Super Smash Bros.: All my Friends are Dead!: Programmer's Guide

Alisuag, Cyril Ian | Hernandez, Neil Jherome | Sandejas, Paolo

AC-M1

About the Game

"Super Smash Bros.: All my Friends are Dead!" is, game inspired from "Smash Ultimate" of Nintendo, is a multiplayer fighting game. The goal of each player is to knock out their opponent out of the arena. The game can be played by two players.

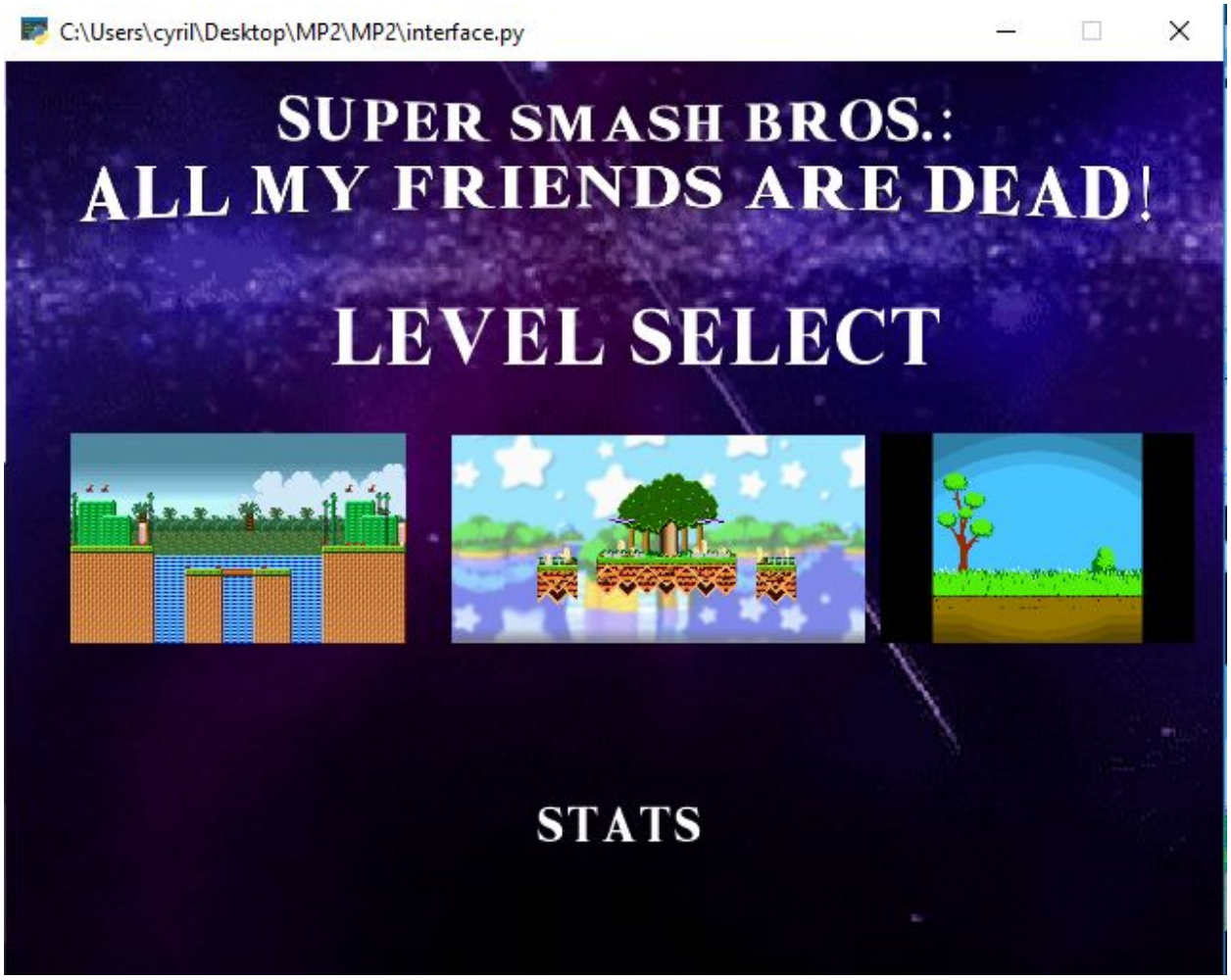

Figure 1. Welcome screen of the game.

Mechanics

How to start:

The players can start the game by selecting the level that they want. At the start of the game, the player can look at their statistics on how many wins and the accumulated points each player has.

