Team Members

First Name	Last Name	ONID	email
James ‘Cam’	Abreu	abreuj	abreuj@oregonstate.edu
Philip ‘Phil’	Jarrett	jarrettp	jarrettp@oregonstate.edu
Christopher	Bugsch	bugschc	bugschc@oregonstate.edu

Introduction

The Caelum team will be creating a custom Mario Party clone using the MonoGame game engine. The game will be called MonoGame Party 2018 or MGP18 for short. The game will follow the traditional play of Mario Party where multiple players take turns moving around a game board. Players must compete in minigames to collect “coins” in order to purchase “stars” located around the game board. Monogame Party will be cross platform on Windows, Mac, and Linux machines using openGL as the graphics rendering system.

Our plan to begin outlines a single player experience in which the user will compete against three other A.I. players. We will start with a tight scope by limiting graphics to basic shapes and focusing more on the game engine and gameplay scripting. Then, as development unfolds we can shift from mechanics to aesthetics and apply more features and tweaks to the game such as more ‘boards’, ‘minigames’, ‘graphics’, ‘music’, ‘sound effects’, as well as an additional feature to allow two local players to play simultaneously. Features will depend on the development speed, which gives us confidence in reaching a ‘working game’ by the end of the 8 weeks.

[top]

User Perspective

MGP18 is being modeled most closely after the second game in the series (Mario Party 2). The user will experience a board game-style video game containing challenging and fun minigames. Mario Party 2 along with the other N64 Mario Party Series games are simple enough ideas for our lightweight and short development scope. In addition, the games set many of the standards in modern casual games and should be familiar with most users. It also removed what was seen by many as Mario Party 1's biggest blunders by taking away the games that relied heavily on rotating the control stick, which caused several problems for Nintendo (Associated Press 1).

Some of the minigames will have elements of both luck and skill, while others will have a higher element of randomness. We all love the balance of 'chaos' and skill that these games had to offer. The user will have the ability to select the difficulty of the A.I. opponents in MGP18 and this will increase the playability of the game once the initial novelty has worn off. Users will be able to select from a handful of different characters. In addition to the A.I. opponents, the user will also have the option of playing against a human controlled character, adding a new element of difficulty to the game.

We have chosen to allow the user to select a lower amount of total game rounds so that they can experience the game more casually. For instance, most Mario Party games have a 15 or 20 round *minimum*, which forces the user to 'bunker down' to play the game for at least 30 minutes. For our game design, the minimum game round amount will be decreased to 7 rounds with additional options for game lengths of 12 and 20 rounds. We feel that this will be a nice design change that fits the more modern on-the-go mentality of this generation of gamers.

[top]

Initial Plan

*Note: The * symbol marks a feature that will be added only after the initial working game is complete*

Game State Engine
There is a single game state engine that is responsible for running the game. This engine will control which states are active which then calls the functions for each active state.

States
States contain a list entities for that specific state.
Splash Screen
• Display graphics and plays music
Opening Dialogue*
• Fun little story intro displayed as text with music
Main Menu
• Create game button • Select map options * (in place of create game button) • About/Credits • Exit
Game Setup
• Choose difficulty * • Choose number of human players (1 or 2) * • Choose number of rounds to play

[top]

• Choose Character • Allow bonus options (eg. extra points for most minigames won)
Map Intro
• Scripted camera movements around map
Board Map
• Zoomed in on current player
Land on Space
• Run that space action • Handle multiple players on same space (graphically)
Round
• Four player turns
Player Turn
• UI Design • Input Management • Confirm player is ready • Roll dice • View Map • Move player
Purchase Star
• Remove x coins from current player • Move star to new position on board • Add star to player's score
Minigames
• Input management • Run functions for minigame
Minigame Results
• Award points

[top]

Game End
• Display game results and winner

Entities
Player
• Human or A.I. • Character • Score
Character
• Name • Possible special trait • Unique Sprite
Score
• Star Count • Gold Count • Hidden minigame win count
Board Space

• Good Space
  ○ Color: Blue
  ○ Add 3 - 15 coins to player
• Bad Space
  ○ Color: Red
  ○ Remove 1 - 10 coins from player
• Star Space
  ○ Has star
  ○ Allow player to buy star
• Duel Space *
  ○ Start minigame duel between current player and random other player

[top]

Button
• Clickable (true/false) • Action function
Star
• Cost • Has Been Bought
Minigame 1

• Round
  ○ List of players
  ○ Current Player
  ○ List of Detonators
  ■ Has been pressed
  ■ Is Dangerous

Camera/Screen Movement
• Zoom in on current player • Allow player to move camera around map to view different areas

Input Manager
• Virtual class that each state will use to determine when and what inputs are accepted.

