LINKS:

* <https://www.bensound.com/royalty-free-music/track/memories>
* <https://images.ecosia.org/Tyedg9OuFw2kuGhNxy4mlnDe-rM=/0x390/smart/https%3A%2F%2Fvignette.wikia.nocookie.net%2Froblox%2Fimages%2F1%2F14%2FCop.png%2Frevision%2Flatest%3Fcb%3D20171125203755>
* <https://t3.rbxcdn.com/beac3ae0a4e5cf80d78772aebd429bee>
* <http://www.orangefreesounds.com/roblox-death-sound/>
* <https://www.freeiconspng.com/uploads/hd-victory-royale-fortnite-transparent-background-12.png>

README:

For my game, I made a cops and robbers type game. The user uses arrow keys to move the cop around the screen. The objective of the game is to use these movement controls to move the cop close enough to the robber to catch him. Upon catching the robber, the robber will instantly move to another location on the screen. This will increase the user’s score by 1 every time a robber becomes caught by the cop. Once the user reaches a score of 10, he/she wins the game and the program terminates.

OUTLINE:

The first thing I did to make this program was the same that I did for all pygame projects. I set up the main window, a while loop, and a kill switch so the user can exit the program at any time. Then, I used a simple blue rectangle as the cop and a red rectangle for the robber. I would worry about getting images for each character later on. After I made the simple block game using this, I simply replaced each rectangle with an image and threw a background on. I polished the game by adding a score feature, ending the game when the user reached 10 points, and by adding sound effects and music.

Upon programming this code, the hardest part was figuring out collision. The way that I figured this out was by making a secondary rectangle per each character as a hitbox, and kept mapping the x and y positions of the hitboxes to the x and y positions of the cops and robbers. In order to keep the rectangles invisible, I mapped them to be drawn on a new surface under the background. I called this surface the “ghost window”, as it wasn’t seen because of it was drawn under the background. Then I simply used the built in colliderect method in order to check if those rectangles ever collided, which triggered the remapping of the robber’s x and y positions and added one to the score of the user.