PyGame Super Smash Bros with Artificial Intelligence

Project Description:

This program will simulate a game of Super Smash Bros against an AI. The player will be able to choose from two different characters and also select the character to go against. I will incorporate features such as hitboxes, knockbacks, and projectile motion to simulate a a version of the game that is as realistic as possible. In addition, I am implementing the use of hitboxes such that different damage will be dealt to different areas of the character. I am also setting a medium and hard mode for the character, which will have a different AI depending on the mode. I will also make it so that characters in the game can grab onto obstacles and hold onto the surfaces to avoid falling down.

Competitive Analysis:

I have only seen one other example online of Super Smash Bros. In the online example, the player could select through various battlefields and various characters to face off against. The game was two-player, with different movements leading to different actions for each character.

However, the online example did not seem to have artificial intelligence implemented

into it, and there were no hitboxes throughout the areas of the character. I also have different modes of difficulty for the artificial intelligence so that the player will have different levels of skill to face off against. The player can first gradually start improving their skills in Super Smash Bros by first using the easier difficulty, and once the easier difficulty becomes to easy for the player, the player can then transition into more difficult modes to face against the AI. This program will better the player’s skills such that the player will become more prepared in the real world whenever he/she is presented with the opportunity to face off a real person.

Structural Plan:

I will utilize Object Oriented Programming. I will have a Character Class with its subclasses being Lucario and Mario. The character will have a range attack, a block function, a close attack function, a draw function, a jump function, a move function, and an \_\_init\_\_ function. The various hitboxes will be \_\_init\_\_. I will also have a Surface Class that shapes all the surfaces in the game (jumping to another surface) that is a “hitbox”, as collisions with it will change the movement of the character. I am considering implementing a Bullet Class that is called whenever the character fires a ranged attack. To implement the whole game, I will have a Menu/Game class that has objects of the other classes in its \_\_init\_\_. It will combine all the classes to run the game. The Menu class has a timerFired, redrawAll, mouse/keyPressed, and an \_\_init\_\_. In my timerFired, I will implement the collisions between the characters with other characters, characters with other objects, and bullets with characters/objects.

In a separate class/program, I will implement the artificial intelligence component. In each mode, I will implement it so that it will make a range of moves based on the scenario. In the easier mode, it will make a less logical set of moves. In the more difficult mode, it will make a calculated set of moves based on the location of the other character and the action of the other character.

Algorithmic Plan:

The trickiest part will be implementing the artificial intelligence. The AI will need to consider the best attack, as certain areas on the character will have a different hitboxes and do different damage. The AI will need to calculate the set of moves that will deal the largest amount of damage or prevent the most damage dealt to the AI.

I will first research the best set of moves to do in Super Smash Bros given each scenario. I will separate the total possible outcomes with a couple scenarios. One is if the character is close range in the x axis but not the y axis. Another is if the character is close range in the y axis but not x axis. Another scenario is if the character is far away in both the x and y axis. And the last scenario is if the character is close range in the x and y axis. In addition, the direction and speed of the player’s character must be calculated in each of the scenarios as well, thus further altering the AI’s judgement.

For the easy mode, I will base the sequences of moves to do on my own judgement. Since it is an easy mode, the moves do not need to be perfect and should be very susceptible to mistakes. Reaction times would be slower for the easy mode (by potentially setting the close range to be lower, etc).

However, on the difficult mode, I will research the best set of moves to do based on what experts are stating. I will also research into the AI programs for one player Super Smash Bros. It will be the best case scenario, and human error will be eliminated in the AI.

Timeline Plan:

11/23: Finish the knockback settings, differering damages for hitboxes, and projectile motion into the game.

11/25: Successfully implement the AI into the program. AI must somewhat work

11/28: Finish the basic structuring of the AI into the program and have it be able to run too

12/6: Fix program so that it is less buggy and runs more smooth. Add comments and eliminate unnecessary code

Version Control Plan:

Git Repository, I will git pull my code onto my GitHub account.

Modules:

PyGame

TP2 Update:

I am no longer doing an easy and hard mode for the AI. Rather, I will implement a different AI for each character. In addition, I am planning on setting the character and opponent choice on one screen to make a more realistic, cleaner look to the project.

TP3 Update:

I made it so that range attack loses damage over time. I have incorporated knockbacks using projectile motion for all characters.

I have made 4 different AI systems, one for each character. I have also made the AI systems smarter. For Lucario, Pikachu, and Pit, I made it so that the AI would be able to detect if the player made an attempt to have a melee attack, and if the player did, the AI would move back and avoid the melee attack. For Bowser, since it is a tank and does not get knocked back as much, I made Bowser focus more on using its own melee attack against the player. Lucario AI’s system is the most balanced, using both melee and range attacks frequently. Bowser’s AI is more offensive and uses slightly more melee attacks than Lucario. Pikachu’s AI is based on its speed. Pikachu is able to stun the player, and if the player is stunned, Pikachu will move to the opponent and make a close attack, as its range attacks are not as powerful and frequent. Pit’s AI has projectile motion calculations. For most positions, Pit will calculate for each angle if the angle will hit the player and avoid any collisions with the platforms. Otherwise, Pit will adjust its position accordingly. Pit will only fire the lowest absolute value of the possible angles, as that arrow will stay in the air for a lower amount of time (since time and cosine of the angle are inversely proportional).

In addition, if the user wants to play with other player, the user can play multiplayer, which has another set of commands for the second user. In multiplayer mode, you can select if you want items to drop. There are three items: heart, apple, and ice. If a sprite collides with the item, the item disappears. Colliding with the heart gives one more life to the sprite. Colliding with the apple temporarily increases damage for the sprite. Colliding with the ice freezes/stuns the opponent temporarily

I also have not used different hitboxes for each character. I focused more on the AI component of the program.