Super Card Bros: Ultimate

Final Report
CS 4361.001

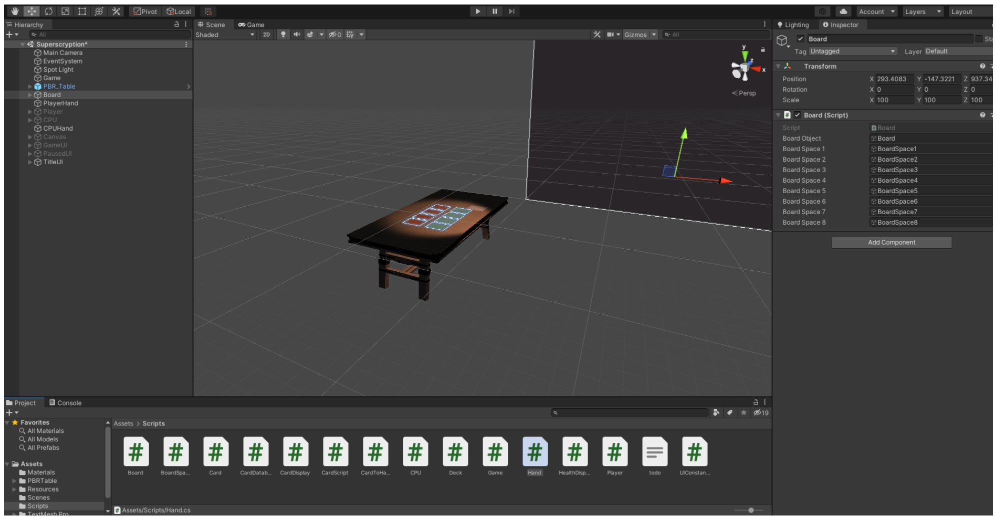
Group Contribution:

Marco Ghercious (25%)
Caden Popps(25%)
Hitesh Sah(25%)
Aditi Prabhu(25%)

Date: 05/05/2022

Project Description

Supercard bros is a fun card battle game inspired by Hearthstone, Inscryption, Smash Bros Ultimate and Super Auto Pets. Supercard Bros is a three-dimensional card game where the player forms a deck of a magical card and plays against a computer CPU. The goal of this game is to draw the card, play your turn, pass it to the computer then defend your card and win it. The view for the game is set in the dark room and the camera is pointed toward the table where the game is being played. Characters displayed on the Cards are adopted from the Super Smash Bros game.



Board
Tag Untagged Layer Default

Transform
Position X 293.4083 Y -147.3221 Z 937.34
Rotation X 0 Y 0 Z 0
Scale X 100 Y 100 Z 100

Board (Script)
Script Board
Board Object Board
Board Space 1 BoardSpace1
Board Space 2 BoardSpace2
Board Space 3 BoardSpace3
Board Space 4 BoardSpace4
Board Space 5 BoardSpace5
Board Space 6 BoardSpace6
Board Space 7 BoardSpace7
Board Space 8 BoardSpace8

Fig 1: View of the Game

There are many rules and features to play this game. The features of the game are health system, attack system, Mana, card rarity and the card probability. Rules for playing this game is that the player chooses cards from their deck to place on the game board. A player can then place up to four cards inline on their side of the board. There are two phases during this game, attack phase and defend phase. During the attack phase, the player attacking may first place their cards and then allow the defender to react.

The defense phase allows the defending player to react to the cards the attacker has on the game board.Once the defender is finished with their move the cards on the attacking side of the board deal damage in a forward manner and apply any bonuses detailed on the card. A player can only take damage during their defense phase in which the attacker's card is not blocked on the board.The damage done is equal to the damage value a card possesses.Damage can be blocked by the defender by putting a card in front of an attacker. Blocking an attacker's card will result in the card blocking losing health equal to the attacking card's damage value. Each card in play will have a damage value and a health value. Additionally, a higher rarity card may have a unique attribute associated with it that will be detailed on the card. A player wins the game when the opponent's health is reduced to zero. The screenshot shown below will help for a better understanding of the game mechanism about how it works and how it is being played against computers.

