Command Input

Jeffrey Popek
EDG-220-01
Project 3 Team 3
Sprint 5

Table of Contents

Delivery Platform

Our game, Command Input, will be played on PC with the alternative controller being a drawing tablet. The game can still be played with the mouse but the main mechanics are much easier with the drawing tablet. Our game is aimed at PC players that have access to a drawing tablet. Another one of our target audiences is mobile game players since they will have a device that can detect drawn input built in.

Development Environment

Unity

Our game was developed in the Unity Engine because of our team's experience with the engine and also the amount of documentation available online. Unity allows us to easily build our game and let a wide range of players play our game. Unity is also easily accessible for each role in our team. We are using Unity version **2021.3.17f**.

Git

Our team will use git for version control because it is supported by Redmine. Most of the team was somewhat experienced with using git from previous projects. Applications such as GitKraken or GitBash will help simplify the version control process for less experienced git users.

Google Drive

Our team uses Google Drive to store and create documentation before finalizing it to be submitted to Redmine. Google Drive is easily accessible to everyone in the team and Champlain gives students a large amount of storage to use for projects. Google Drive also allows each team member to easily access and edit any existing documents for easy collaboration. We can also use redmine’s macros to show our documents directly in the wiki.

Audacity

Since we do not have a sound designer for this project, we are using Audacity to edit sounds. Audacity is the best option for our team because it is easily accessible and free.

