PS3b - N-Body Simulation: Using Newton's Laws of Physics, Animate the Universe

Part B of this assignment was to add Newton's laws of physics to move the planets, which was by-far the hardest part of the assignment. I had difficulty moving the planets the way it was suppose to be moving. To simulate the motion of N particles in the plane mutually affected by gravitational forces and to animate the results took some time.

Using the back-bone implementation that I created in PS3a, I had to add in the physics. With the particle already in position using standard drawing, I had to repeat this process each time step until a designated stopping time. I was to basically to keep drawing the planets rapidly until I killed the program.

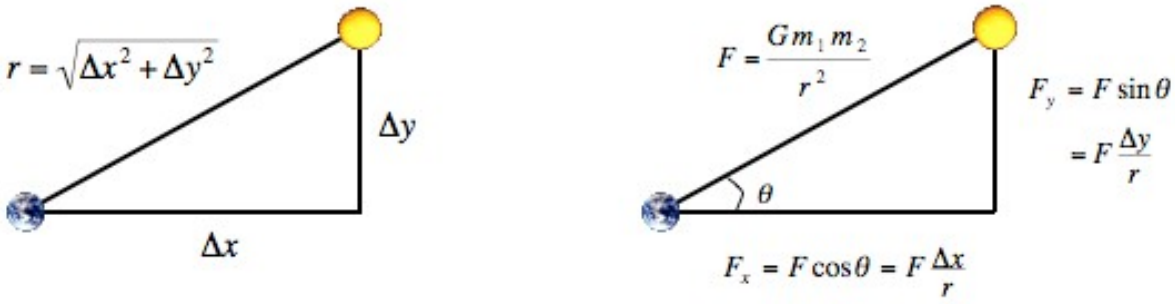
The Body class was extended with mutators so that the physics simulation can modify the velocities of each object. One key function that I wrote outside the Body class and in main was the function:

9: sf::Vector2f force(Body p1, Body p2)

which is Newton's law of universal gravitation. It asserts that the strength of the gravitational force between two particles. It first finds the product of the two masses then divided by the square of the distance that is between them and then scaled by the gravitational constant:

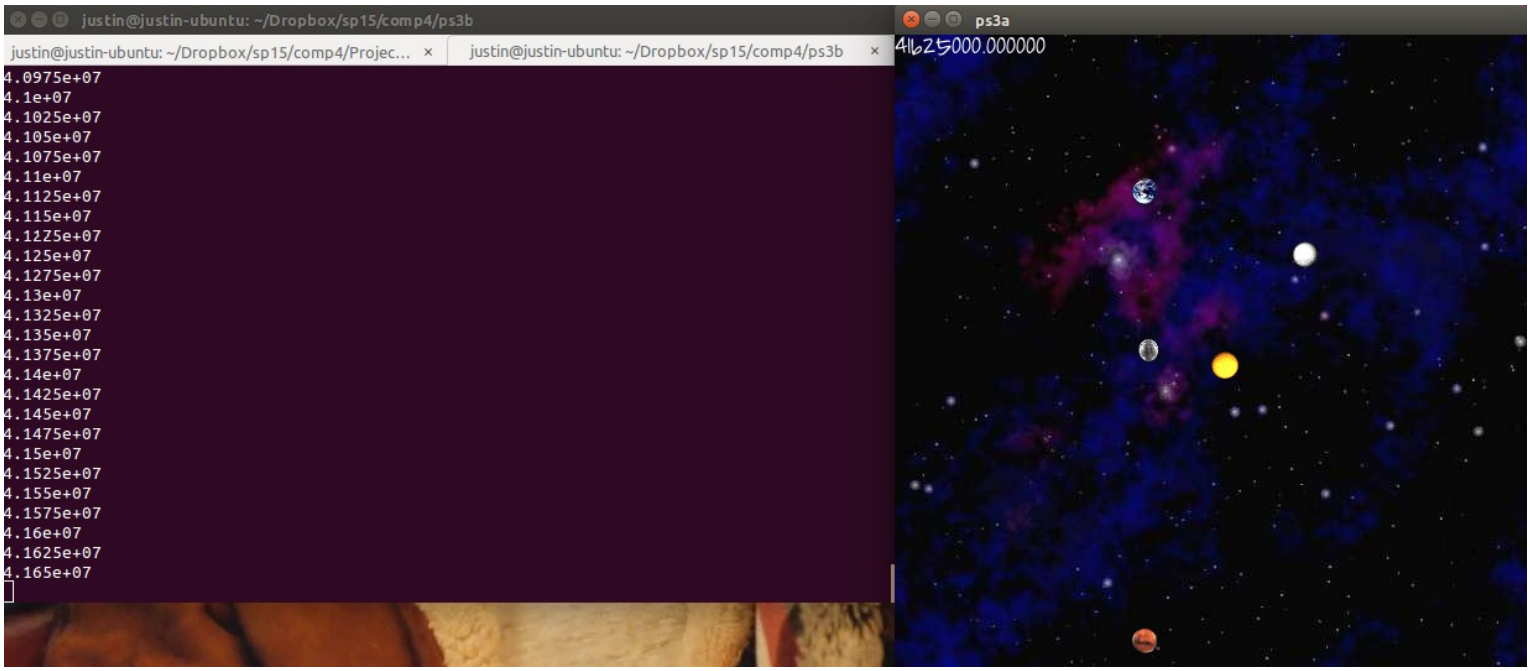
6.67 × 10^-11 N m^2/ kg^2

In this implementation, the Cartesian coordinate system was used to represent the position of each planet so we can break up where each planet by breaking it up into x and y components (or Fx, Fy).



