**Work in Progress Report 2**

*Due Dec 4th*

Matthew Meade & Abhi Agrahari

Major developments/breakthroughs(reference specific code please):

The majority of this 2 week time was spent reorganizing our main code (improving the architecture). As you know, we were able to create classes that extended one another and showed that inheritance works. We created a sprite class, as you suggested, so that nearly everything was a sprite (the only thing that isn’t a sprite is the eye and the button). We also implemented audio, a give up button, and spikes into our game! The latest and most stable release is 4.0: Ewok.

Major Challenges/setbacks( reference specific code please):

It did take a couple days to develop inheritance. It also took us a while to understand what you wanted us to do involving sprites. The main challenge was reorganizing everything to become a sprite as seen in version Colo Claw Fish, and creating the eye that moves but is fixed to an area.

We originally had vectors hold the position of the player. So one of the most time consuming tasks when transitioning to sprites was trying to get everything to work in general. Our sprite constructor used vectors to store their position, but we found this very troublesome when creating boxes, and other sprites like the door, as we had to use “vPosInitial” to be able to construct the sprite. After that, lots of things just stopped working: the door fell with our crab, and the boxes were all on top of each other. Because of this, we scraped the idea of using vectors for position and went back to normal “fX” and “fY”.

We had tried multiple ways to get the pupil to remain within the eye. The first way that we tried to do it involved using the “atan2()” function, but that failed horribly as seen in the scratch program “circle\_that\_looks\_at\_you”. If you moved the mouse to certain positions, the “eye” would start jumping around violently. After two other versions, we had found a stable way to get the eye to move. In “eye\_moves\_ver2”, we tried to use only PVectors and their functions to do it, but what ended up happening is the eye moved in a (smaller) proportion to the rectangle (the user), so it didn’t look like the eye was “looking” at the rectangle.

Any modifications to your specifications/release schedule:

As we think that the reorganization of our code, to include inheritance and sprites, is a major development, we would like to add version 3.0.1: Colo Claw Fish.

In version Clone Trooper, we have decided to only include 4 levels and save the rest for later so that we are able to add lasers to the later levels. This way we can design better/more evened out/ progressively harder levels. We will still have 42 levels by the end of our game.

**Description of your scratch/test program:**

Describe the generic concept you needed to test out:

We needed to find a way to create a stationary eye that looks at you.

Source any web site/book that helped you with that concept:

The code from your GitHub and website helped us understand how to use PVectors. We also looked at the PVector functions on the Processing Reference website which is listed in the sources document.

Describe the code and the lesson that you learned from it:

The final version of the scratch for the stationary eye is “Stationary\_Eye\_That\_Looks\_FINAL”. You move by using “WASD”.

When you (the rectangle) moves right, the eye moves slightly right. When you move left, the eye moves slightly left to match your movement. If you were to move up, depending on where you are on the screen, the eye follows you. This happens by having one vector “vEye” set to the position of the rectangle. “vCenter” is then subtracted from “vEye”. The screen is translated by vCenter before drawing the “eye”. vEye is normalized so that the ratio of the lengths remains the same, but is shortened. We then multiply vEye by 20 so that the movement is visible.

We learned how to get an eye to follow someone, and that there was a fairly simple method to do it using PVectors.

Describe any challenges that you enjoyed in integrating this scratch code into your major project:

Although it was time consuming, fortunately, there were no problems integrating the eye scratch into our main program!

We had problems integrating our give up button scratch (“Button\_SpritesheetAnd\_array\_of\_sounds”). In this scratch we had all of the images we wanted to display in one large image, and then displayed part of the large image to show the smaller ones. When we added it to our main program, no matter what we did it didn't work. We realized that we had to split the spritesheet up into an arraylist, like you do in SS 10. Once we made that change, our main program started working.