[top]

Key Mapping System
A class that determines what the default key is for a specific action.
Action Button
• Space Bar
Cancel Button
• Esc
Directional Buttons
• Keyboard Arrows • ‘W’ ‘A’ ‘S’ ‘D’ (if adding 2 player feature) *
Start/Select Button
• Enter/Return
Minigame Extra Buttons
• Button 1: TBD • Button 2: TBD

[top]

Music
Pre-game
• Splash Screen/ Intro • Main Menu • Beginning of Game (loading map) *
During gameplay
• Map 1 • Map 2 * • Pause Screen • Minigame 1 • Additional Minigames *
Post-game
• End Game

Sound Effects
Menus
• Menu move selection, Menu Select (confirm choice) • Individual Character Selection *
Gameplay
• Start Game • Roll Dice • Gaining coins / Removing coins* • Purchasing a star*
Minigames
• Minigame 1 (multiple) • Additional Minigames (multiple) *

[top]

Graphics
Characters
• Character A • Character B • Character C • Character D • Additional Characters *
Menus
• Main Menu Background/Buttons • Character Selection Menu (if 2-player) * • Map Selection Menu/Buttons *
Game Board
• Game Board Background • Game Board “good” Tile • Game Board “bad” Tile • Game Board Decorations (several) *
Minigames
• Minigame Instruction Screen/Buttons • Minigame 1 Background/Objects • Additional minigame Background/Objects * • Minigame Result Screen/Buttons
Other
• Game Result Screen/Buttons • Game Instruction Screen/Buttons • About/Credit Screen *

[top]

Board Designs

Game Board 1

- Single path board with a high good to bad space ratio.
- Simple and easy to play

Game Board 2 *

- More bad spaces to increase difficulty
- More complex path with multiple splits that allow the player to choose a direction

Minigame Designs

Minigame 1: Bowser Blowout style game

Game begins with five detonators. Players take turns choosing a detonator to press. If that detonator blows up the bomb, that player is eliminated from the game. If it does not, the next player goes. If all players take a turn and do not detonate the bomb, the round resets. Each time a player is eliminated, one detonator is removed and a new round begins. The game is over when only one player remains.

- Game of chance
- Can easily add levels of difficulty by giving the A.I. a better chance of choosing a safe detonator
- Dark themed graphics and music style

Optional: Minigame 2 - Honeycomb Havoc style game

The game begins with a row of shapes and a spinning die. The die contains only the numbers 1 and 2. Players take turns selecting how many shapes should be removed from the row by stopping the spinning die on the number they want. The correct number of shapes is then removed and the next player takes their turn. The person who removes the last shape from the row is eliminated from the game. Each time a player is eliminated, the row of shapes is reset and a new round begins. The game is over when only player remains.

- Incorporates some skill and logic with a little bit of chance
- Light themed graphics with a fun music style
- Could expand on original design and say that various ‘shapes’ have points as well.

[top]

Optional: Minigame 3 - Live Action style game
This style minigame would involve the user pressing the left and right arrow keys in an alternating pattern. Doing so would increase the player's win column. The first player to completely fill their win column, wins the minigame. ● Winning is based on skill with chance not being a factor ● All players compete at the same time. This would not be turn based. ● Water themed graphics and music style
Optional: Duel Minigame
This game would be played by two players when a player lands on a "duel" space. The player that lands on the space would first choose another player to duel. Once decided, the minigame would start by displaying both players on the left and right side of the screen. There would be a countdown timer in the center that would start at five seconds and decrease with each second. When the timer reaches zero, both players would hit there action button. The player that hit their action button first is the winner. If a player hits the action button before the timer reaches zero they automatically lose.

