How it Does it

Here is one of our methods from the Engine class, this is where we incremented gravitational force in between two objects.

```
public double gravAttraction(Projectile p, Projectile p2, double dist) {
        if (dist != 0) {
                double force = G * p.getMass() * p2.getMass() / (dist * dist);
                double acc = force / p.getMass();
                double ang = Math.atan2(p2.getY() - p.getY(), p2.getX() - p.getX());
                p.changexAcc(acc * Math.cos(ang) * timeSpeed);
                p.changeyAcc(acc * Math.sin(ang) * timeSpeed);
                return force;
        }
        return 0;
}
```

Here we used the equation F = (G * m1 * m2)/ r^2, or Newton's Law of Gravitation, to calculate for the acceleration for the objects.

Challenges

This is some weird syntax that we learned while programming the project, in this way you can declare a certain method to polymorph with each instantiation of a class.

```
buttons[0] = new Button("Mass", "create a massive object", p.width - 50, 100, 30, true) {
        public void click(PApplet p, Engine e) {
                e.addProjectile(inputFrame.getInputs()[0], inputFrame.getInputs()[1], p.mouseX, 
p.mouseY);
        }
        public void init() {
                addInputFrame(2, 20, 100);
                String[] names = {"Mass", "Radius"};
                inputFrame.setInputNames(names);
                float[] inputs = {400000, 100};
                inputFrame.setInputs(inputs);
        }
};
buttons[1] = new Button("Vector", "change the velocity and direction of an object", p.width - 50, 150, 
30, true) {
        public void click(PApplet p, Engine e) {
                e.vector(inputFrame.getInputs()[0]);
        }
        public void init() {
                addInputFrame(1, 20, 100);
                String[] names = {"Scale (1:n)"};
                inputFrame.setInputNames(names);
                float[] inputs = {1};
                inputFrame.setInputs(inputs);
        }
};
```

This is the really awkward input frame we programmed because we wanted to be able to change inputs while running the program, and you can't really do that with console input and JOptionPane

```
public InputFrame(int numInputs, float x, float y) {
        super("", "", x, y, 0, 0, false);

        inputs = new float[numInputs];
        inputNames = new String[numInputs];
        buttons = new Button[numInputs * 2];
        super.setW((BORDER_SIZE * 2) + (BUTTON_SIZE * 2) + MID_GAP_SIZE);
        super.setL( ((BORDER_SIZE * 2) * inputs.length) + (BUTTON_SIZE * inputs.length) );

        for (int i = 0; i < inputs.length; i++) {
                inputs[i] = 0;
        }
        for(int i = 0; i < inputNames.length; i++) {
                inputNames[i] = "Input " + i;
        }
}

public void init() {
        int index = 0;
        int index1 = 0;
        for(int i = (int) (y + BORDER_SIZE); i < y + l; i += BUTTON_SIZE + (BORDER_SIZE * 2)) {
                buttons[index] = new Button("", inputNames[index1] + "-", x + BORDER_SIZE, i,
BUTTON_SIZE, false){
                        public void click(PApplet p, Engine e) {
                                inputs[index1]--;
                        }
                };
                if (index < buttons.length) index++;
                buttons[index] = new Button("", inputNames[index1] + "+", x + BORDER_SIZE + BUTTON_SIZE +
MID_GAP_SIZE, i, BUTTON_SIZE, false){
                        public void click(PApplet p, Engine e) {
                                inputs[index1]++;
                        }
                };
                if (index < buttons.length) index++;
                if (index1 < inputNames.length) index1++;
        }
}
```

Me and Michael weren't exactly sure what we wanted to have 2 objects do when they collided, so we decided to have them either break up into pieces or absorb one another's masses when at higher energy impacts.

```
public void collision(Projectile p, Projectile p2, double dist) {
        if (dist < p.getDiameter() / 2 + p2.getDiameter() / 2) {
                if (p.getMass() <= p2.getMass())
                        collide(p, p2);
                else
                        collide(p2, p);
        }
}

// p2 is the larger mass
public void collide(Projectile p, Projectile p2) {
        int limit = (int) (Math.random() * 9) + 1;
        if (p.getMass() > p2.getMass() * .5)
                for (int i = 0; i < limit; i++)
                        addRanProjectile(p.getMass() * .5 / limit, p.getX(), p.getY(),
                        p2.getxVel(), p2.getyVel(), p.getDiameter());
        p2.setxVel(.9 * (p2.getxVel() + (p.getxVel() * (p.getMass() / p2
                        .getMass()))));
        p2.setyVel(.9 * (p2.getyVel() + (p.getyVel() * (p.getMass() / p2
                        .getMass()))));
        p2.setMass(p2.getMass() + p.getMass() * .25);
        p.setMass(1);
}
```

Along with many other git problems that we're not sure how to show you, that's most of what dragged our program down.