Reflection

Project Overview

Based on the mobile app game 'Fruit Ninja', our project combines game mechanics with a motion controlled user interface. ninja.py uses the OpenCV library to track hand motion, and correlate any movement to a fruit-slicing sword.

Results

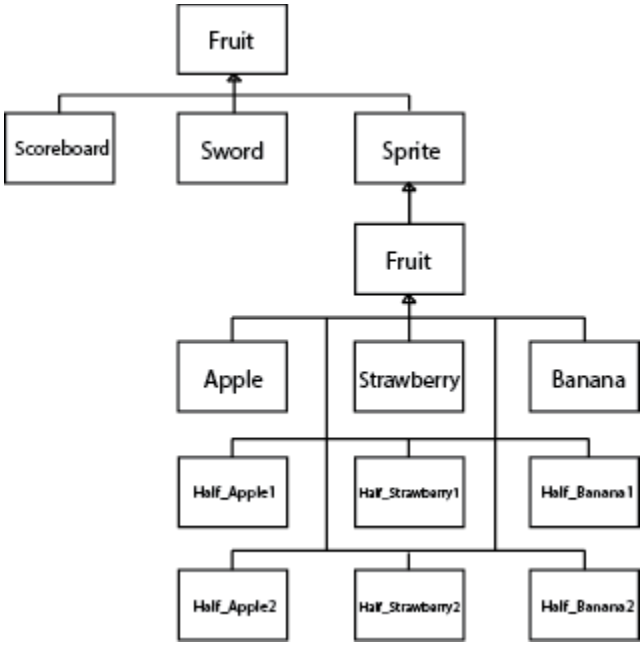
In the way we scaffolded our project, we ended up creating multiple levels of our gaming experience. Our first deliverable took the form of a mouse controlled Fruit Ninja, found in tests/ninja_oldv.py. The focus of this was familiarizing ourselves with PyGame, object-oriented programming, and graphics.

Our next major push in this project was learning to use OpenCV for object recognition. We used a background subtraction model to recognize and track any novel object (e.g., a hand).

Our final product had a webcam based user input, where the centroid of the player's tracked hand corresponds with the coordinates of the sword on the screen. To step up the graphics and complexity, this version also shows the fruit being cut in half once the sword intersects with it. A simple scoreboard is kept at the top to let the user know their stats as they play.

Implementation

We used three objects in this project: Fruit, Sword, and Scoreboard.



The fruit object was called on 9 separate occasions, with the only difference between the different fruits the image that appeared on the screen. Otherwise, all of the fruits would act the same way. The three fruits – Apple, Strawberry, and Banana – were tossed into the air using the class defined 'toss' function. The halves of the fruits fell according to the 'fall' function.

The sword object was used to represent input from OpenCV. The object was set up to take information from an easily changeable input, like a mouse or OpenCV, and move around the screen accordingly. When the rectangles associated with the different sprites collided, the fruit was removed from the fruits cache, and two corresponding half fruit objects were generated.

A big design decision made in this project was how we would take meaningful data from OpenCV. We debated whether we would base the process on color detection or background subtraction, but ultimately decided on the latter. It is more intuitive for the player, as they can use their hand instead of having to hold an object. While it has the disadvantage of being sensitive to background changes, it is ultimately more robust in allowing the user to play in any setting!

Reflection

Our workflow for this project was opposite of how most projects work: we essentially finished our project in the first week. This was mostly so that our second week could be freed up for our other classes, but had a secondary effect of burning us out. That being said, we were originally worried about the scoping of our project, but we had framed our idea in such a way that the MVP was achievable within hours, and some of the stretch goals were reached by the end of the week. We split the project up into manageable chunks: creating falling fruit, cutting the falling fruit with a mouse, throwing and

cutting the fruit. After we got a decent MVP working with a mouse, the OpenCV input was added to give the game another dimension.

As a team, we worked very well together. A lot of the programming and learning was done outside of the meetings, and we were never working on the same issue at once. Once the MVP was created, one of us worked on game mechanics like the motion and cutting of the fruit, and the other worked on the OpenCV input. During the work phase of this project, we would check in almost every other day to look over and compile each other's work, and create a new plan for moving forward. While this was a very efficient approach, this project became more about reaching a final product instead of creating experience pair-programming. Honestly, we would both probably use this approach next time since both of us are goal-oriented rather that process-oriented.