Software/Apps Used
Microsoft Visual Studio
MonoGame (C# Game Development Libraries/Environment)
Windows 10 Development Environment
GitHub
Google Docs
Discord
● Separate channels for communicating about different topics/sharing ideas ● Setup Git integration to track and notify teammates of changes/commits ● Audio/Video communication for virtual meetings and screen sharing
Trello
● Post-it like app for creating and maintaining schedules/deadlines

[top]

Team Member Tasks

Christopher Bugsch

Week 3	
Task	Estimated Time (hrs)
Create space entities	4
Work on key map system	1
Work on input manager for mini game 1	2
Work on graphics for minigame 1	5
Begin game setup state	4
Week Total	**16**

Week 4	
Task	Estimated Time (hrs)
Finish game setup state/graphics	6
Work on board game graphics	4
Complete purchase star state	6
Create/Submit Progress Video	0.5
Week Total	**16.5**

[top]

Week 5	
Task	Estimated Time (hrs)
Finish working on main game state	8
Complete main menu state	6
Work on endgame state/graphics	6
Collaboratively Complete Instructions for Mid-Point Project Check	2
Week Total	**20**

Week 6	
Task	Estimated Time (hrs)
Complete mini game results state/graphics	6
Unit testing/Game testing	2
Start duel minigame	5
Create/Submit Progress Video	0.5
Week Total	**17.5**

Week 7	
Task	Estimated Time (hrs)
Finish duel mini game	5
Work on minigame 3	10
Collaboratively Complete Project Poster	3
Create/Submit Progress Video	0.5
Week Total	**18.5**

[top]

Week 8	
Task	Estimated Time (hrs)
Complete minigame 3	10
Collaboratively Complete Final Report	5
Week Total	**15**
Total Estimated Time	**103.5**

[top]

James Abreu

Week 3	
Task	Estimated Time (hrs)
Work on game state engine	8
2 other character entities	2
Create player entity	1
Create buttons entity	1
Start land on space state	5
Week Total	**17**

Week 4	
Task	Estimated Time (hrs)
Finish game state engine	6
Finish land on space state	4
Work on player turn state	6
Create/Submit Progress Video	0.5
Week Total	**16.5**

Week 5	
Task	Estimated Time (hrs)
Finish player turn state	8
Begin Music/SFX	8
Collaboratively Complete Instructions for Mid-Point Project Check	2
Week Total	**18**

[top]

Week 6	
Task	Estimated Time (hrs)
Complete splash state	6
Complete Music/SFX	6
Create/Submit Progress Video	0.5
Complete Camera/Screen Movement	5
Week Total	**17.5**

Week 7	
Task	Estimated Time (hrs)
Work on adding 2 player mode	6
Add buttons to key mapping system for 2 player	1
Opening dialogue state	5
Collaboratively Complete Project Poster	3
Create/Submit Progress Video	0.5
Week Total	**15.5**

Week 8	
Task	Estimated Time (hrs)
Finish 2 player mode	4
Upgrade graphics	6
Collaboratively Complete Final Report	5
Demonstrate Project	4
Week Total	**19**
Total Estimated Time	**103.5**

[top]

Philip Jarrett

Week 3	
Task	Estimated Time (hrs)
Two character entities	4
Score entity	2
Star entity	2
Work on minigame 1	6
Week Total	**14**

Week 4	
Task	Estimated Time (hrs)
Complete player entity	2
Complete map intro state	6
Complete round state	4
Create/Submit Progress Video	0.5
Work on minigame 1	6
Week Total	**18.5**

Week 5	
Task	Estimated Time (hrs)
Complete board map state	6
Collaboratively Complete Instructions for Mid-Point Project Check	4
Prepare mid-point check and submit	0.5
Work on minigame 1 or 2	8

[top]

Week Total	**18.5**

Week 6	
Task	Estimated Time (hrs)
Work on minigame 2	8
Complete game end state	6
Integration testing	4
Create/Submit Progress Video	0.5
Week Total	**18.5**

Week 7	
Task	Estimated Time (hrs)
Work on minigame 2	7
Collaboratively Complete Project Poster	6
Create/Submit Progress Video	0.5
Week Total	**13.5**

Week 8	
Task	Estimated Time (hrs)
Integration testing	4
Work on high priority issues/bugs or extra features	7
Collaboratively Complete Final Report	7
Week Total	**19**
Total Estimated Time	**102**

[top]

Task Delegation

Assignment	Format	Submitted By	Due Date
Project Plan	PDF	ALL	07/05/18
Week 4 Progress Report	Video	ALL	07/16/18
Mid-Point Project Check	ZIP File	Phil	07/23/18
Week 6 Progress Report	Video	ALL	07/30/18
Week 7 Progress Report	Video	ALL	08/06/18
Create Poster	PDF	Chris	08/10/18
Create Final Report	PDF	Chris	08/17/18
Demonstrate Project	ZIP File	James	08/17/18

[top]