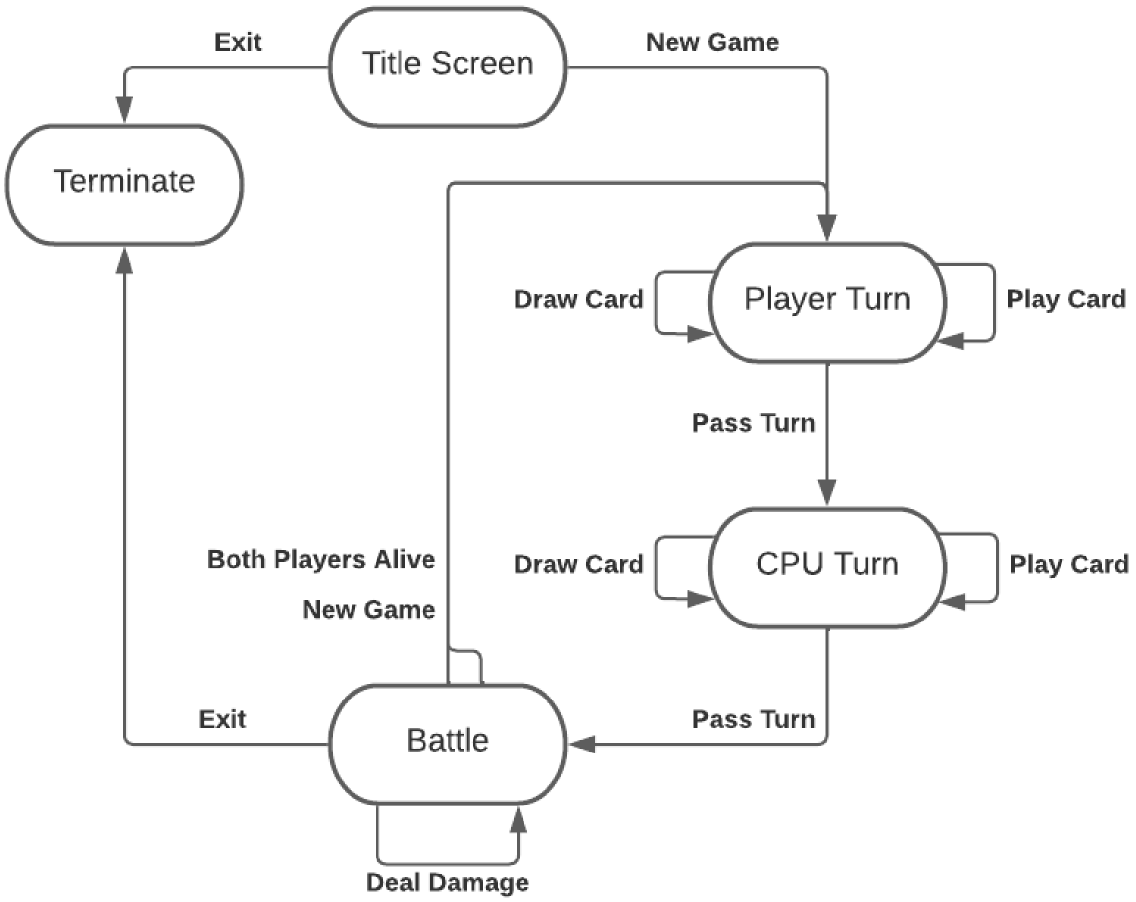


Fig 2: FSM of Game

Tools and Software Used

We have used Unity 3D for the game engine and C# platform for implementing the code. Unity has good support for creating the game in a 3D environment. It is easy to edit the code and better to understand as well.Unity operates in conjunction with the object-oriented scripting language C# to create custom actions and interactions within the game. Unity 3D has a variety of professional tools programmers.Unity offers a workspace that blends artist-friendly tools with a component-driven design that makes game creation a breeze. It has introduced a more user friendly approach for writing the code and for the visual scripting. Unity employs a component-based approach to game development that revolves around prefabs. Prefabs allow game designers to construct items and environments more efficiently and quickly.

