

# Technical Document

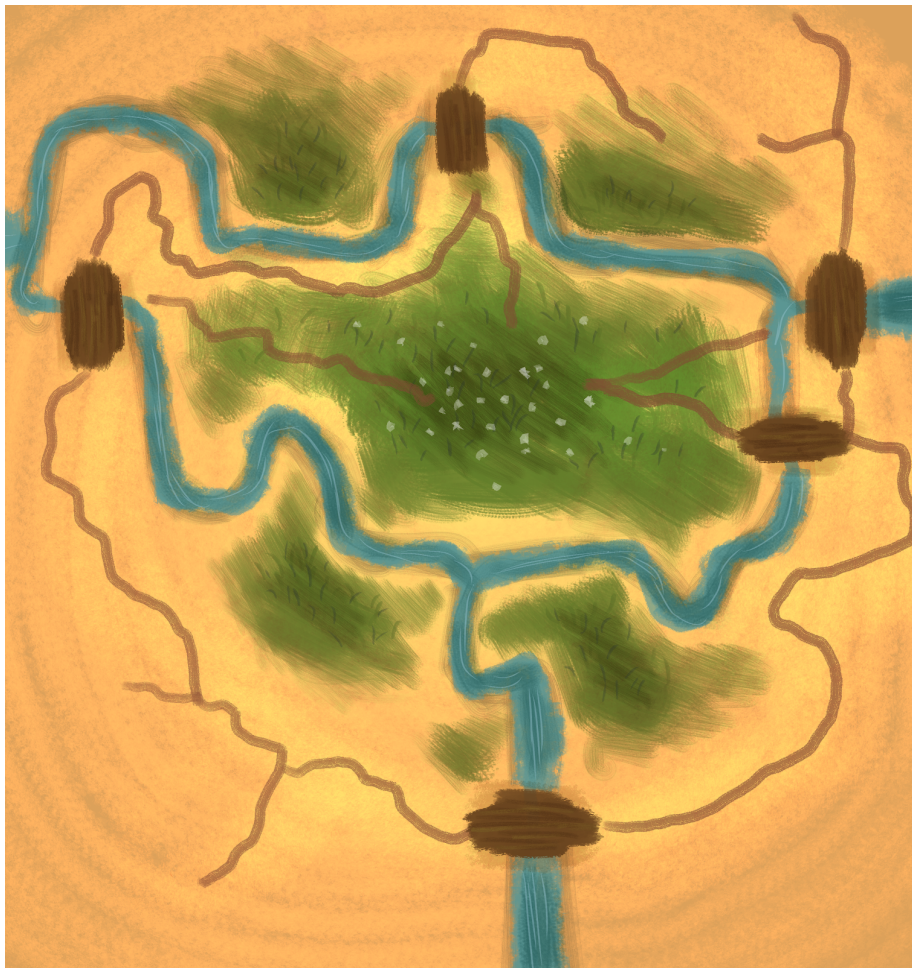
Section 5 Team 4 Sprint 3

**Author:** Amelia Payne

**Team Members:** Brooke Pendleton, Matthew Randolph, Alejandro Shaw-Correa

## Jinn's Curse

January 28, 2019








# Table of Contents

<b>Table of Contents</b>	<b>2</b>
<b>Difficulty Rating</b>	<b>5</b>
<b>Delivery Platform</b>	<b>6</b>
Windows PC	6
Mac/Linux	6
IOS/Android	6
Steam	7
<b>Development Environment</b>	<b>8</b>
Unity	8
<b>Architecture</b>	<b>9</b>
<b>Logical Flow Diagram</b>	<b>10</b>
Abstract Model	10
Turn Flow	11
<b>Game Mechanics and Systems</b>	<b>13</b>
Spell Phase	13
Player given income	13
Buying Inventory	13
Casting Spells	14
Exiting Spell Mode	14
Move Phase	15
Capture a Temple	15
Harvest a Farm	16
Exiting Move Phase	16
<b>Art Pipeline</b>	<b>17</b>
<b>Design Pipeline</b>	<b>18</b>
Global Data	18
Spells Scriptable Object	18
Spell Buttons	18
Temple Script	19