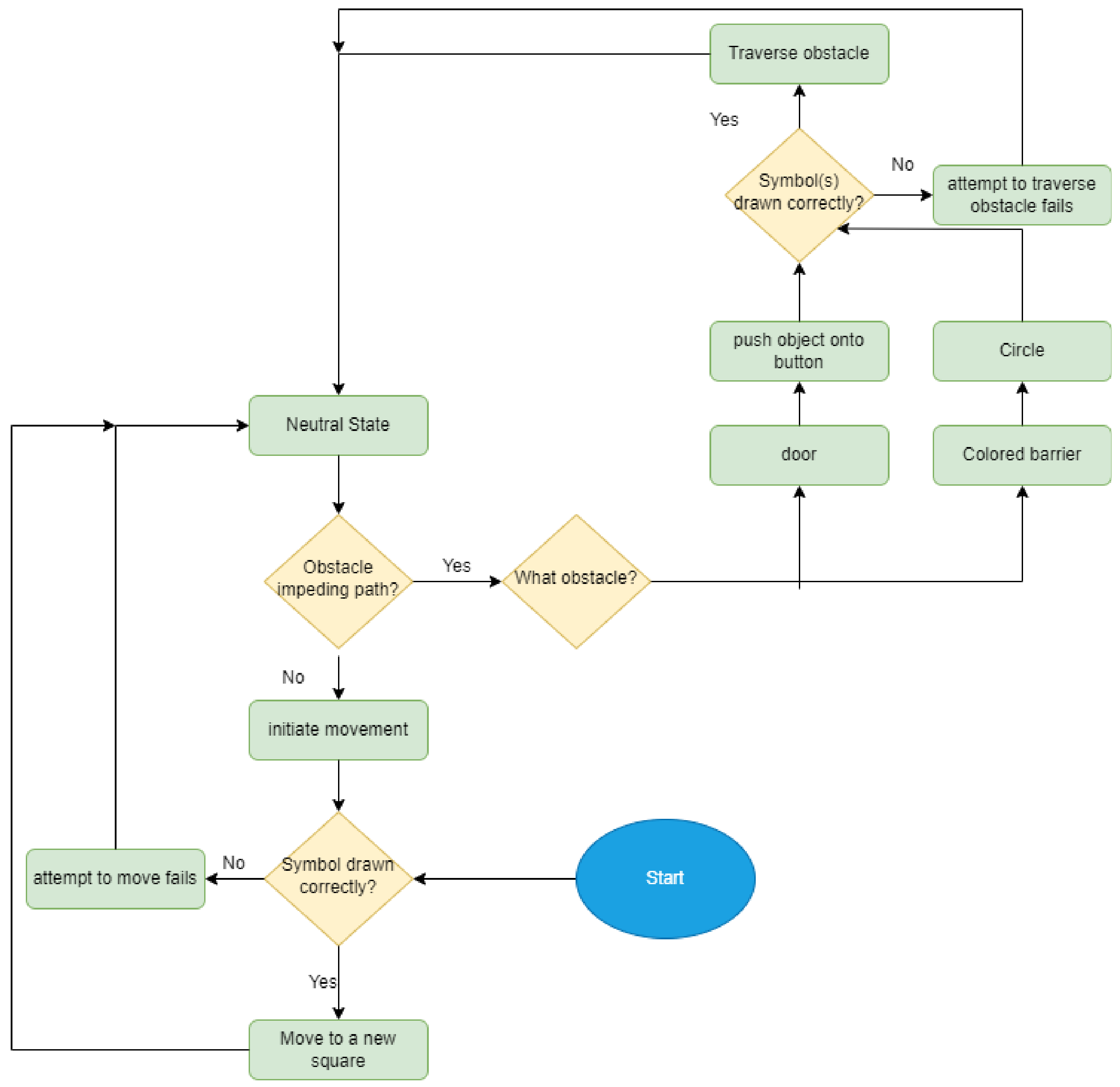
Photoshop

Our artist uses Photoshop to create and edit their art. This program makes creating and implementing art as easy as possible for them.

Rider

Our Programmer used Rider to create scripts for the game. Rider is our best choice because it is easily accessible, free for college students with the github student package, and works well with unity.