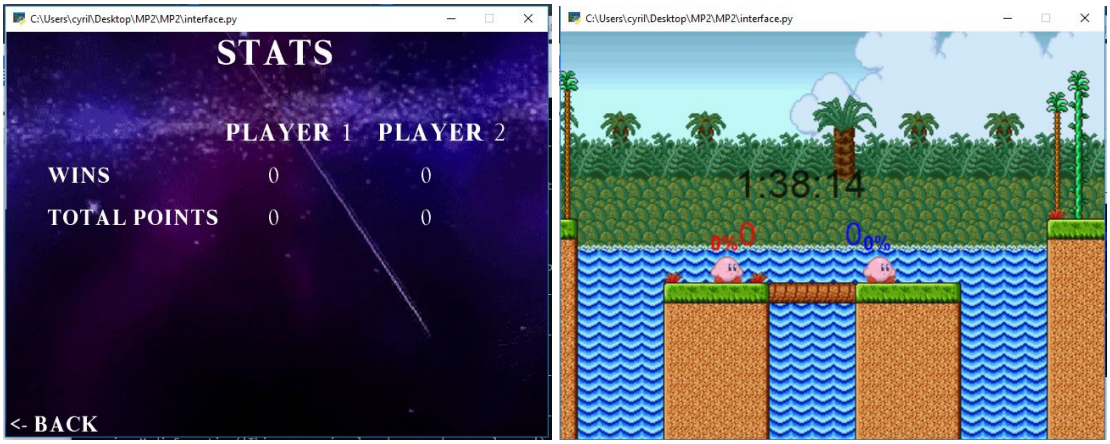

Figure 2. Starting states of the game.

How to play:

Once the players were able to select the level that they want, the game will go to the arena, and the characters will spawn right at the middle of the arena. Within two minutes, players who has the most points wins the game.

Controls:

Player 1: W, A,S,D for movement, G for jump, and H for attack, J for special

Player 2: Arrow keys for movement, 1 for jump, and 2 for attack, 3 for special

Double jump - [up] while in air - the players can only double jump once after touching the land again)

Attack Controls:

Normal Attack - [attack] while in ground

Forward Smash - [left/right] + [attack]

Upward Smash - [up]+[attack]

Downward Smash - [down]+[attack]

Forward Aerial Smash - [left/right]+[attack] while in air

Upward Aerial Smash - [up] + [attack] while in air

Downward Aerial Smash - [down] + [attack] while in air

Forward Special - [left/right] + [special]

Upward Special - [up] + [special]

Downward Special - [down] + [special]

How to win:

The goal of each player is to knock out their opponent out of the arena by hitting them. Hitting an opponent increases their meter. Each attack has different knockbacks which is amplified by the meter of the enemy. A player scores (KO) whenever he/she knocks out his/her enemy to any side of the edge of the screen.

Figure 3. The game ends. Player 1 wins since he has outscored player 2.

How the game ends:

The game ends when the game runs out of time. At the end, the player who has the better score will win the game. The game will save the points and wins accumulated by each player.

Installation

The game has a variety of integral files in the folder, some of them are the engine, objects, interface, and player, the resources folder, and the stats file. The game can be played by opening the interface.py.

Engine: controls the transitions and which objects to be displayed and updated on screen.

Objects: This module contains the user-defined objects used in the game. The objects used are the following:

Box, which contains the coordinates and dimensions of the boxes used for different purposes in the game such as the hitboxes, platform, and sprites.

Hitbox, which contains the box that causes damage. It contains the properties of the box class that does damage whenever it hits the opponent.

GameTimer, which contains the 2-minute timer restraint per level.

Scores, which outputs the score of each player whenever they successfully knocks out an opponent.

DamageMeter, which outputs the damage meter of the player whenever they are hit by their opponent.
Input, which contains booleans for each player command using the keyboard.
Camera, a camera that follows the two players.
Platform, which contains the boxes that serves as the arena of the players.

Interface: This module contains the set-up of the game itself. This module is to be executed when the game should be started.
Player: This module contains the sprite dictionary from the resources folder. This also contains the player class that have the properties and methods of each characters.
Resources folder: This folder contains the individual pictures of the sprites' movements and attacks.
Stats file: This file contains the current stats of the players such as their total number of points and number of wins.
Other files: Other assets used in the game.

To play the game, the user must first activate the virtual environment using "*source venv/Scripts/activate*" while on the main folder then launch the game using *"python interface.py*".

