Project Reflection

Overview

Skyroads is a popular video game in which a player must avoid hitting barriers, while making sure they don't fall into an open void. In our simplified version of Skyroads, a player must move the car to avoid getting hit by barriers. When they get hit by a barrier, the game will end with a display message saying "Game Over".

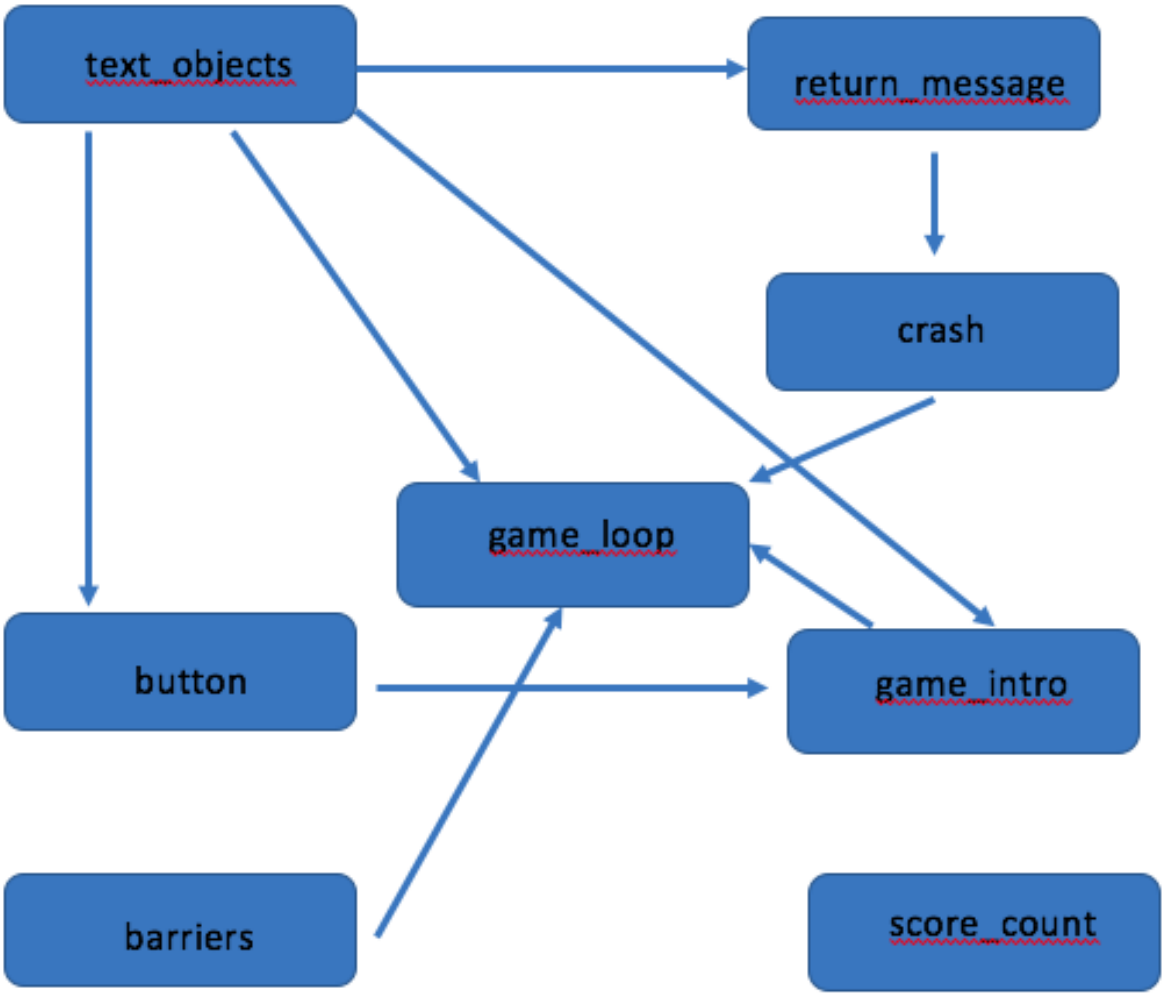
Results

Screenshots from the game

In our program, the user moves the car from right to left to avoid barriers (blue blocks) that are falling from random positions at random intervals. Once the car hits a barrier, the game will display a message called “Game Over” and return to the home menu after 2 seconds.

There is a score counter in the top left hand corner of the screen to display the score as you avoid barriers. The player can begin playing the game once they click on the “Play Now” button on the bottom of the third image. The player can quit the game at any point in time by pressing the escape key on the keyboard or simply closing the program.

Implementation *[~2-3 paragraphs + UML diagram]*



UML Diagram for our game

For Skyroads, we made use of a lot of different functions that played a role in the game_loop() function. This include making and moving the barriers, detecting collision between car and the barriers, and returning to the home screen once the game is over. We did not feel the need to make use of classes in this game. If we were doing something maybe a bit more complex, it would make sense to use classes. The functions except for the score_count function are all integrated and have been called in multiple other functions in order for the program to fully run efficiently.

There were several choice that needed to be made in order to make this game a success. One in particular that stood out to us was randomizing the barriers positions and how often they come. We could have had a barrier class and used the objects within the class to manipulate the barrier movement; however, we felt as though this was a better and more efficient way to handle the barriers.

Reflection

Throughout the project, we did a good job staying on track and reaching our goals. The project was appropriately scoped to what we wanted to do. There were challenges at times such as when Shreya's pygame wouldn't work, but we overcame the challenges and created a final game.

We went with the divide and conquer method. We took one aspect of the game that we wanted to focus on, implemented changes, and pushed our code. It worked out pretty well for the most part. There were some issues with github and merge conflicts; however, once those issues were resolved, the code came together and we have a working game.

If we were to do this again, we would probably try to pair program maybe a little bit just to see if it is something that would work for us.