sender, "chapel_maintenance_item_check_results", false, true, false)
```

Figure G1, G2: Outdated Chapel Maintenance Code



```
@rpc("any_peer") func chapel_maintenance_remote_stone_stolen(theif):
    var sender = multiplayer.get_remote_sender_id()
    if ID == theif:
        if get_tree().current_scene.has_node("/root/Game/GameLog"):
            get_node("/root/Game/GameLog").new_message = "You" + " stole a stone from " + str(player_names[get_my_fortress(sender)]) + "(" + str(sender) + ")" + "."
            await get_tree().process_frame
            await get_tree().process_frame
            get_node("/root/Game/GameLog").new_message = ""
    if get_tree().current_scene.has_node("/root/Game/GameLog"):
        get_node("/root/Game/GameLog").new_message = str(player_names[get_my_fortress(theif)]) + "(" + str(theif) + ")" + " stole a stone from " + str(player_names[get_my_fortress(sender)]) + "(" + str(sender) + ")" + "."
        await get_tree().process_frame
        await get_tree().process_frame
        get_node("/root/Game/GameLog").new_message = ""


@rpc("any_peer") func chapel_maintenance_remote_glass_stolen(theif):
    var sender = multiplayer.get_remote_sender_id()
    if ID == theif:
        if get_tree().current_scene.has_node("/root/Game/GameLog"):
            get_node("/root/Game/GameLog").new_message = "You" + " stole a glass from " + str(player_names[get_my_fortress(sender)]) + "(" + str(sender) + ")" + "."
            await get_tree().process_frame
            await get_tree().process_frame
            get_node("/root/Game/GameLog").new_message = ""
    if get_tree().current_scene.has_node("/root/Game/GameLog"):
        get_node("/root/Game/GameLog").new_message = str(player_names[get_my_fortress(theif)]) + "(" + str(theif) + ")" + " stole a glass from " + str(player_names[get_my_fortress(sender)]) + "(" + str(sender) + ")" + "."
        await get_tree().process_frame
        await get_tree().process_frame
        get_node("/root/Game/GameLog").new_message = ""


@rpc("any_peer") func chapel_maintenance_remote_glass_and_stone_stolen(theif):
    var sender = multiplayer.get_remote_sender_id()
    if ID == theif:
        if get_tree().current_scene.has_node("/root/Game/GameLog"):
            get_node("/root/Game/GameLog").new_message = "You" + " stole a glass and stone from " + str(player_names[get_my_fortress(sender)]) + "(" + str(sender) + ")" + "."
            await get_tree().process_frame
            await get_tree().process_frame
            get_node("/root/Game/GameLog").new_message = ""
    if get_tree().current_scene.has_node("/root/Game/GameLog"):
        get_node("/root/Game/GameLog").new_message = str(player_names[get_my_fortress(theif)]) + "(" + str(theif) + ")" + " stole a glass and a stone from " + str(player_names[get_my_fortress(sender)]) + "(" + str(sender) + ")" + "."
        await get_tree().process_frame
        await get_tree().process_frame
        get_node("/root/Game/GameLog").new_message = ""


@rpc("any_peer") func chapel_maintenance_item_check_results(just_glass, just_stone, both):
    if both == true:
        hand.append("glass")
        hand.append("stone")
    elif just_glass == true:
        hand.append("glass")
    elif just_stone == true:
        hand.append("stone")
```

```
@rpc("any_peer") func chapel_maintenance_played():
    var sender = multiplayer.get_remote_sender_id()
    if get_tree().current_scene.has_node("/root/Game/GameLog"):
        get_node("/root/Game/GameLog").new_message = str(player_names[get_my_fortress(sender)]) + "(" + str(sender) + ")" + " played a Chapel Maintenance."
        await get_tree().process_frame
        await get_tree().process_frame
        get_node("/root/Game/GameLog").new_message = ""
```

Figure G3, G4: Outdated Chapel Maintenance Code

```gdscript
func chapel_maintenance(_my_fortress, button_name, _action_card):
	last_action_played = "chapel_maintenance"
	hand.remove_at(int(button_name))
	discard.append("chapel_maintenance")
	rpc("update_discard", discard)
	rpc("chapel_maintenance_log_rpc")
	rpc("remote_player_shield_check", last_action_played)


func chapel_maintenance_log():
	if turn_order[0] == ServerManager.ID:
		if get_tree().current_scene.has_node("/root/Game/GameLog"):
			get_node("/root/Game/GameLog").new_message = "You played Chapel Maintenance."
			get_node("/root/Game/GameLog").log_message()
	else:
		if get_tree().current_scene.has_node("/root/Game/GameLog"):
			get_node("/root/Game/GameLog").new_message = str(player_names[get_my_fortress(turn_order[0])]) + " played Chapel Maintenance."
			get_node("/root/Game/GameLog").log_message()


