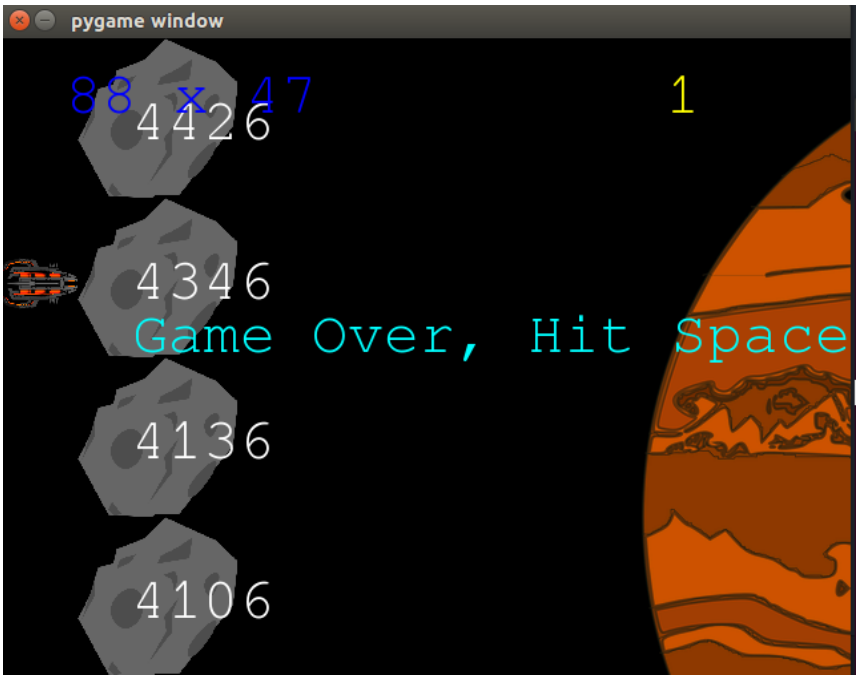
Project Overview

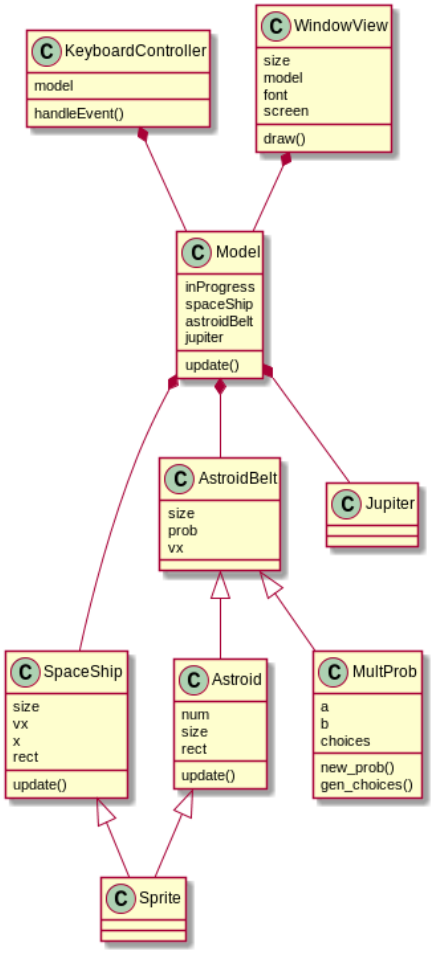
Flappy Nerd: Spaceship edition lets the user engage by solving multiplication problems and flying a spaceship through asteroids.

Results



We were able to accomplish an aesthetically pleasing and feedback heavy experience that allows users to test their mental math, score points, travel through space, and receive auditory feedback if they done what's required of them. Individually, we met our learning goals. Michelle was able to learn more about classes through exposure and guidance from Kevin. Kevin explored adding dynamic traits to enhance user experience from feedback, iteration, and Michelle's background.

We delivered an elegant demonstration of a package (that arguably has hard documentation) and provide an experience that people feel inclined to "try again" once more.



Implementation *[~2-3 paragraphs + UML diagram]*

The final resulted from iteration of our MVP, first by adding sprites and then by adding sounds. Each sprite or image in our code is a class object that gets called and meets in "Model", which updates on each restart of the game. "Model" is then called in WindowView to display what is inside the Model and controlled by KeyboardController.

Asteroid and Spaceship inherent from sprite, and Asteroid Belt is built by a culmination of a sprite and random generation of multiplication problems. Because Jupiter is a background, it is not a sprite.

One thing to note is that we had to have some floating values outside of pygame's rectangle (but we have rect round it up later.) An observation we made is that sprite cannot move an object a fraction of a pixel, so we kept the fractional value and round it off later when appropriate.

When we first created our MVP, we used a plain rectangle shape instead of sprites and decided to incorporate them, though adding number of classes, in order to enhance the user experience.

Reflection

Off that bat, the team could have benefited from more onboarding. Pygame's strict notation required a learning curve and began to feel like learning another language in it of itself. However, once used to notation, it began easier to iterate.

We took advantage of studio time to pair program and where able to get most of sprites taken care of in class. We met outside of class around 3-5 times to fix any issues that may have arose, Kevin open to answer any question, and Michelle ready to brainstorm how to get to the next step. Though we met often, we could have used this time to pair-program some more, as much of the games experience enhanced when we both worked together instead of apart.

We had no issues and think worked quite well, understanding of each other's limitations and/or level of engagement with the project. As always, there are areas we could have dedicated time to work and enhance the UX, such as adding levels and

not using double digits off the bat, but building it allowed us to grow comfortable with Pygame.

Kevin admits, moving forward, he should delegate more so as to not overwork and Michelle can be more vocal about what she can and cannot accomplish. Michelle should also push to Github more (she has a problem with pushing to the internet for everything, all the time.) Project four prepared us for project 5, in which, we have to do this over, but on a larger scale. We enjoyed working together and had fun building Flappy Nerd.