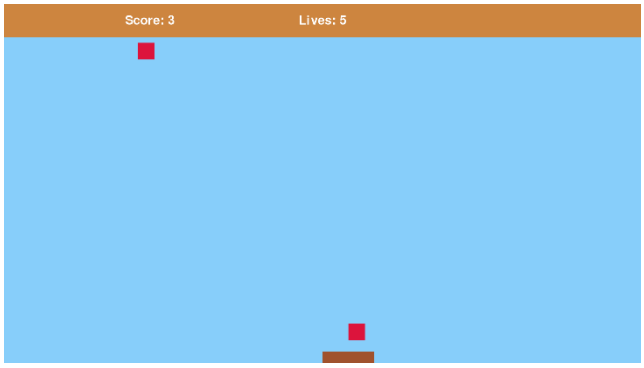
Chris Aring

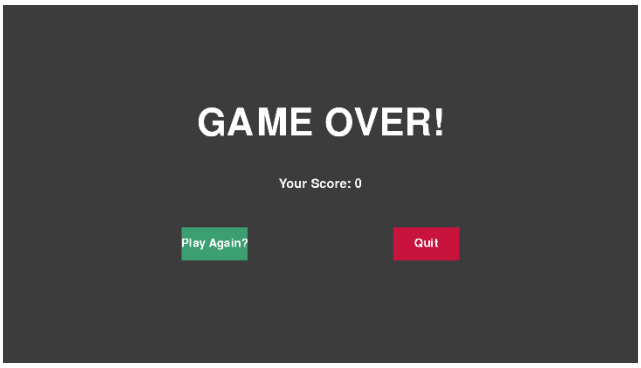
Apple Catcher

For my project I made a game where apples fall and players need to catch them. There are different types of apples with different effect and many changing variables that handle difficulty and fairness.

In the end my game functioned as expected. Apples are spawned in with a weighted chance and fall with a constant speed. The player moves a platform along the x-axis in an attempt to catch falling apples. The player starts with five lives and is supposed to catch red apples and drop green apples. Every green apple that they catch or red apple that they drop, they lose a life. Catching a red apple, however, increases their score by one. I added some variables and logic into the game to handle fairness so that every apple is catchable even as the difficulty increases. For example, an apple can't spawn more than half the screen size away from the previous apple and apples have a minimum time in-between spawns. This is to make sure that no apples spawn right after the previous apple so that every apple is catchable.

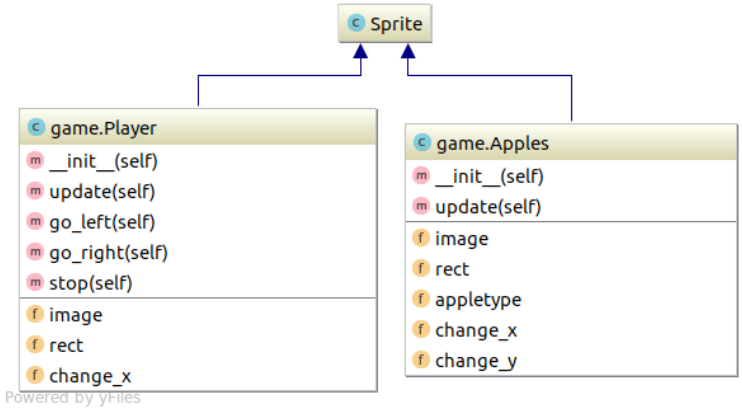


On every frame an apple has a chance of spawning but to make sure that the game remains fair, even if an apple is told to spawn on a frame, it won't unless the minimum time between spawns has passed. Over time, the apple's y-velocity increases, the player's x-velocity increases, apples have a higher chance of spawning on each frame, and the minimum time between each apple spawn decreases. These variables help to increase the games difficulty. All of these variables are changed to make things more difficult every 10 seconds but are all capped to make sure the game does not become impossible. The chance of spawning on a given frame does not start increasing until 30 seconds have passed and the minimum time between spawns does not start decreasing until all other factors have been capped at their maximums.



I have two main classes. One for the apples and one for the player. The player just has basic variables related to horizontal velocity and size / color. The apple class has variables and logic that decided what type of apple will be dropped and their position and vertical velocity. The rest of the logic is located in a main function. This function handles all game logic. The apples are stored in a sprite group and in another group that also contains the player (a group of all sprites). This allows the apples to be recognized individually for the game logic but also within an "ALL" class with the player that allows the scene to be wiped.

The game is split up into small logic components mainly in the "main" function. There are if statements that determine how hard the game is over time based on a timer. The timers are run based on the number of frames because the game is set to 60 frames a second. This allows for most of the logic because it is based on time. The apple class is called from the main loop based on logic which is based on the number of frames that have passed since the last apple spawn. Then the type of apple, position, and velocity of the apple are determined in the apple class. Then the player class is used when the user hits an arrow key to move the player. If the player catches the apple, the apple is removed from the apple sprite group and the combined sprite group and a point is added. If an apple is dropped, the apple is also removed from both sprite groups but a life is taken.



I had to choose the best way to keep track of the time. At first I used timer classes and other logic based on the timers but they slowed down the game and did not function as needed. It was too cumbersome to call the necessary functions and activate the timer itself. It was unnecessary because I already had a "timer" because the number of frames is a set value. I decided to base the time variables, on the frame count because I didn't need to add any more timers and it was an easy count to store and access.

The whole project went well and I learned a lot. I had to work alone because I was sick during the duration of this project so I missed out on the team aspect of the project which I felt would have made the project much different. Besides that, I enjoyed developing a game because it was something I was interested in. I was able to split the project up into parts that I feel comfortable with. For example I like to work on the bare functionality first, including the main game loop and basic features. Then I like to add more complexity to the game logic and small

additions like start screen and end game screen. Once I have all the functionality done, I then add the graphics and design the different screens to give a positive feel that matches the game. Then I test and adjust. I felt like I was able to do that during this project and it worked very well.

Going forward I have a much stronger understanding of, not only Pygame, but how python logic works and classes. I am able to create more complex games and similar projects because of my learned skills. I want to take this project further and add to it in many ways that I thought of while working on it. I only spent about 2 hours on it so I would like to add more in the future.