Player Data	19
Player Manager	19
Inventory Script	19
Spell Manager	19
Other Features	20
<b>Milestone Updates</b>	<b>21</b>
Sprint 1:	21
Sprint 2:	21
<b>Sources:</b>	<b>22</b>
Clock Icon	22
Information	22

## Difficulty Rating

Symbol	Difficulty
	This feature will be quite easy to implement
	This feature is technically complex but within the realm of the developers experience
	This feature is very complex and outside the realm of the developers experience and will, therefore not be implemented
	This feature will take significant time to implement
	This feature is expensive to implement

## Delivery Platform

### Windows PC → 🐎

Jinn's Curse will primarily be played on Windows PCs. Windows is the most prevalent system for PC gaming, therefore be most practical to release for this platform first. Furthermore, the developers of this game are familiar with the Windows system and own windows PC's which will make testing and debugging sympler. However, since Jinn's Curse will be built in Unity it can easily be ported to other systems.

### Mac/Linux → 🐎

Jinn's Curse would ultimately be available on all major PC operating systems so we could reach more of our potential potential customers. Unity makes it easy to save projects for other operating systems so this presents very little risk.

### IOS/Android → 🐎 🐎 🕒 🍞

Mobile gaming is rapidly growing market worth approximately \$50 billion dollars so it requires serious consideration [1]. Jinn's Curse is well suited to a mobile platform as it is casual, however, releasing on mobile presents its own risks. Although Unity allows any game to be built for IOS or Android, the UI and some key systems would need to be redesigned which would take significant time. Additionally, IOS requires that games be built on an Mac computer which raises costs. However, despite these issues, it may be worth it to build for mobile.

**Steam** → 🐎 ⌚ 🍞

Steam is the way a vast majority of gamers purchase PC games[2]. Releasing on this platform will give us access to the 90 million potential customers [2]. Jinn's Curse will be released through Steam Direct. Steam Direct presents few risks or costs but will take significant time to activate as there is a minimum 1 month waiting period. Additionally, releasing on Steam costs \$100 dollars.

# Development Environment

## Unity →

Jinn's Curse will be developed in Unity because it allows for the game to be built very rapidly. All team members are familiar with Unity which makes everything easier. Unity also provides a number of tools that allow the designer to change game stats without coding. This will allow for more flexibility and make it easier for the designer to balance the game. Additionally, these features will lessen the chance that the designer breaks something critical while trying to modify the code.

The team will use Unity Colab to collaborate as it is easy to use does everything we need for this project. Since the programmer is not familiar with version control using an outside tool such as Git or SVN presents more risks than benefits.

Unity allows us to release the game on multiple platforms which will give us more flexibility to expand our market. Finally, Unity is free to use for any company making less than \$100000 per year which lowers our overall cost.

## Architecture →

Jinn's Curse is implemented based on the MVC (Model View Controller) design pattern. The model consists of a scriptable object that stores spell data and the player manager which manages scripts that store data about the players status, spell ownership, inventory and movement. The view is managed by a group of UI scripts that handle different elements of the UI.