Modules

1. Interface.py

   This is the file ran to start the game, takes all input, and displays the output. It contains the following functions:

   - Update - It contains all the game tick updates in the game and is ran 120 times every second using the function "*pyglet.clock.schedule_interval(update, 1/120)".* This function activates the update function in the engine.py module to pass the update and updates controls the camera.
   - on_draw - this is a pyglet function to draw sprites and other objects on screen
   - on_mouse_press - a pyglet function that handles the mouse clicks
   - on_key_press - a pyglet function that detects when a keyboard button is pressed
   - on_key_release - a pyglet function that detects when a keyboard button is released

   on_key_press and *on_key_release* controls boolean values in the Input class to control the player

2. Engine.py

   This controls the transitions and which objects to be displayed and updated on screen. Engine.py has an array which contains which objects are to be drawn on the screen and which objects need to be updated. It starts by loading the font, images and sound files needed then loads the data inside the *stats.txt* file. It plays the background music immediately and displays the Main Menu screen where the player can start the game by choosing a level or go to the stats menu to view the stats. Going to a level loads the proper background image, spawns the players, scores, timer, and the platforms and exiting a level will make it go back to the main menu.

3. Objects.py

This includes all the Classes the Engine.py imports (except for the player Class) which are the following:

- Box - It contains variables for the the x and y positions, width and height, and the center coordinates for any box that will be used inside the game. This include player hitboxes, platform boxes, the camera box, etc.
- Hitbox - This is a box that spawns and despawns quickly after a player attacks and detects if the enemy player is inside to apply the hit. If the opponent box is detected inside the hitbox, then it applies the force, adds damage, and sets various variables of the opponent to apply what happens when a player gets hit.
- GameTimer - This is set with an integer which sets how many minutes are there in the round. If the time reaches 0, it zooms out the camera and displays which player won or if it is a draw. It updates the stats file and then tells the engine to go back to the main menu after a certain amount of time.
- Scores - This is displayed right below the timer. Each player has their own score object that they create and is color coded (red: player 1; blue: player 2). If the enemy dies at the edge of the screen, then one point is added to the player's score.
- DamageMeter - This contains a label that follows the player and displays their current damage percent on top of their head that is also color coded the same as the scores.
- Input - Both the player has their own Input class objects. This contains booleans for all the possible buttons the player can control and contains booleans for single tick commands from the controls.
- Camera - The camera class is where the interface.py gets values from for controlling the camera - the x and y position, scaling, and zoom values. It first calculates the position it needs to go, which is based on the leftmost and lowermost player. Then, based on that player's distance from the other player, it calculates the zoom value it needs to let both players be shown inside the camera and be centered. Finally, it applies a smoothing algorithm to smooth the camera.
- Platform - These are boxes that can be instantiated with a custom x and y position and scaling. The player object detects these as platforms and interacts with them accordingly.
- Stage - The stage class contains information about the background image to be used as guides by the engine module.

4. Player.py

This is the biggest chunk of the code which contains the player class which controls everything about the players in game. The player class does the following:

- Load and assign the sprites to be used by the character.
- Apply the sprites needed accordingly
- Control the physics engine of the players and move the player's position each frame
- Make the character interact with the platforms (landing, grabbing, etc.)
- Make the character die and respawn when at the edge of the bounds of the stage
- Detect all inputs from the Input class and detect all possible input combinations for various attacks and movement
- Make the player do various attacks and apply the proper hitbox and sprites

Resources/References

Github website:

https://phzixis.github.io/mp2/

https://github.com/phzixis/mp2

References:

Programming Guide: Pyglet

https://pyglet.readthedocs.io/en/pyglet-1.3-maintenance/programming_guide/gl.html

Camera behavior in pyglet:

https://stackoverflow.com/questions/50899831/camera-behavior-in-pyglet

Fonts in pyglet:

https://pyglet.readthedocs.io/en/pyglet-1.3-maintenance/modules/font.html

Sprites in pyglet

https://pyglet.readthedocs.io/en/pyglet-1.3-maintenance/modules/sprite.html

Resources:

Kirby Sprites

https://www.spriters-resource.com/custom_edited/supersmashbroscustoms/sheet/17186/

Stages

https://www.spriters-resource.com/custom_edited/supersmashbroscustoms/sheet/76394/

https://www.spriters-resource.com/custom_edited/supersmashbroscustoms/sheet/76263/

https://www.spriters-resource.com/custom_edited/supersmashbroscustoms/sheet/104078/

Font

https://fontmeme.com/super-smash-bros-font/

Music: Super Smash Bros. Ultimate Main Theme - Lifelight on Kazoo by Slimecicle

https://www.youtube.com/watch?v=w0T-0Pt6iK8