We also used Github for the collaboration and for the file management of our project. While using Github, we faced some technical issues like creating the git repository for the project. Because of the large file we were having issues with synching the Unity scene.

Design and Implementation

We proposed our design and feature ahead of its implementation and then later we decided what features to implement in our game so that it can be more appealing and fun to play. Saying that, we designed our game with C#, where we introduced a variety of features in it. The screenshot shown below is the home screen of our game. As we can see, there is a new game button which starts the new game, control button which gives the various controls of the game like select card, unselect card, draw new card, select left, select right, show and hide hands and end the turn of a player which is shown in the next screenshot. Quit button simply quit the game.

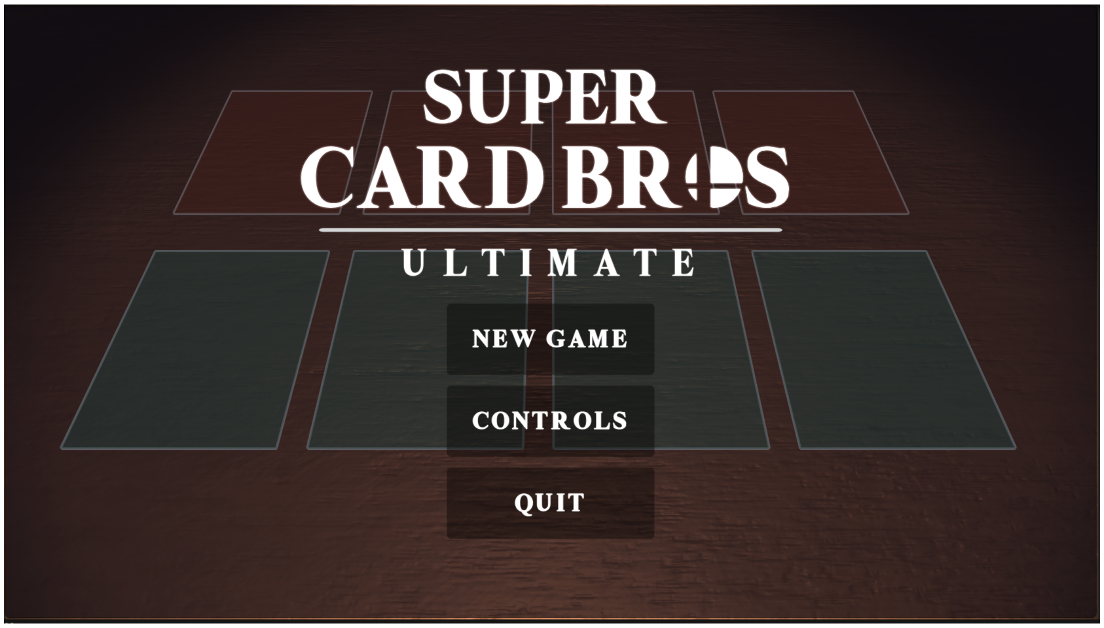
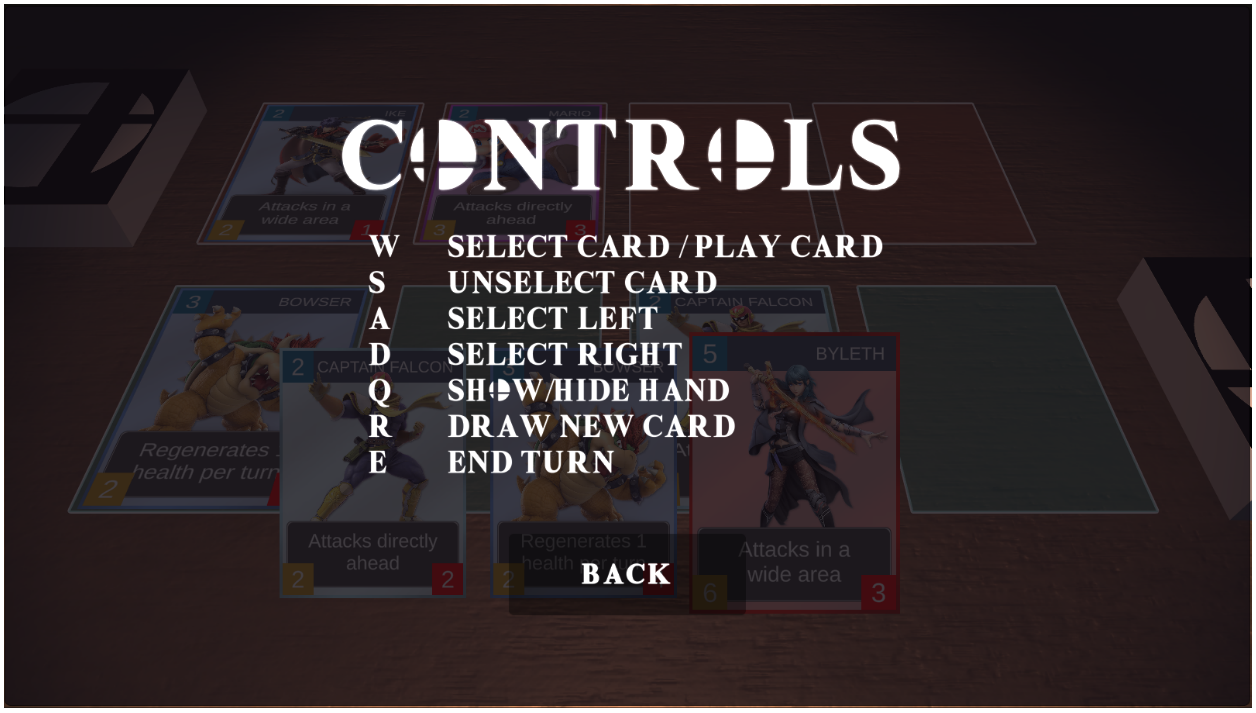