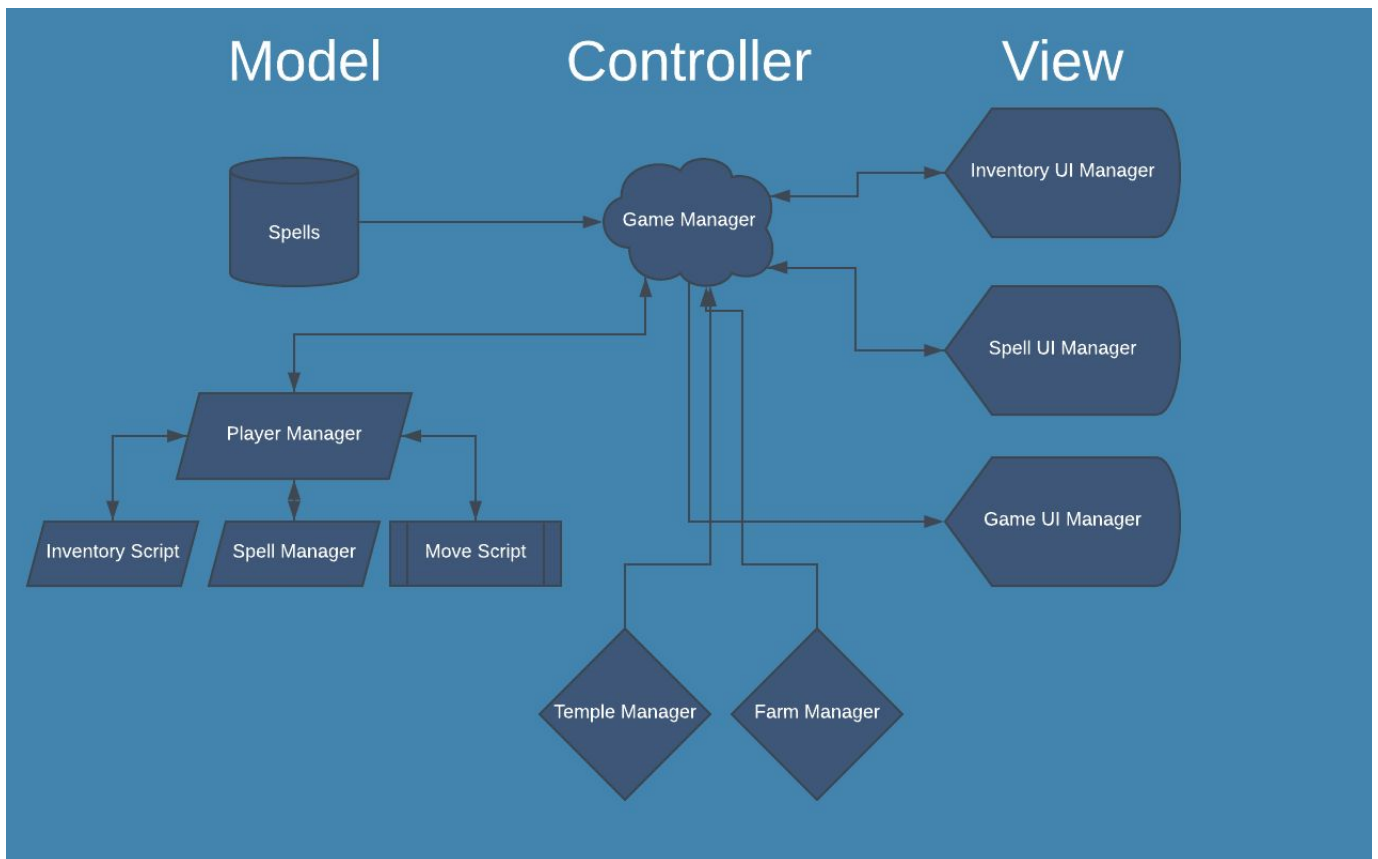
The most important script in the game is the Game Controller, which manages the turn flow and contains references to the current player and the other player. These references allow the game to easily update or read in data. The game controller is responsible for providing the view with any needed data and updating the model based on input from the view. The Game Controller is a singleton and all other scripts can reference it.

While this architecture will ultimately make the game easier to modify and more logical, it takes more time to implement in the short term.