Graphical Examples

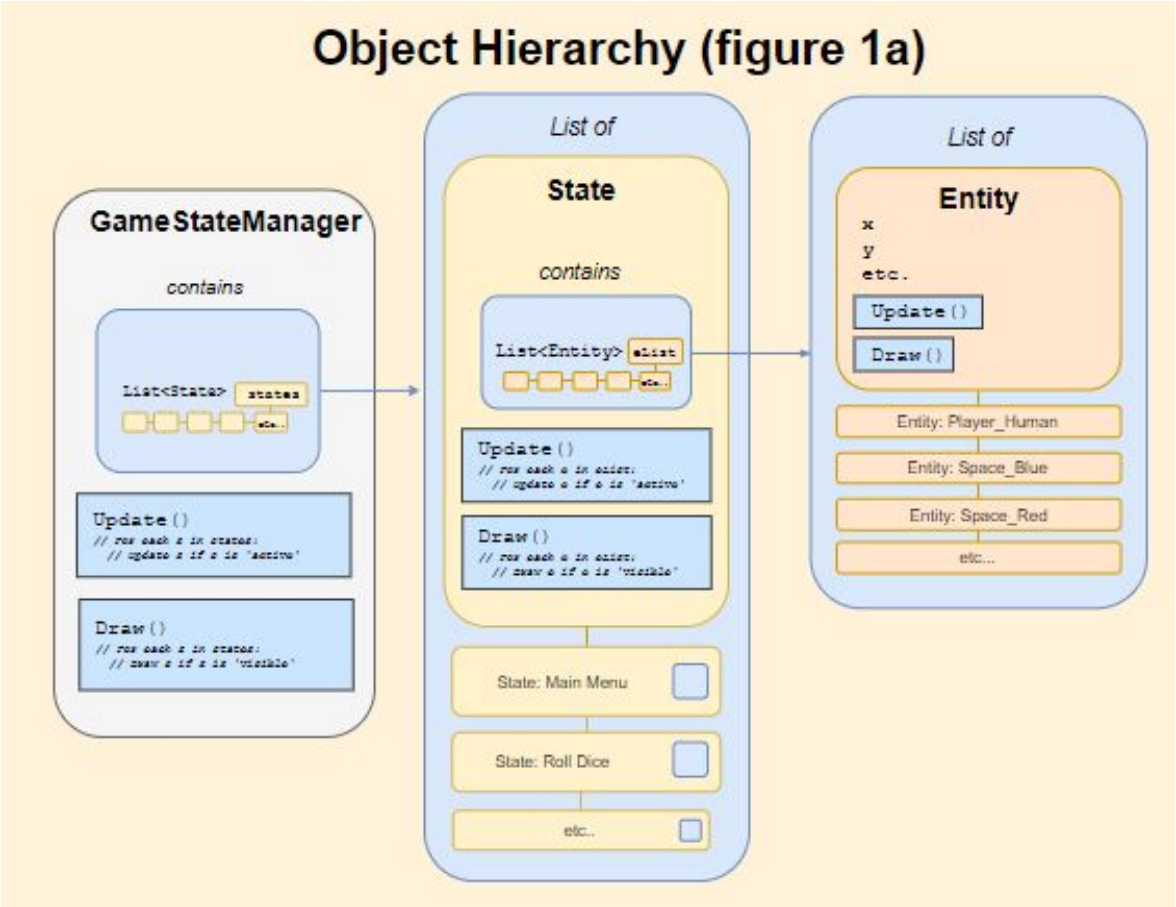

Figure 1a: an Entity and a State are the basic building blocks of our game engine. A player or a space where the players move on the board might be an entity. A State may use or not use Entity instances to keep track of game objects. But a State does not need to have Entity instances to function. For instance, states such as a main menu, a pause menu, etc… might not need to create states and instead draw their own UI and run their own code. Other States such as the main board game, rolling dice, or a minigame might all use instances such as ‘spaces’ or ‘dice block’ or ‘player’ Entity instances. On the broader scope the GameStateManager manages all of these State instances. A GameStateManager will loop through each of its States and run its code as long as the State is ‘active’ or ‘visible’ (see below). There is only one GameStateManager.