Fig 3: Home screen of the game

In the control section, we have multiple sub buttons and each has a different function. Select button select the card of our own choice, draw new card button draws the card on the table from the new deck. Select left chooses the card that is on the left side of the player and select right button chooses the card that is on the left of the player.


CONTROLS

W SELECT CARD / PLAY CARD
S UNSELECT CARD
A SELECT LEFT
D SELECT RIGHT
Q SHOW/HIDE HAND
R DRAW NEW CARD
E END TURN

Fig 4: control menu of the Game

Furthermore, the screenshot shown below shows the attack and defense phase. Where we can see that the player is attacking and based on our attack and defense our health is calculated. Also we earn or lose points based on our successive attack and defense.

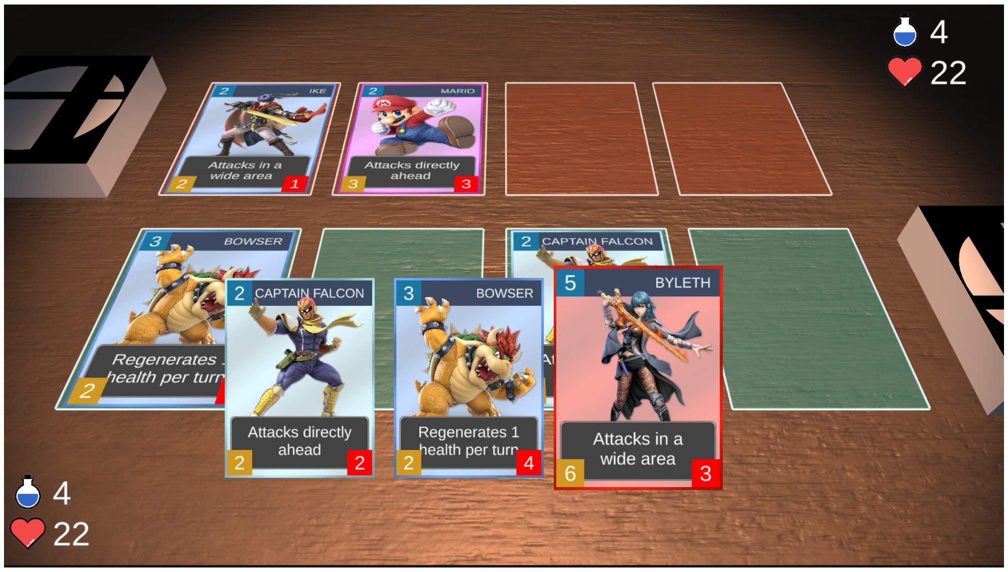
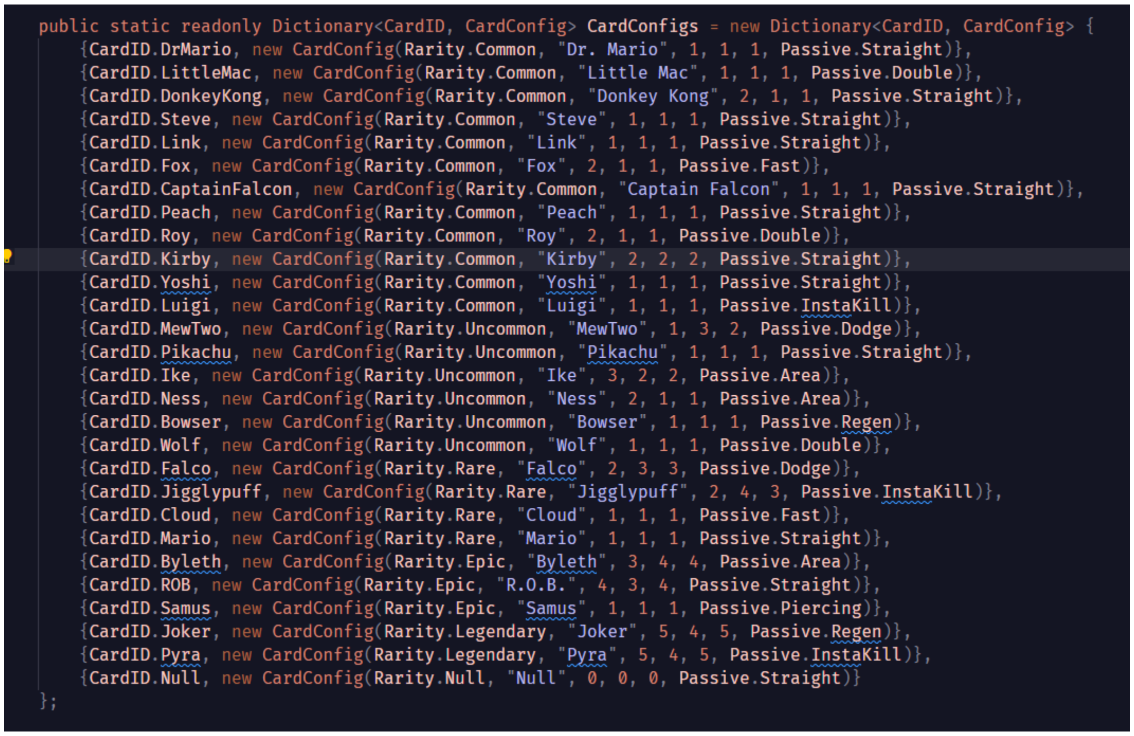