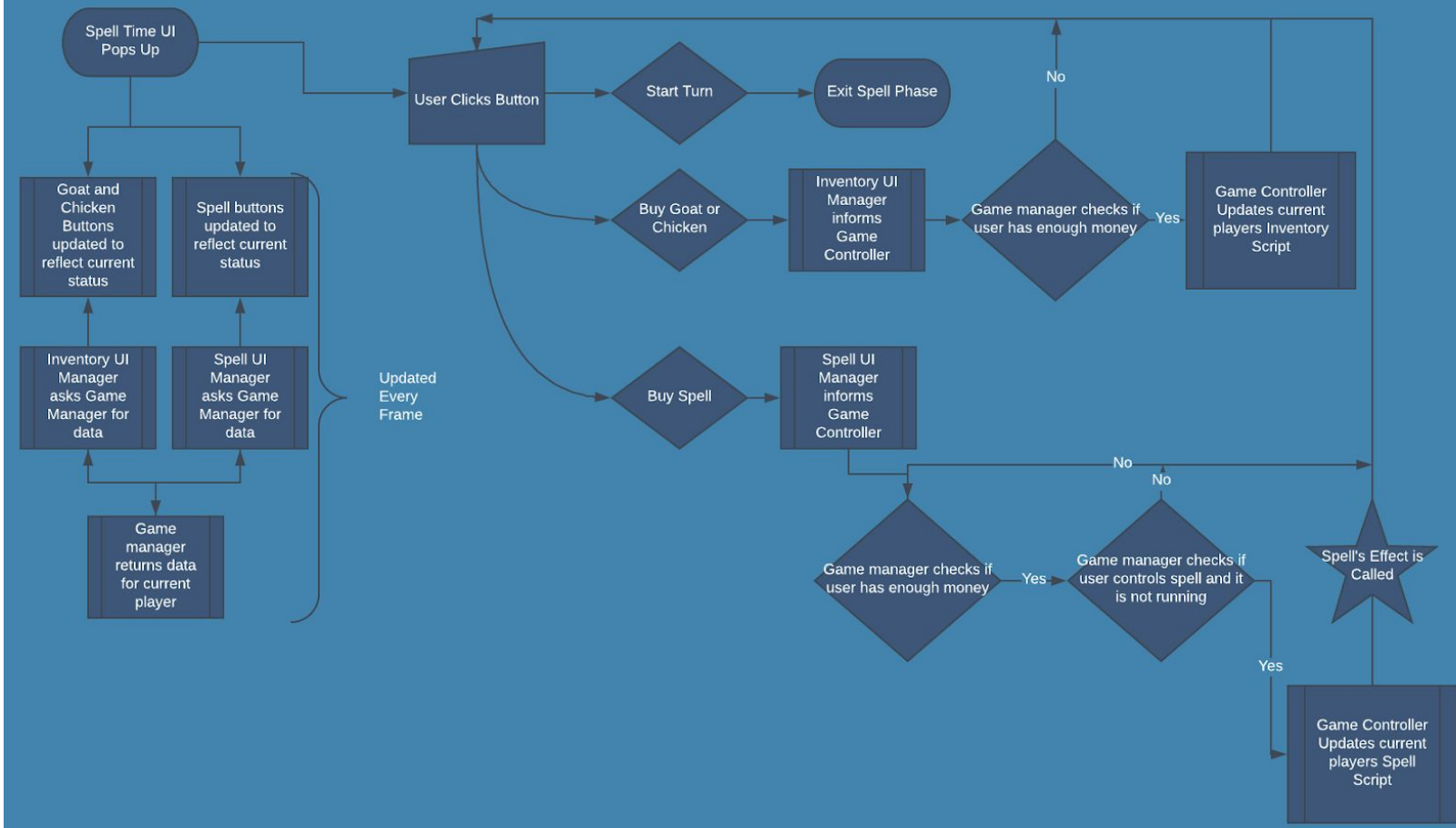
# Logical Flow Diagram

Abstract Model → 

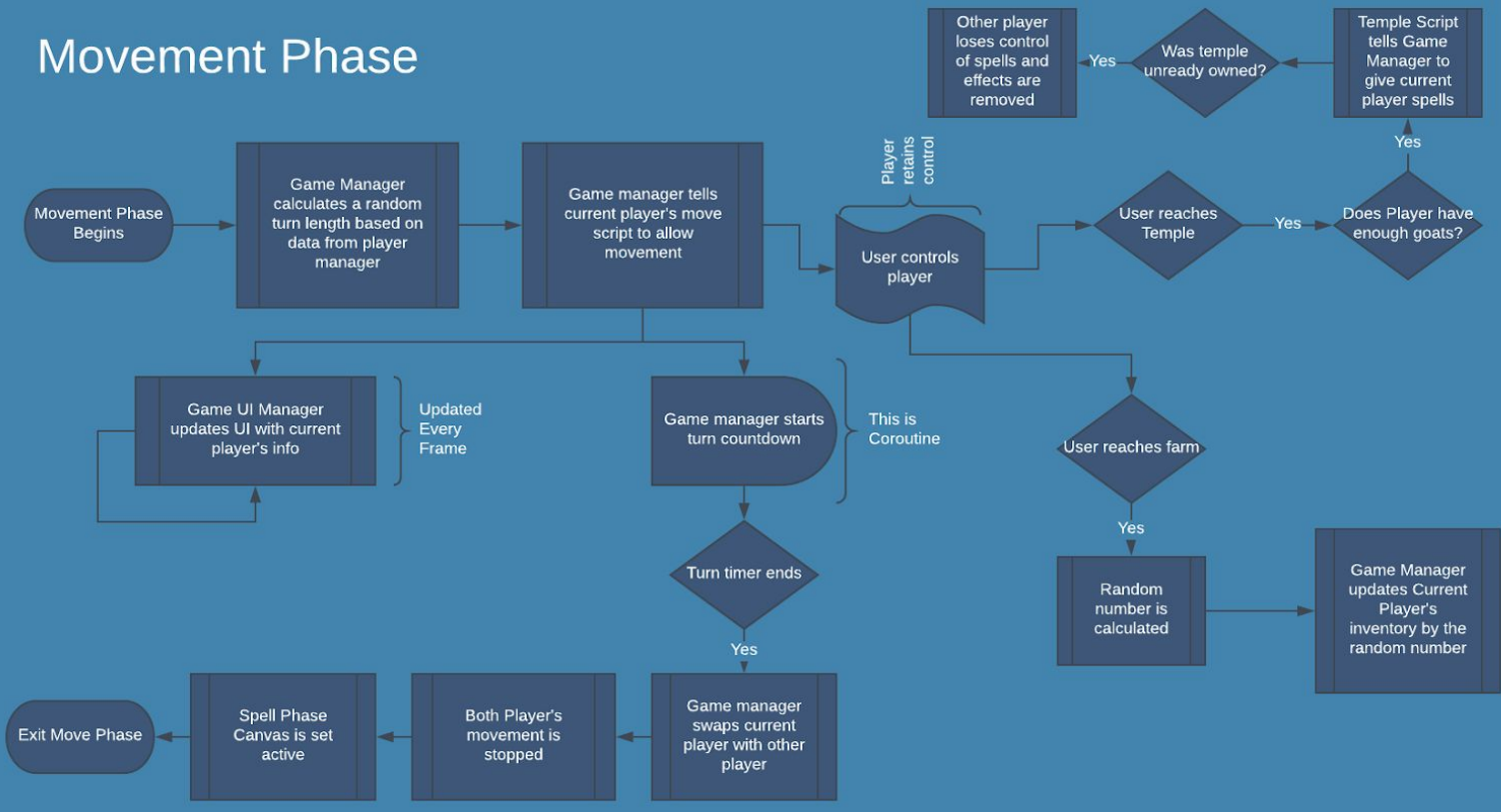


# Turn Flow →

## Spell Phase



# Movement Phase



# Game Mechanics and Systems

## Spell Phase →

The spell phase occurs at the start of every turn. During the spell phase players can not move, instead they are presented with a UI containing buttons representing spells and inventory. The spell phase allows players to manage their resources and plan their turn without worrying about a timer.