[top]

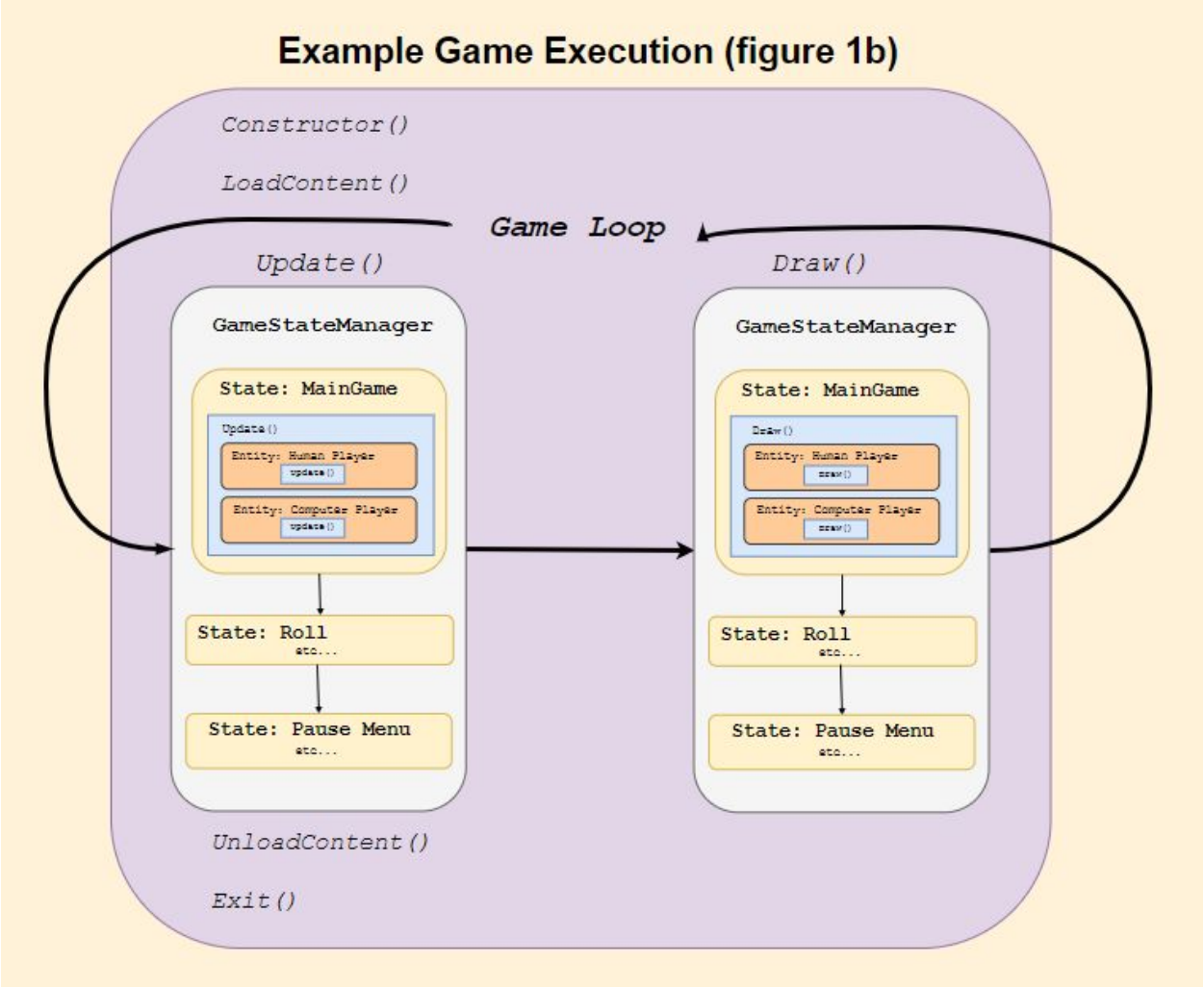

Figure 1b: The main program execution is purple. The Update and Draw functions make up the main game loop. The (one and only) GameStateManager contains all States and thus all Entity instances. The GameStateManager will loop through all of its State Instances and run their respective Update code (if active) and Draw code (if visible). The pattern is replicated in each 'state' in which each State will loop through its list of Entity instances (if it contains any) and run their respective Update code (if active) and Draw code (if visible).

[top]

