Vision

Our original goal was lofty. Our team aimed to create an expandable physics engine with a graphical interface. We believed this was a viable project with components perfectly tailored to OCaml and functional programming with other more challenging (yet achievable) aspects. The OCaml Module system would allow us to create an expandable codebase while limiting code reuse—all things are governed by some equations but some situations might not use friction for example. Functional Programming in general fits physics as they are a better representation of our basic model of mechanics instead of using imperative methods and procedures from other languages. Some aspects of implementing this were easy: movement can be very easily modeled by functions (i.e. with $x(t) = xi + vi\ t + \frac{1}{2}\ a\ t^2$ all we need are initial conditions xi and vi the mass and the net force on an object and we can get the position at any time—perfectly fitting functional programming). However, we discovered complications while implementing collisions. With functional programming, these are not as easily implemented. The most complicated piece by far is the graphics. OCaml itself has a few graphics modules provided. We found an OCaml implementation of openGL, a C-based graphics library. We believed that integrating this library with our codebase would be, even though difficult, feasible. However, we found out that the package itself is more inclined to pre-determine images rather than the continuously drawn frames that the physics provides. We were able to get graphics displaying test images and moving objects but could not put it together with our state system. We did implement a counsel output that allows us to see the output.

Summary of Progress

Between MS2 and MS3 we focused on implementing collision detection, vibrations/oscillations, and graphics.

- The collision detection checks for intersections between objects and changes the forces, momentum, and velocities. The math for collisions changes depending on the objects involved so collisions require us to check for each combination of object types (squares and rectangles).
- Vibrations and oscillations are our second set of simulations. The movement of a mass-spring-damper system is determined by the mass, spring constant, damper coefficient, and the forcing function being applied to the mass.
- We continued to attempt to integrate graphics. We successfully made test implementations of the lablGL library but could not integrate it with our state system.

Activity Breakdown

Salvatore Natale

Collision Detection

Hrs: 25

Jake Lawson

Vibrations

Hrs: 20

Alec Galin

Graphics

Hrs: n/a

Eric Yoon

Graphics

Hrs: n/a

Productivity Analysis

In some areas of our productivity, we improved while in others we were stagnant. We were able to complete our sprint goals for the physics aspects of the project and our time estimates were relatively accurate. Collisions were relatively difficult but we did get it implemented eventually. We could not implement our graphics goals. Our estimates and goals were based on previous experience in other classes with GUIs. However, those languages had better implementations of graphics. Because of OCaml's lack of support, we had to build many of the systems from scratch and we were unable to achieve this.