Logical Flow Diagram

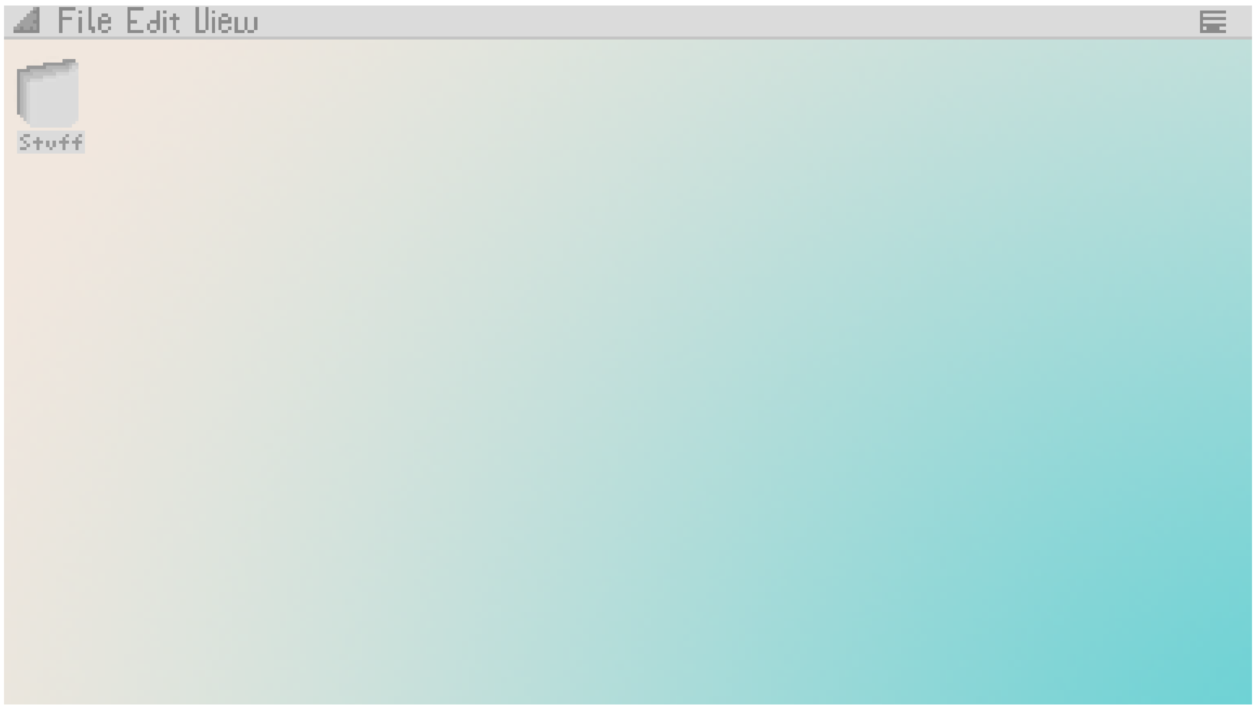


Game Mechanics and Systems

UI

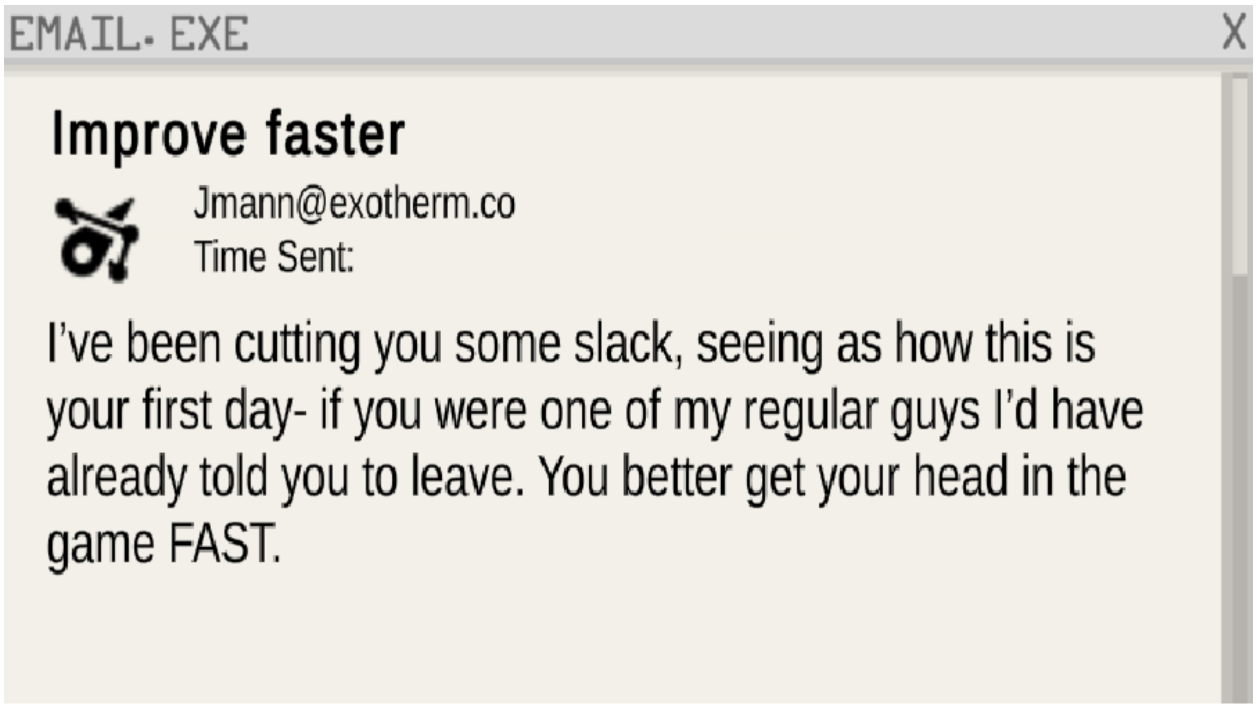
Our game is based around being on a macintosh computer so the UI will be a big part of the game. The tutorial will be themed around the macintosh era so pop ups would be emails from your boss. The UI of the game would be very similar to the menus since the game takes place on a macintosh computer. The hardest part of creating the menus and UI would be creating art, implementing it, and making it all look cohesive.

Risk: Medium



Emails

The emails of our game are connected to the UI but are different enough to be its own thing. The emails in our game are the tutorials of the game. In most levels you will receive an email from your boss giving you instructions. The emails are all themed around our game being in an office setting. The hardest part of creating the emails would be creating the UI engine for it and writing the email text.