@rpc("call_local", "any_peer") func chapel_maintenance_log_rpc():
	chapel_maintenance_log()


func continue_chapel_maintenance(shield_played_is_true, remote_id):
	print("continue_chapel_maintenance")
	if shield_played_is_true == false:
		print("if shield_played_is_true == false:")
		rpc_id(remote_id, "chapel_maintenance_resource_checker")


@rpc("any_peer") func chapel_maintenance_resource_checker():
	print("chapel_maintenance_resource_checker")
	var sender = multiplayer.get_remote_sender_id()
	var cm_resource_list = []
	if hand.has("stone"):
		hand.erase("stone")
		cm_resource_list.append("stone")
	if hand.has("glass"):
		hand.erase("glass")
		cm_resource_list.append("glass")
	rpc_id(sender, "chapel_maintenance_resource_check_results", cm_resource_list)
```

Figure G5: *Current Chapel Maintenance Code*

Current Chapel Maintenance Code

```
@rpc("any_peer") func chapel_maintenance_resource_check_results(result_list):
    var sender = multiplayer.get_remote_sender_id()
    print(chapel_maintenance_resource_check_results)
    if result_list.is_empty() == false:
        for item in result_list:
            hand.append(item)
        if result_list.has("glass") and result_list.has("stone"):
            rpc("chapel_maintenance_resource_check_results_log_rpc", false, false, true, sender)
        elif result_list.has("glass"):
            rpc("chapel_maintenance_resource_check_results_log_rpc", true, false, false, sender)
        elif result_list.has("stone"):
            rpc("chapel_maintenance_resource_check_results_log_rpc", false, true, false, sender)


func chapel_maintenance_resource_check_results_log(cm_glass, cm_stone, cm_both, remote_id, sender):
    if sender == ID:
        if cm_both == true:
            if get_tree().current_scene.has_node("/root/Game/GameLog"):
                get_node("/root/Game/GameLog").new_message = "You took a glass and a stone from " + str(get_player_name(remote_id)) + "."
                get_node("/root/Game/GameLog").log_message()
        elif cm_glass == true:
            if get_tree().current_scene.has_node("/root/Game/GameLog"):
                get_node("/root/Game/GameLog").new_message = "You took a glass from " + str(get_player_name(remote_id)) + "."
                get_node("/root/Game/GameLog").log_message()
        elif cm_stone == true:
            if get_tree().current_scene.has_node("/root/Game/GameLog"):
                get_node("/root/Game/GameLog").new_message = "You took a stone from " + str(get_player_name(remote_id)) + "."
                get_node("/root/Game/GameLog").log_message()
    elif remote_id == ID:
        if cm_both == true:
            if get_tree().current_scene.has_node("/root/Game/GameLog"):
                get_node("/root/Game/GameLog").new_message = str(get_player_name(sender)) + " took a glass and a stone from you."
                get_node("/root/Game/GameLog").log_message()
        elif cm_glass == true:
            if get_tree().current_scene.has_node("/root/Game/GameLog"):
                get_node("/root/Game/GameLog").new_message = str(get_player_name(sender)) + " took a glass from you."
                get_node("/root/Game/GameLog").log_message()
        elif cm_stone == true:
            if get_tree().current_scene.has_node("/root/Game/GameLog"):
                get_node("/root/Game/GameLog").new_message = str(get_player_name(sender)) + " took a stone from you."
                get_node("/root/Game/GameLog").log_message()
    else:
        if cm_both == true:
            if get_tree().current_scene.has_node("/root/Game/GameLog"):
                get_node("/root/Game/GameLog").new_message = str(get_player_name(sender)) + " took two cards from " + str(get_player_name(remote_id)) + "."
                get_node("/root/Game/GameLog").log_message()
        elif cm_glass == true:
            if get_tree().current_scene.has_node("/root/Game/GameLog"):
                get_node("/root/Game/GameLog").new_message = str(get_player_name(sender)) + " took a card from " + str(get_player_name(remote_id)) + "."
                get_node("/root/Game/GameLog").log_message()
        elif cm_stone == true:
            if get_tree().current_scene.has_node("/root/Game/GameLog"):
                get_node("/root/Game/GameLog").new_message = str(get_player_name(sender)) + " took a card from " + str(get_player_name(remote_id)) + "."
                get_node("/root/Game/GameLog").log_message()
```