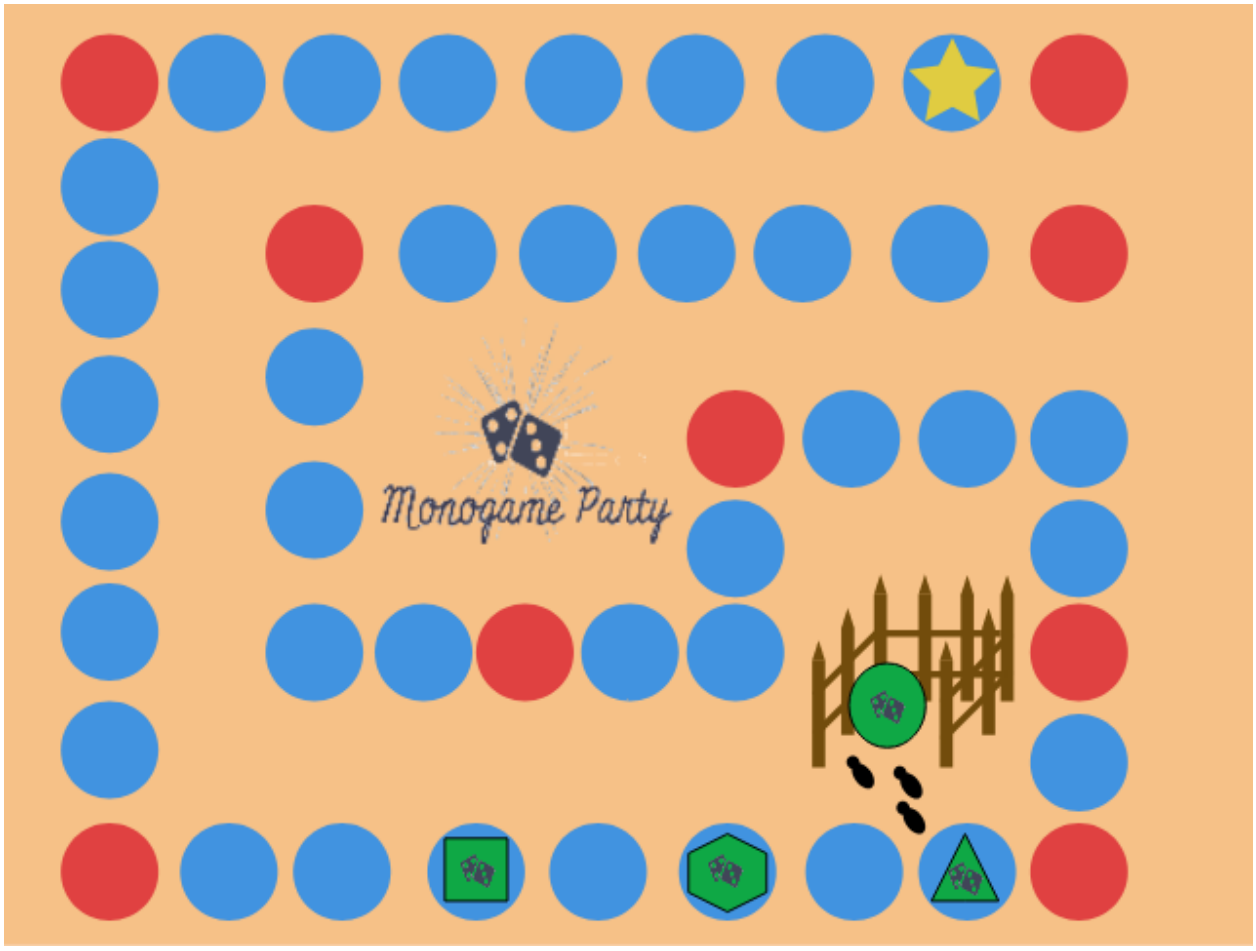

Figure 2: This is an example of our beginning game board. The board consists of the red and blue spaces that the players can move to, the first star space, and the assorted green shapes for player pieces. We will be using a basic design like this in the beginning so we can focus on coding the game mechanics. After we have implemented all of the features in the game we planned to, as time allows, we will upgrade these graphics to more detailed ones.

[top]

Conclusion

By the end of week 6, we hope to have a fully functional game that a single player can play from start to finish created with the Monogame platform. The entire game should take us a minimum of 300 hours to complete. We have chosen to aim for an early deadline to give us time to handle unexpected bugs and delays. If the additional time is not needed, we have planned to add extra features that includes additional minigames and the option to have two human players, as well as upgrade our original graphics to more detailed custom made graphics.

Throughout the process of creating this game, we will gain valuable experience with working in a team, utilizing GitHub to maintain version control and working with a graphical user interface.

We hope you enjoy team Caelum's Monogame Party 2018!

- James 'Cam' Abreu, Christopher Bugsch, & Phil Jarrett

[top]

References

[1] Associated Press. “Nintendo Agrees to Provide Protective Gloves to 'Mario Party' Game Owners.” Los Angeles Times, Los Angeles Times, 9 Mar. 2000, articles.latimes.com/2000/mar/09/business/fi-6902.

[top]