Paul Krusell & TJ Kim
Software Design
Interactive Programming

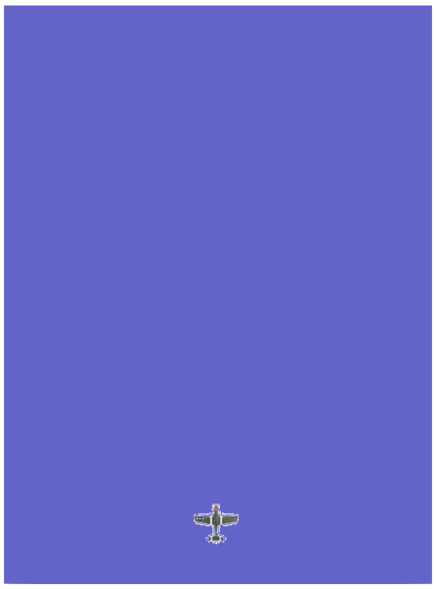
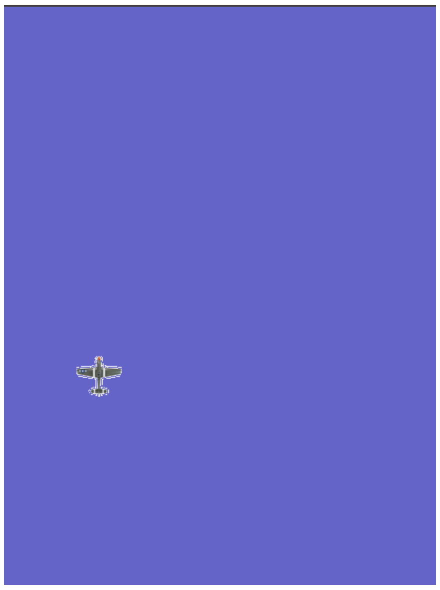
Project Writeup and Reflection

Project Overview

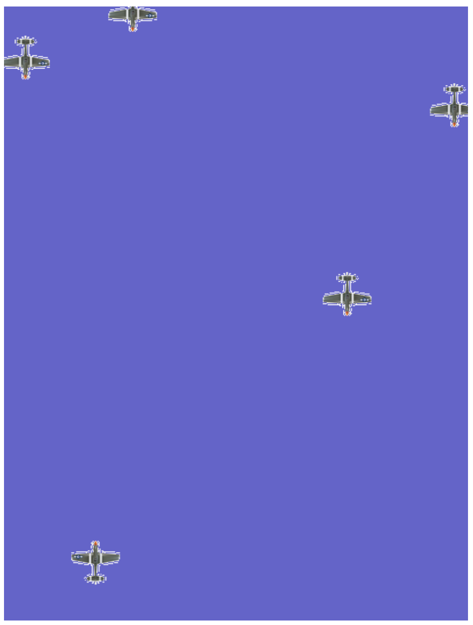
The main objective of this project was to create a simple 2d game that is reminiscent of an old arcade game. We created a game with an airplane flying from the top-view, dodging enemy planes trying to crash themselves into the player's. For major objectives, we tried to randomize foes, and have a user sprite controllable by the direction keys.

Results

For the mid-project check-in, our group successfully implemented pygame to get a user-controllable character on the screen. The arrow keys would move the character accordingly. Moreover, we successfully imported a sprite image to use as the character. Below are the images of the user plane traveling on the screen.

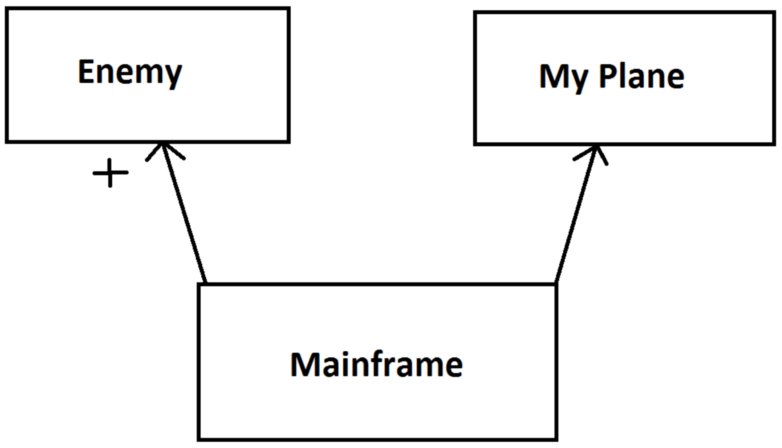



Afterwards, we added the enemy planes to appear fly downwards at the player. 5 planes were generated outside the screen at different locations and flew downwards. Once they flew across the screen, their path was reset to fly down over and over again. However, each plane's initial position would be randomized so the enemies didn't fly at the user in a straight line.



Once the user’s plane collides with the incoming plane, the game would end. However, the user’s plane must be moving when the collision happens, or else the enemy plane flies straight through the user’s plane.

Implementation



The mainframe function is the entry function for our game. It calls upon the enemy and myplane class to run. With every loop, the screen is updated accordingly to the other classes. First it loads the sprites from the other two classes. The My_Plane class initializes the position and velocity of the user’s plane. It also interacts with the while loop in the Mainframe mainloop function. As it updates itself, it checks for updates in position and collisions.The Enemy class is similar to the My_Plane class except there is no user input but there is a randomized component. The initial position of each enemy plane is randomized so they do not travel in one line. Once the collision happens, the while loop of the Mainframe's Mainloop recognizes it and exits.

A major design decision made was to not fully organize the python scripts in a distinct model-controller-view format. Many components of each seem to be overlapping with each other, and at some points it is a bit hectic to follow the code. Our team decided to leave the code the way it is because the example code (Pygame Pyman tutorial) we based our own code on had a similar format, and there were time restraints on the project.

Reflection

This project was a helpful experience for both group members. We generally created the game with most of the components we wanted. Our project was somewhat appropriately scoped. We got the major components we wanted: randomization, and a user-controlled character. However, we did not implement a point system as we assumed that shooting bullets out of our plane would be too difficult of a task for two beginner coders, and changed the game to dodging planes rather than shooting them. We had a decent plan for unit testing as we incrementally developed all parts of the game, first the user's character, then the enemy planes, then collision detection.

There were definitely issues with working together as a team. It was difficult scheduling common time in our busy schedule, so we each separately developed our own parts of the code and came together to merge them together. There were minor disagreements and misunderstandings we had to work out, but in the end, we were both content with the code. Next time, we would do all programming in a pair-programming situation and emphasize a higher priority to team projects.