Improve faster

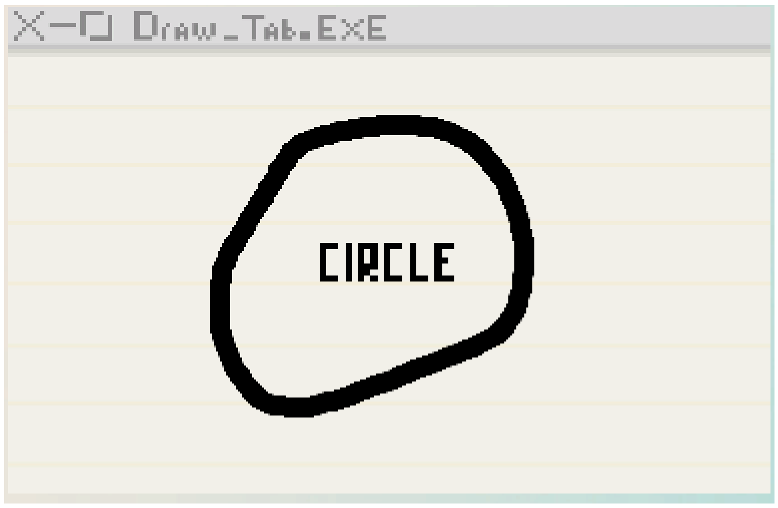
Jmann@exotherm.co
Time Sent:

I've been cutting you some slack, seeing as how this is your first day- if you were one of my regular guys I'd have already told you to leave. You better get your head in the game FAST.

Risk: Low

Input

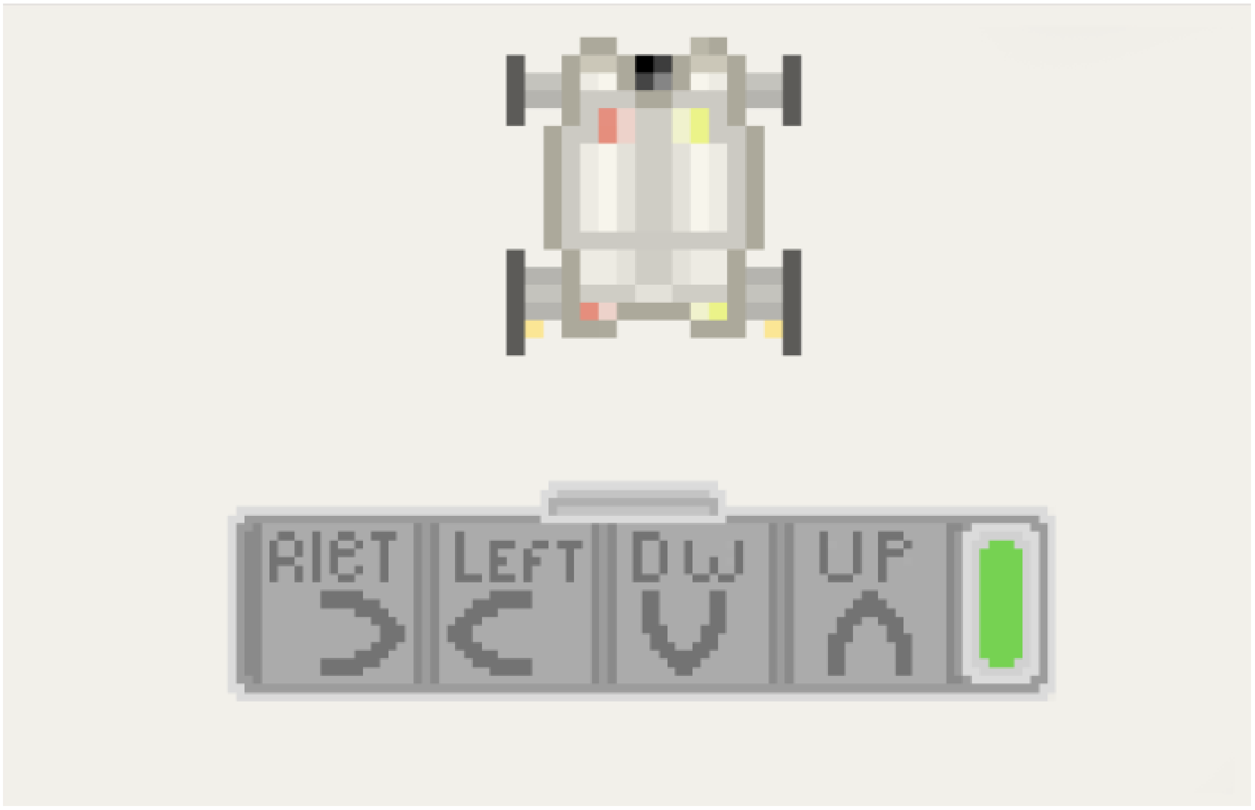
Our alternative input is the drawing tablet and drawing symbols on it is the core mechanic to our game. Getting the drawing tablet to work is very simple since it is basically plug and play. Getting the tablet to draw in unity will be a bit harder but not impossible since its functions are somewhat similar to a mouse. The drawing area for the game will be the entire screen. The hardest part will be recognizing symbols and having them each do different actions in game. Our team is using a unity asset that helps with the symbol recognition but it will still be a challenge to integrate it into our game.