Fig 5: Game play

Once, we designed what we wanted, then we started implementing all the designs that we had proposed. In order to do that we wrote 100s of lines of code for each functionality and setting of the game. We wrote the script of C# for the card generation, attack system, defense system, health, card drawn, passing turn, card selection, end menu, select left setting, select right setting etc. Unity makes it a lot easier for all the above tasks. Its built-in package helps us to set up the project. Some of the code section that use while building the game is shown below: In fig 6, it shows the code for the card generation that is written in C#. This code block is simply generating the card used in game inline and is then saved to the dictionary. Once the card generation is saved then it is being used to instantiate a new card at the runtime with proper properties of the card.



```
public static readonly Dictionary<CardID, CardConfig> CardConfigs = new Dictionary<CardID, CardConfig> {
    {CardID.DrMario, new CardConfig(Rarity.Common, "Dr. Mario", 1, 1, 1, Passive.Straight)},
    {CardID.LittleMac, new CardConfig(Rarity.Common, "Little Mac", 1, 1, 1, Passive.Double)},
    {CardID.DonkeyKong, new CardConfig(Rarity.Common, "Donkey Kong", 2, 1, 1, Passive.Straight)},
    {CardID.Steve, new CardConfig(Rarity.Common, "Steve", 1, 1, 1, Passive.Straight)},
    {CardID.Link, new CardConfig(Rarity.Common, "Link", 1, 1, 1, Passive.Straight)},
    {CardID.Fox, new CardConfig(Rarity.Common, "Fox", 2, 1, 1, Passive.Fast)},
    {CardID.CaptainFalcon, new CardConfig(Rarity.Common, "Captain Falcon", 1, 1, 1, Passive.Straight)},
    {CardID.Peach, new CardConfig(Rarity.Common, "Peach", 1, 1, 1, Passive.Straight)},
    {CardID.Roy, new CardConfig(Rarity.Common, "Roy", 2, 1, 1, Passive.Double)},
    {CardID.Kirby, new CardConfig(Rarity.Common, "Kirby", 2, 2, 2, Passive.Straight)},
    {CardID.Yoshi, new CardConfig(Rarity.Common, "Yoshi", 1, 1, 1, Passive.Straight)},
    {CardID.Luigi, new CardConfig(Rarity.Common, "Luigi", 1, 1, 1, Passive.InstaKill)},
    {CardID.MewTwo, new CardConfig(Rarity.Uncommon, "MewTwo", 1, 3, 2, Passive.Dodge)},
    {CardID.Pikachu, new CardConfig(Rarity.Uncommon, "Pikachu", 1, 1, 1, Passive.Straight)},
    {CardID.Ike, new CardConfig(Rarity.Uncommon, "Ike", 3, 2, 2, Passive.Area)},
    {CardID.Ness, new CardConfig(Rarity.Uncommon, "Ness", 2, 1, 1, Passive.Area)},
    {CardID.Bowser, new CardConfig(Rarity.Uncommon, "Bowser", 1, 1, 1, Passive.Regen)},
    {CardID.Wolf, new CardConfig(Rarity.Uncommon, "Wolf", 1, 1, 1, Passive.Double)},
    {CardID.Falco, new CardConfig(Rarity.Rare, "Falco", 2, 3, 3, Passive.Dodge)},
    {CardID.Jigglypuff, new CardConfig(Rarity.Rare, "Jigglypuff", 2, 4, 3, Passive.InstaKill)},
    {CardID.Cloud, new CardConfig(Rarity.Rare, "Cloud", 1, 1, 1, Passive.Fast)},
    {CardID.Mario, new CardConfig(Rarity.Rare, "Mario", 1, 1, 1, Passive.Straight)},
    {CardID.Byleth, new CardConfig(Rarity.Epic, "Byleth", 3, 4, 4, Passive.Area)},
    {CardID.ROB, new CardConfig(Rarity.Epic, "R.O.B.", 4, 3, 4, Passive.Straight)},
    {CardID.Samus, new CardConfig(Rarity.Epic, "Samus", 1, 1, 1, Passive.Piercing)},
    {CardID.Joker, new CardConfig(Rarity.Legendary, "Joker", 5, 4, 5, Passive.Regen)},
    {CardID.Pyra, new CardConfig(Rarity.Legendary, "Pyra", 5, 4, 5, Passive.InstaKill)},
    {CardID.Null, new CardConfig(Rarity.Null, "Null", 0, 0, 0, Passive.Straight)}
};
```

Fig 6: code block for the card generation.