## Player given income →

At the start of each turn the player automatically receives income. The amount of income can be set for each player in the inspector. This presents no technical risks.

## Buying Inventory →

While this UI is active players can exchange spice bags for chickens or goats by clicking on buttons. Each button displays the current count of the item and is updated constantly. If a player does not have enough currency to buy a given item the button becomes inactive and is grayed out.

The inventory system was relatively easy to implement and presents few risks. However, it is not very expandible so problems may arise if more types of inventory are added. Additionally, inventory is not currently presentant between runs which could present an issue in the future.

## Casting Spells →

During each spell phase the player can cast one spell by clicking on the corresponding button. The buttons are only active if a player is allowed to cast a given spell. The following conditions must be true for a spell to be cast:

- 1) The player must control the spell
  - a) Players gain control of spells by capturing temples during their turn
- 2) The player must have enough chickens to pay the casting cost
  - a) Players buy chickens with bags of spices
- 3) The spell must not already be running
  - a) Buttons for spells that are currently active turn green and become inactive

Once cast each spell has a random chance of succeeding which can be set in the inspector by the designer.

Casting spells presents few risks but is time consuming. While most elements of the game including the names and costs of spells can be modified in the inspector the effects of spells must be defined in code which is time consuming. If spells change in the future time will be needed to update everything.

Spells present an additional risk because they are used in multiple places. When the designer updates the name of a spell or the order of the spell list he must be careful to also update the onclick effect of the corresponding button and the name in spell list held by each temple.

## Winning →

A player wins when they cast the divinity spell. In order to unlock the divinity spell the player must unlock and cast all other spells. Once unlocked the divinity spell can be cast as long as the player has enough resources. The amount of resources required can be entered in the inspector. If the player succeeds in casting the spell the game ends and they are taken to a win screen.

## Exiting Spell Mode →

Players can start the move phase of their turn at any time by clicking a start button. This presents no technical risks.

## Move Phase →

During the move phase players can explore the world. They can attempt to capture temples or gain spices from farms. Any spells that were cast that affect the current players movement take effect during the move phase. The time allowed for the move phase is based on a random number within a range plus a booster which can be set by spells. The player uses keyboard input to move and their speed is based on a set rate plus or minus any spell effects.

## Capture a Temple →

When a player reaches a temple they automatically attempt to capture it. If they have sufficient goats in their inventory they sacrifice one to gain control. The player can now cast any spells controlled by the temple during their next spell mode. If the other player previously controlled the temple they now lose control. The other player can no longer

cast spells that are tied to the temple, and the effects of any spell they had previously cast are removed.

Capturing a temple presents very little technical risk. One problem could arise when removing the effects of a spell. Since the magnitude of spells such as exhaustion are set by the designer removing their effects could prove problematic since the program does not know what the original magnitude was. This issue could be addressed by using a modifier instead of setting speed and other attributes directly.

### Harvest a Farm →

When a player reaches a farm they can harvest it to gain valuable spices. The amount of spices they receive is a random number chosen from a range. Players do not control farms and any player can visit any farm. Once a farm has been harvested it is burned out and can not be harvested again during the same turn. Farms present very little technical risk.

### Exiting Move Phase →

When the players turn ends the game manager passes control to the other player. The Spell Phase UI is then activated and the other players spell phase begins.

## Art Pipeline →

The art will be created in Clip studio paint 2019 and saved as .png files following the convention:

**[AssetName]\_[AssetVersionID].png**

. All art will have transparent backgrounds. The art sizes will be as follows:

1. **Characters:** 200 X 200
2. **Inventory Items:** 200 X 200
3. **Spell Icons:** 100 X 100
4. **Background:** 2580 x 2542

After creating the art the artist will place it in the shared Art folder on google drive (Production Project 1 > Art).