Risk: High

Movement

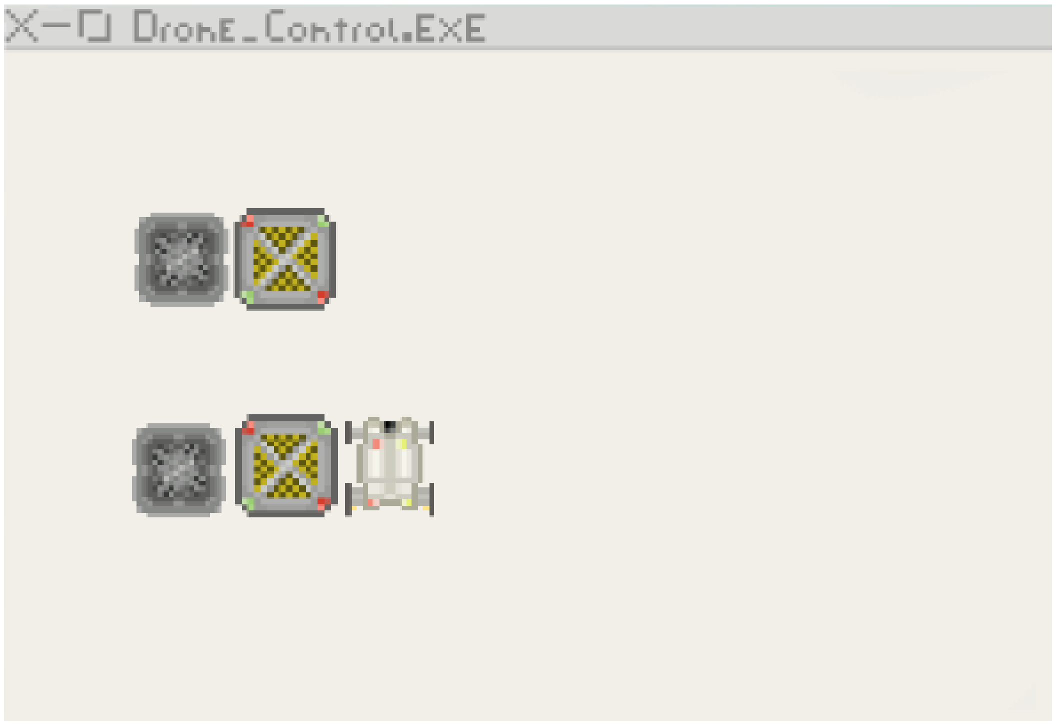
The movement of our game is directly connected to the drawing input of our game. The movement of our game will only move when a symbol is recognized. Currently the movement of our character is simple but will become more complex as we add more mechanics. The hardest part of this would be connecting the movement to the symbols.



Risk: Medium

Obstacles

Obstacles will be the way we create challenges in our game. There will be a variety of obstacles. The main ones we have designed at the moment are buttons plus moveable objects to move over the buttons and walls that switch based on the symbol drawn. As we move along the project timeline we will add more complexity to our obstacles and increase the difficulty of the game. The hardest part of this would be designing and programming the obstacles to work well with our game and fit our theme.



Risk: Medium

Pipelines

Art Pipeline

Creating the Art

Our artist will create their art in a default photoshop template, it can be scaled down later if necessary.