Lessons Learned

Developing any game is not easy, but with Unity 3D game engine we can say that it makes our task easier. While developing this game we learned so much about Unity itself. We are confident to say that we can create a new game with varieties of cool features easily, given the vast array of features and packages prebuilt into Unity. By analyzing the games that inspired our project, we were able to better understand what happens behind the scenes of many popular games and everything that goes into creating functional and appealing game mechanisms, from the lowest level of coding to the high-level final touch details. In doing this research, we were able to make more informed decisions about the mechanisms of our own game, and how it would need to be designed and implemented to accomplish what we needed it to in the development timeframe that we were given. As a result, learning to effectively research existing games and frameworks was vital to making our game feasible before a single line of code was even written.

Once development began, the lessons learned became very technical in nature. We faced a number of technical issues between coding, setting up a GitHub repository, handling game properties and game objects, and threading within the game execution. Collaborating through GitHub was initially difficult due to the large size and quantity of files that needed to be uploaded to the repository. After most of the files had been imported, it was still a challenge to organize and navigate through the many directories that Unity automatically creates for every project, so we had to learn more about the file structure of the project. During the coding stages, one challenge was creating draggable properties and game objects in a way that was consistent across executions, but we were eventually able to accomplish this through using the System.Serializable attribute on the desired objects. Another development challenge was the use of threading / coroutines in the runtime of the game. We had to create and run threads correctly in order for the game to perform basic functions such as starting and stopping, alternating turn order, and creating new games.

All in all, many technical and non-technical lessons were learned through this project, and they will definitely be applicable to our future projects and endeavors, as they covered everything from the pre-design process, to frontend graphics and user experience, to backend procedures and low-level processes.

Future Plans

In its current state this game project is full of potential, and there are many different directions we can go from here. Related to gameplay, there are quite a few possibilities for new features and functions that can be added. The deck can be expanded to include more cards with new characters, each having unique mana, health, attack, and rarity values to increase the gameplay possibilities. To add to the visual experience, more animations and digital effects can be added, such as cutscene-type character fighting animations so that the players can visually see which cards are taking and inflicting damage, as well as make gameplay that much more entertaining. The current CPU player can perhaps be adjusted to have varying difficulty levels that the human player can choose from prior to the game beginning. A multiplayer feature can also be added where instead of playing against a computer, players can play against each other, which would then enable tournament-style games that many current Super Smash players are already accustomed to. Finally, a version of this game could potentially be released commercially on Steam or other gaming services, however to avoid copyright strikes most aspects of the game would need to be rebranded to have original, non-copyrighted characters on the cards instead of the licensed Nintendo imagery it currently contains. There are many possibilities for this game moving forward, and this is really only scratching the surface.

Conclusion

Our group is very proud of the finished project we have worked so hard on this semester. With an open ended assignment we were excited to build our game from the ground up with full creative freedom. We decided to develop a card game in a 3D scene because we could create our game logic in reference to popular games we play. As development began the path to a completed game became more challenging than expected due to our group lacking unity experience and general issues with software. Over the development cycle we wrote and rewrote many scripts as we learned more about unity development. Once all backend development was completed we were able to introduce some of our favorite video game characters into our game to add the level of polish we were expecting. We enjoyed game development so much that we have tentative plans to continue work and a possible release on a game marketplace.

References

Petty, Josh. “What Is Unity 3D & What Is It Used for?” *Concept Art Empire*, 14 Mar. 2019, https://conceptartempire.com/what-is-unity/.

SmashWiki. “Super Smash Bros.. Ultimate.” *SmashWiki*, SmashWiki, 24 Apr. 2022, https://www.ssbwiki.com/Super_Smash_Bros._Ultimate.

“Unity Forum.” *Unity Forum*, https://forum.unity.com/.