The team will then approve the art after which the programmer will place it into the art folder in Unity. The team will communicate their approval over Google Hangouts chat. The programmer or designer can then implement the art into the game.



## Design Pipeline →

Jinn's Curse was architected with Designers in mind. Nearly everything in the game can be modified without touching the code. The following information is available for the designer to modify in the inspector.

### Global Data

#### Spells Scriptable Object

The spells object controls the spells available and their costs. While the designer can modify any spell data from the inspector it is important that they keep in mind that the name of the spell serves as a unique identifier if updated here it must be updated in the temple script. The designer must also keep in mind that the spells are read into their buttons in the same order as this list. If the designer wishes to add a spell they must also add a corresponding button. The designer can modify the following variables for each spell:

- Name
- Cost
- Level
- Description

#### Spell Buttons

Each spell button corresponds to one item in the spell list. It's on click event triggers the casting of a spell. The designer can change the effect of any spell button by modifying its onclick event in the inspector. Many spell functions also take an argument of amount that determines the magnitude of the spell. The designer can modify the magnitudes in the onclick event for each button.

## Temple Script

Each temple script contains a list of spells that it controls. The designer can modify the number and type of spells controlled by entering them into the list that appears in the inspector for this script. Keep in mind the capitalization and spelling must be exact.

## Farm Script

The designer can modify the minimum and maximum payout for each farm.

## Player Data

The designer can modify a great deal of statistics for each individual player. There are a few key scripts that handle this data.

## Player Manager

The player manager handles data about the players turn. The players turn length is calculated using the following formula:  $\text{Random.Range}(\text{min turn length}, \text{max turn length}) + \text{booster} - \text{decrementer}$ . The designer can modify the following values:

- Minimum Turn Length
- Maximum Turn Length
- Booster
- Decrementer
- Gold per turn
- Player Colors
- Player Icon
- Player Token

## Inventory Script

This script manages the player's inventory of items such as spices, chickens and goats. The designer can modify the following values:

- Starting Spice Sacks (internally called gold)
- Starting Chickens
- Starting Goats
- Income per Turn

### **Spell Manager**

In the spell manager the designer can set spells to be owned as start for a given player. Please keep in mind that the size of this list must always be equal to the number of spells. Also keep in mind that it is not advisable to set spells to be running at start as they will appear to be running but not actually have any effects.

### **Other Features**

If the designer would like to modify any other element of game play or create more spell they will contact the programmer or bring it up at a meeting.

# Milestone Updates

## **Sprint 1:**

Helped the build the physical prototype. No work was completed on electronic prototype

## **Sprint 2:**

Most of the core features of the game are finished. Next sprint will consist of finishing a few features and refactoring code, UI polish, and balance

## **Sprint 3:**

The game has been polished. The remaining core features were implemented and an minimap was added. The UI was polished. The code was refactored and organized.

## Sources:

### Clock Icon

[https://play.google.com/store/apps/details?id=com.google.android.deskclock&hl=en\\_US](https://play.google.com/store/apps/details?id=com.google.android.deskclock&hl=en_US)

### Information

[1]“The \$50B Mobile Gaming Industry: Statistics, Revenue [Infographic],” *Mediakix / Influencer Marketing Agency*, 19-Jun-2018. [Online]. Available: <http://mediakix.com/2018/03/mobile-gaming-industry-statistics-market-revenue/#gs.5NvgsXOt>. [Accessed: 28-Jan-2019].

[2]C. Smith, “34 Interesting Steam Statistics | By the Numbers,” DMR, 21-Jan-2019. [Online]. Available: <https://expandedramblings.com/index.php/steam-statistics/>. [Accessed: 28-Jan-2019].