Sean Foley
Tehya Stockman

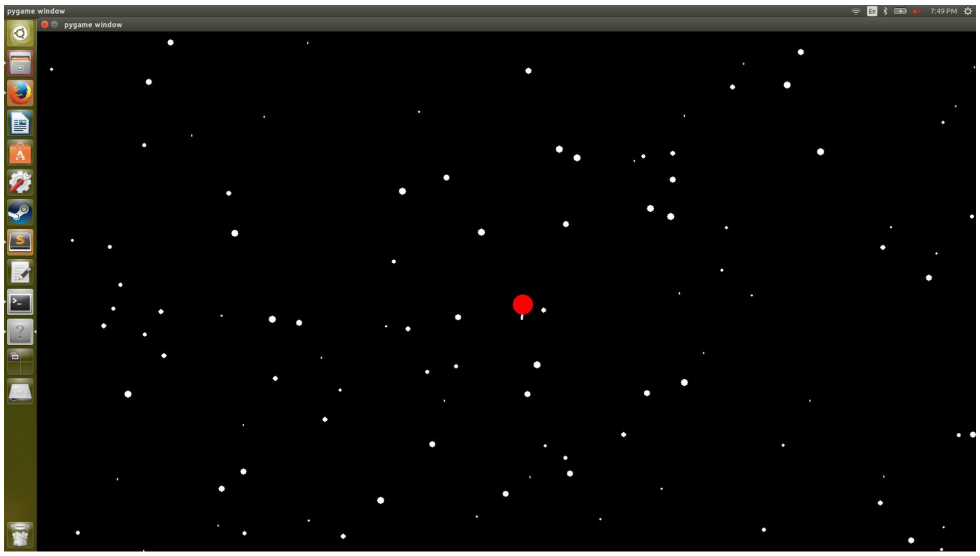
# A Trip through Space

**Project Overview:**

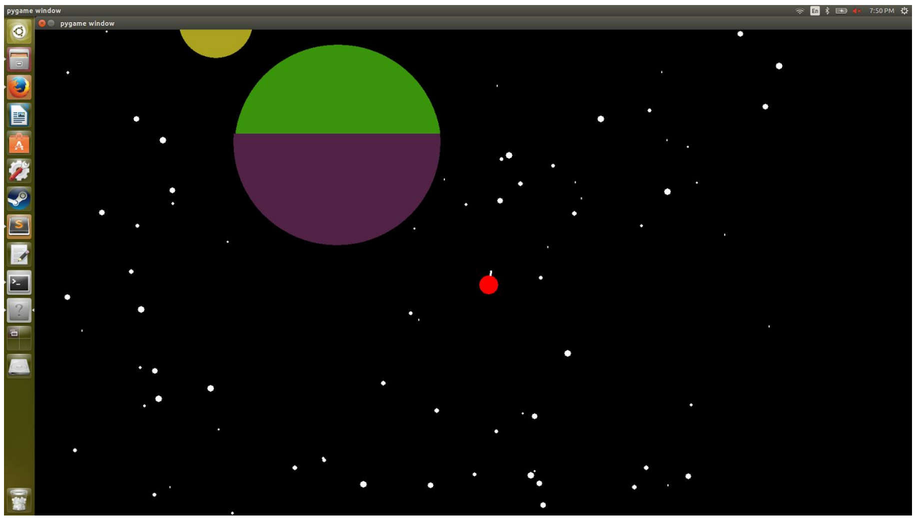
Come on an intergalactic journey through the stars! Maneuver around stars while watching them combine and supernova. Feel the effects of gravity and full advantage of the stars that cross your path. If you want an even more mesmerizing experience, press caps lock for a trippy effect. Goals don't matter when you're exploring the universe, just have fun!