Implementation

1. Have the art you want to add to the repo ready in the correct format. (.png)

2. Pull the git repo to make sure your project is up to date. If you have any git issues stop and contact the programmer unless you know exactly what the issue is.
3. Copy your file(s) into the project folder Assets/Sprites. There are a few different folders so pick one that fits the type of art you are adding (there is Environment, UI, and NPC).
4. Once the art is in you can commit and push to the repo.

Audio Pipeline

Finding Audio

Since we have no sound designer for this project we will find all our audio online. Anyone can look for audio and put it into the shared drive folder.

Editing Audio

Once a sound is found we will determine whether it needs to be edited or not. If so we will bring it into Audacity where we can edit it.

Implementation

1. Have the sound(s) you want to add to the repo ready in the correct format. (.wav)
2. Pull the git repo to make sure your project is up to date. If you have any git issues stop and contact the programmer unless you know exactly what the issue is.
3. Copy your file(s) into the project folder Assets/Sounds. There are a few different folders so pick one that fits the type of art you are adding (there is Environment, UI, and NPC).
4. Once the sound(s) are in you can commit and push to the repo.

Design Pipeline

Design Stage

First our designers will come up with an idea and get it written down. Next they will discuss with the team if the idea is in scope or not. If it is then they can begin to polish the idea and ask the artists or the programmer for any additional tools.

In Engine

We want the designers to have an easy time creating our map so we will be using prefabs. Prefabs allow our designers to drag and drop premade assets into the scene and still be able to change small aspects. The prefabs will be created by the programmers with the input from the designers and the artists.

Implementation

1. When you have an idea you want implemented you must first run it through the team first to see if it is possibly in scope for this project.
2. If in scope then the programmers and artists will begin to create assets. If the idea is out of scope you can scrap it or try and rework it to be in scope, repeat from step one when reworking your ideas.
3. Pull the git repo to make sure your project is up to date. If you have any git issues stop and contact the programmer unless you know exactly what the issue is.
4. Find out from your programmers and artists what tools and assets were created to implement your idea.
5. Create your design in the scene and save it as a prefab.
6. Commit and push to repo.

Sprint Updates

Sprint 1 - Game Concepts

Deliverables

- Visual Design Document
    - A Guide for our game’s mechanics.

- Game Loop Flow Chart
    - Explains the flow of our game and how each action would lead to the next.

- Game Concept Document
    - Explains the concept of our game in detail.

- Art Concept Document
    - Possible art directions for our game to go in.

Goals for next Sprint

- Maintain documentation
- Have a working digital prototype to show the game’s core mechanics.

Sprint 2 - Digital Prototype

Deliverables

- Game Build
    - A working prototype of our game showing off the core mechanic.

- Updated Documentation

    - Maintained each of our documents from the previous sprint.

- Finalized Documentation
    - Finalized documents that do not need to be iterated anymore.

- Visual Design Guide
    - Explains how our game will play out and what it takes to get something implemented.

- Art Document
    - Decided on one art direction to take our game in and put it into a document.

- Testing Plan
    - Team prepares to have our game tested in the GTL with documentation and a prototype.

Goals for next Sprint

- Maintain documentation.
- Continue working on digital prototype.
- Have our game tested at the GTL.

Sprint 3 - Continue Digital Prototype

Deliverables

- Game Build
    - A working prototype of our game showing off the core mechanics.

- Updated Documentation
    - Maintained each of our documents from the previous sprint.

- Finalized Documentation
    - Finalized documents that do not need to be iterated anymore.

- Implementing Art
    - Artists implement art into game.

- Bring game to GTL
    - Have game tested at GTL and get feedback

Goals for next Sprint

- Maintain documentation.
- Continue working on digital prototype
- Bring game to GTL

Sprint 4 - Continue Digital Prototype

Deliverables

- Game Build
    - A working prototype of our game showing off the core mechanics.

- Updated Documentation
    - Maintained each of our documents from the previous sprint.

- Bring game to GTL
    - Have game tested at GTL and get feedback

Goals for next Sprint

- Complete documentation
- Complete digital prototype

Sprint 5 - Complete Digital Prototype

Deliverables

- Game Build
    - A working demo of our game showing off the core mechanics.

- Completed Documentation
    - Every piece of documentation will be completed and submitted to the redmine wiki.

- Final Pitch Presentation
    - The team will present our game to the class in a pitch style presentation.