```
@rpc("call_local", "any_peer") func chapel_maintenance_resource_check_results_log_rpc(cm_glass, cm_stone, cm_both, remote_id):
    var sender = multiplayer.get_remote_sender_id()
    chapel_maintenance_resource_check_results_log(cm_glass, cm_stone, cm_both, remote_id, sender)
```

Figure G6, G7: Current Chapel Maintenance Code

Appendix H: Notes

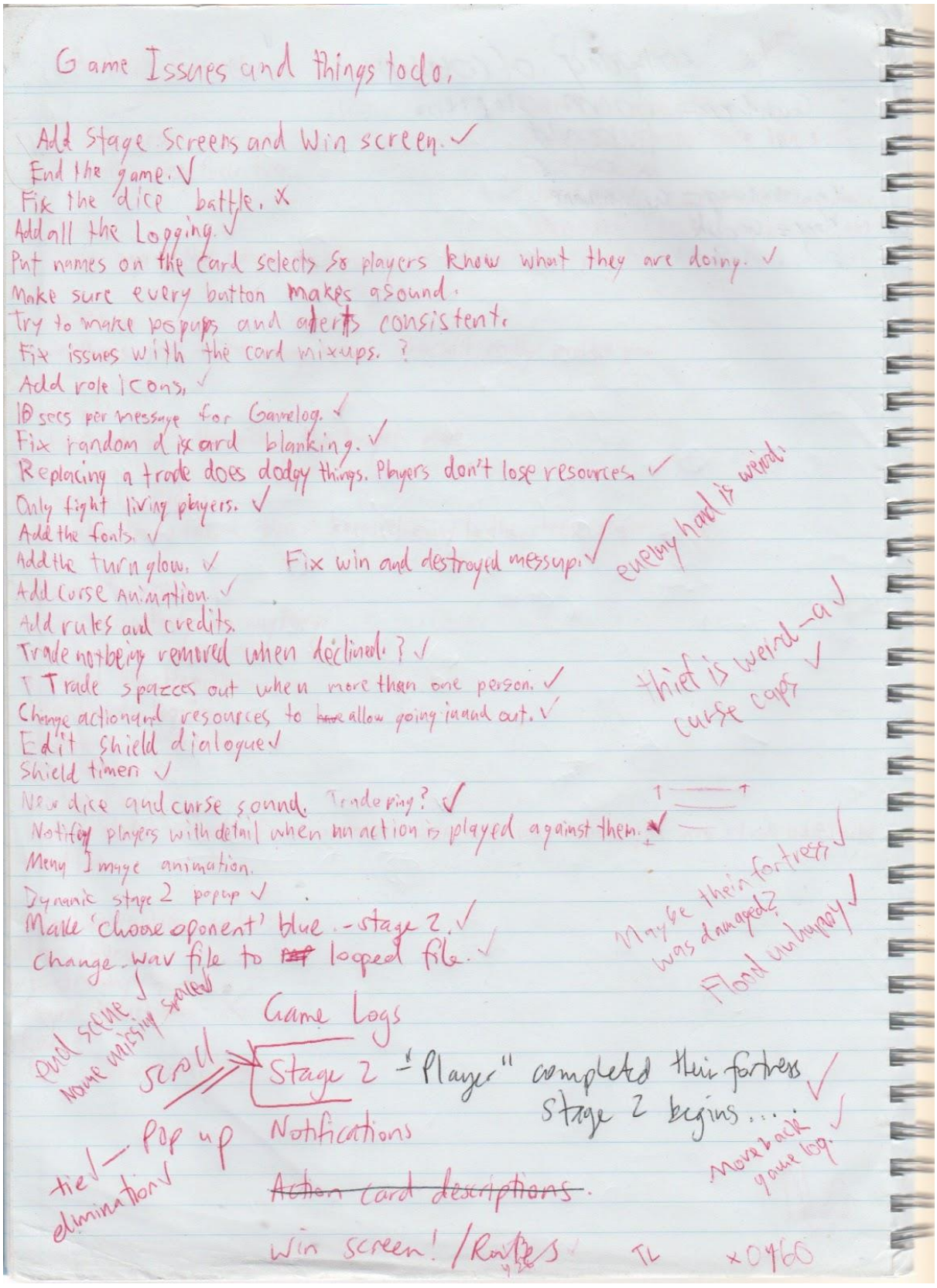


Figure H1: Scan of notes.