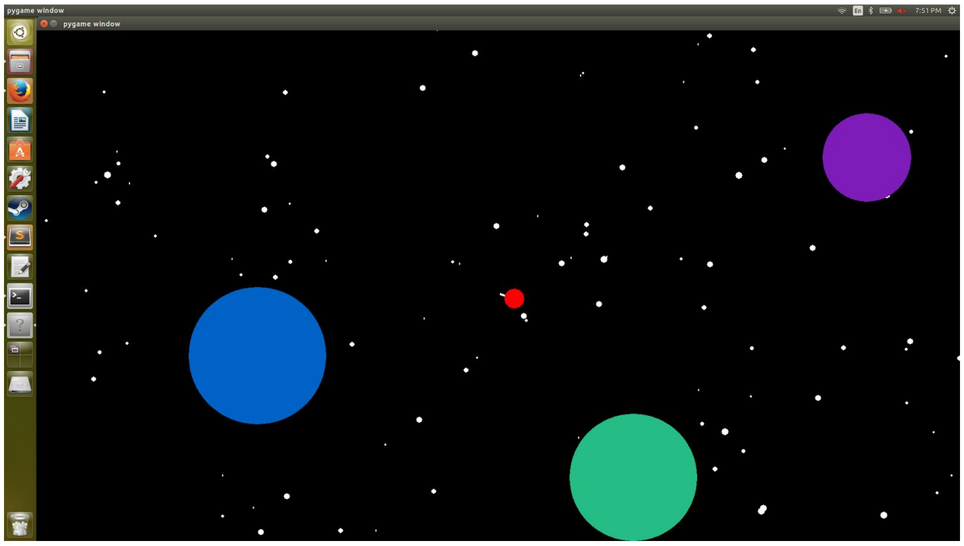
**Results:**

In our game, the user controls a spaceship and travels around a world  of space (of which some features are randomly generated at the beginning).  This game currently has no objective and is instead mainly focused on the interactions between different objects in the game. We decided to add increasingly complex interactions between the objects as we developed the game. The game will conclude, however, when the spaceship collides with one of the stars.



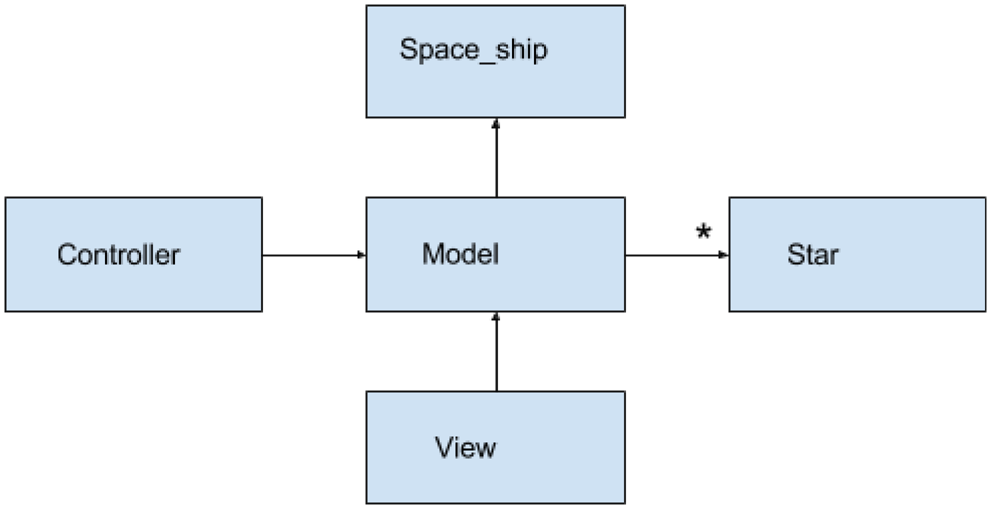
We created stars of a random variety of colors, masses, and radii, which attract other stars as well as the user's space ship.  These stars can move throughout space, combine with other stars that they collide with, and split when they reach a certain size.  The biggest challenge we faced was implementing the combine and split functions.





## Implementation:

We divided our program into model, view, and controller classes, as was recommended. This made our code more organized and easier to add to when we then wanted to make the game more complex.



**Model:** This takes in Space_ship, including all of its attributes, as well as a list of all of the stars (as they are created) and all of their attributes.
**Space_ship:** This is described by a starting position, velocity, radius, and mass.
**Star:** This class is very similar to Space_ship though has additional features.
**View:** This class takes in the screen_size, world_size, and the model to create the background and visual representation of the model at each step.
**Controller:** This class checks a list of user input that the main loop has and adds to it according to user input.

Our major algorithms describe how the stars and spaceship interact with each other. These include: gravity (force/acceleration/velocity), collisions, combine, and split. One of our major design decisions came about when implementing the combine and split operations for the stars. In order to implement these functions we had to make a copy of the initial stars_list so that we could mutate the list without affecting the for loop iterating through it. At the same time, because we were doing operations on elements of the copied list based on the original list, it was important to ensure the stars we were referencing were actually in the copy. This issue was the one really challenging bug we faced, and we ended up solving it with a series of if statements. Because we found the problem to be a recurring issue only after the split function was also being called, we decided to alter the split function so that there is no chance for newly-created stars to be touching each other on the step that the star is split.

**Reflection:**

In the future, it would be great to add a distinct goal to the game or expand upon the features to include some of the more outlandish ideas we were originally considering for this project. We could have also organized our functions in a more systematic way within the classes. One of the main issues that we are currently facing is how our combine function sometimes draws an error now that we have implemented the split function.

We worked mostly by pair programming, although we did implement a few functions apart from each other. There were no issues with working together; we stayed on task and got a lot of things done in each sitting.