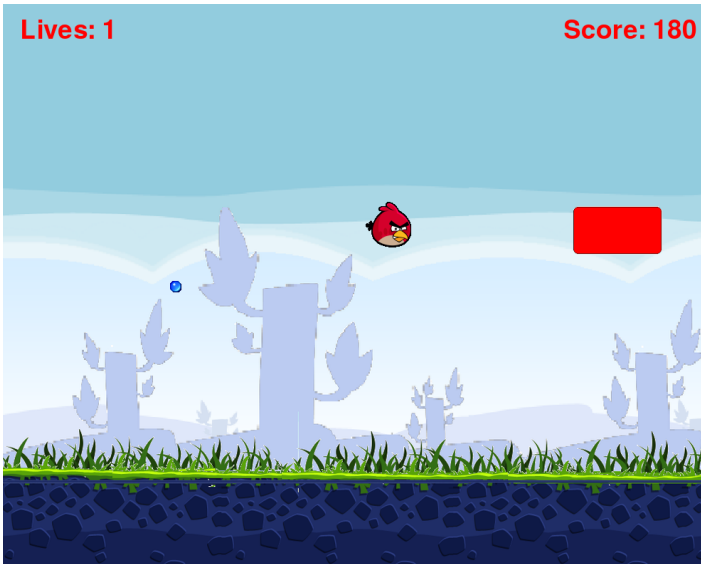
Project Overview:

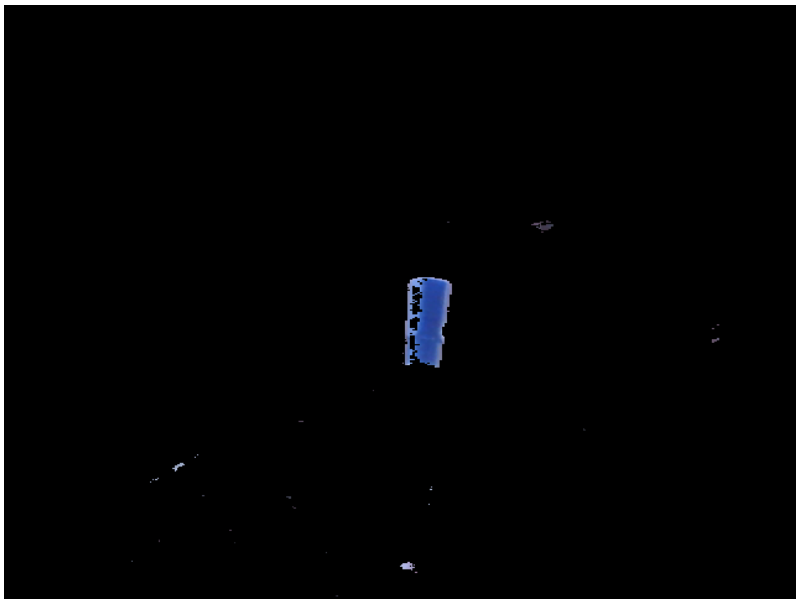
Through this project, we worked to create a variation of the popular game ‘Angry Birds’. Using the PyGame library, we explored the mechanics of creating a game with 2D-physics. At the beginning of the project, our main focus was to create an entertaining Pygame game with a bird the players could control through color tracking in OpenCV. While we were able to create a fully functional game, we did not have the time to integrate our webcam thresholding code into the game. However, we were able to successfully mask video feed to only show red objects.

Results:

Using keyboard controls, the player can move the bird relative to a point on the screen, and can then launch it with the space bar in a manner similar to a slingshot. There are three levels of the game; as the player hits the target, the next stage has an increasingly more difficult, moving target to hit. In each game, the player is given five birds, or ‘lives’. The player is able to see their current score and amount of lives. When they run out of lives, the game automatically closes. Below are a few sample pictures of what our game looks like:

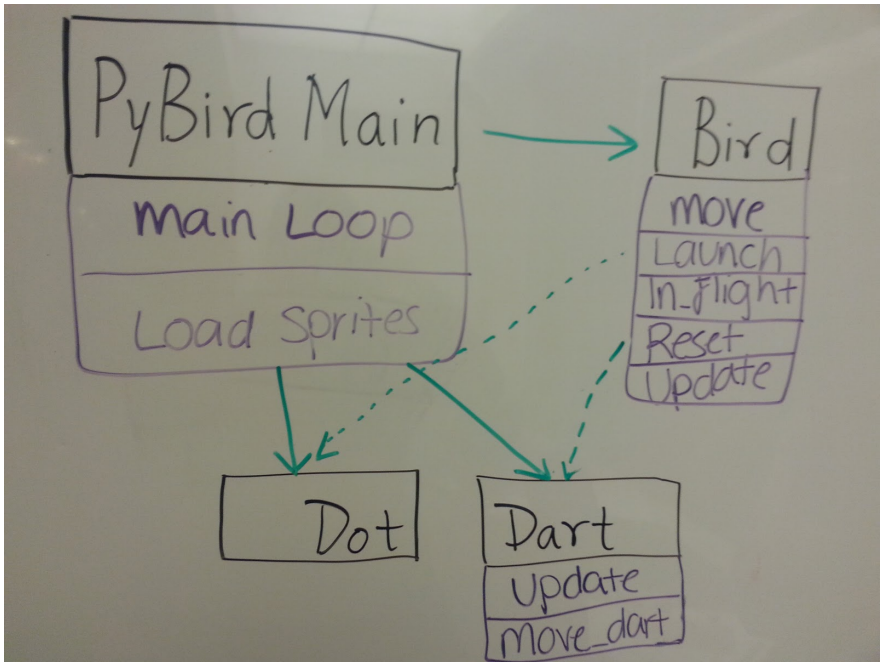


The other aspect of the project that we worked on involved learning how to take webcam video feed and use OpenCV to manipulate it in multiple ways. Using the sample code provided on the OpenCV-Python Tutorials website, we were able to adjust it in order to mask for red objects, blue objects, and basic black and white video feed. The images below demonstrate what we were able to accomplish and figure out throughout the duration of this project:



Unfortunately, we did not have enough time to both create a satisfactory game and integrate OpenCV controls, but the next step would have been to use HoughCircles circle detection in order to find a marker cap or some similar object and track the coordinates, using them as the coordinates of the bird.

Implementation:



The code has a class for each of the sprites in the game. It also has a main class called PyBirdMain which contains the major elements of the code like the main loop and the function which loads all the sprites. The dart class is updated using the move_dart to give the game some complexity.

We did take a design decision to try and test various numbers to satisfy the projectile motion of the bird. We initially decided to use actual physics to compute the velocities and accelerations in x and y directions. But, we ended up hardcoding numbers that gave out a sensible output. We took the decision because the time scale and frames processed in a second would differ in different situations making it difficult to use real gravitational constants.

Reflection:

Overall, we were successful in that we typically tried to figure out how to program and debug ourselves rather than asking someone else, meaning that we learned a lot through tutorials and online blogs. Conversely, this also hindered us because we ended up modelling our code based off of tutorials that were not organized well. We could have improved upon this by using the Model-View-Controller format to make our code more readable and easy to debug. At many times when we wished to test certain parts of our code, we had to print out statements and infer what they meant, which did work, but using a more structured format would have greatly sped up the process.

While we set a stretch goal for our project that we knew we would most likely not be able to reach (integrating OpenCV to control actions in the game), we generally set very realistic goals and scoped the project accordingly with the amount of time we had for it. Going forward, the biggest lesson we probably took from this was to ensure that our code is properly

formatted in a way that makes logical sense. In terms of this project, that would have meant using Model-View-Controller format, which would have made things easier.

With regards to teamwork, we had initially decided that Shruti would work on the Pygame part of the project, and Shivali on the OpenCV part. However, we ended up doing more pair-programming as we held team meetings, and because we were more efficient that way, we mostly both began working on the same things. There were no team-related issues that came up, but in terms of doing things differently next time, we both realized halfway through that our code was not optimally structured but did not change it, so next time we would definitely consider it more seriously.