Newton's law of universal gravitation in terms of how it is positioned using Fx and Fy on the right.

The elapsed time and when to terminate the simulation when time has elapsed beyond the time limit at the command line are kept tracked of in the main program. The time elapsed is also shown at the top right of the simulator. As extra credit, the theme to 2001: A Space Odyssey is also used!



Screencap of the simulator running with the time elapsed in the terminal and on the top right of the simulator.

```
    1: #include <SFML/Graphics.hpp>
    2: #include <SFML/Window.hpp>
    3: #include <SFML/Audio.hpp>
    4: #include <iostream>
    5: #include <vector>
    6: #include <cmath>
    7: #include "Body.hpp"
    8:
    9: sf::Vector2f force(Body p1, Body p2)
   10: {
   11:   sf::Vector2f planet_force;
   12:   double totalForce, dx, dy, radius, radius_squared;
   13:   double grav = (-6.67e-11);
   14:
   15:   dx = (((p1.getPos()).x) - ((p2.getPos()).x));
   16:   dy = (((p1.getPos()).y) - ((p2.getPos()).y));
   17:
   18:   radius_squared = pow(dx, 2) + pow(dy, 2);
   19:   radius = sqrt(radius_squared);
   20:
   21:   totalForce = ((grav) * (p1.getMass()) * (p2.getMass())) / (radius_squared)
;
   22:
   23:   planet_force.x = totalForce * (dx/radius);
   24:   planet_force.y = totalForce * (dy/radius);
   25:
   26:   return planet_force;
   27: }
   28:
   29: void removeWhitespace(std::string& str)
   30: {
   31:   for (size_t i = 0; i < str.length(); i++)
   32:     {
   33:       if (str[i] == ' ' || str[i] == '\n' || str[i] == '\t') {
   34:         str.erase(i, 1);
   35:         i--;
   36:       }
   37:     }
   38: }
   39:
   40: int main(int argc, char* argv[])
   41: {
   42:   int planet_count;
   43:
   44:   double univ_radius;
   45:   double time, time_step, current;
   46:
   47:   std::string planet_name;
   48:   std::string planet_filename;
   49:
   50:   std::vector<Body> planets;
   51:
   52:   sf::Vector2f t_force;
   53:
   54:   sf::Image background;
   55:
   56:   sf::Music music;
   57:
   58:   sf::Font font;
   59:   sf::Text timestamp;
   60:
```

```
   61:   if(argc != 3)
   62:     {
   63:       std::cout << "error: not enough/or too much arguments" << std::endl;
   64:       exit(1);
   65:     }
   66:
   67:   time = atof(argv[1]);
   68:   time_step = atof(argv[2]);
   69:
   70:   if(!music.openFromFile("2001.ogg"))
   71:     return -1;
   72:
   73:   music.setLoop(true);
   74:   music.play();
   75:
   76:   if(!(font.loadFromFile("DJB Poppyseed.ttf")))
   77:     return -1;
   78:
   79:   timestamp.setFont(font);
   80:   timestamp.setScale(0.5f, 0.5f);
   81:   timestamp.setPosition(0.f, 0.f);
   82:
   83:   if(!background.loadFromFile("starfield.jpg"))
   84:     return -1;
   85:
   86:   sf::Texture texture;
   87:   texture.loadFromImage(background);
   88:
   89:   sf::Sprite sprite;
   90:   sprite.setTexture(texture);
   91:   sf::Vector2f size;
   92:
   93:   size.x = background.getSize().x;
   94:   size.y = background.getSize().y;
   95:
   96:   std::cin >> planet_count >> univ_radius ;
   97:
   98:   for (int i = 0; i < planet_count; i++)
   99:     {
  100:     Body *planet = new Body (planet_count, univ_radius, size);
  101:
  102:     std::cin >> *planet;
  103:     getline(std::cin, planet_filename);
  104:     removeWhitespace(planet_filename);
  105:     planet_name = planet_filename;
  106:     planet->setName(planet_name);
  107:     planet->setFile(planet_filename);
  108:     planets.push_back(*planet);
  109:     }
  110:
  111:   sf::RenderWindow window( sf::VideoMode(size.x, size.y), "ps3a");
  112:
  113:   while(window.isOpen()) {
  114:     sf::Event event;
  115:     while(window.pollEvent(event)) {
  116:       if(event.type == sf::Event::Closed) {
  117:         window.close();
  118:         break;
  119:       }
  120:     }
  121:     window.draw(sprite);
```

```
  122:
  123:     for(std::vector<Body>::iterator iter = planets.begin(); iter != planets.
end(); iter++)
  124:       {
  125:         iter->setUpPos();
  126:         window.draw(*iter);
  127:       }
  128:
  129:     if (current < time)
  130:       {
  131:         for(std::vector<Body>::iterator iter1 = planets.begin(); iter1 != pl
anets.end(); ++iter1) {
  132:           sf::Vector2f Force_net;
  133:           for(std::vector<Body>::iterator iter2 = planets.begin(); iter2 !=
planets.end(); ++iter2) {
  134:               if (iter1->getName() != iter2->getName())
  135:                 {
  136:                   t_force = force(*iter1, *iter2);
  137:                   Force_net.x = Force_net.x + t_force.x;
  138:                   Force_net.y = Force_net.y + t_force.y;
  139:                 }
  140:           }
  141:           iter1->setFnet(Force_net);
  142:         }
  143:
  144:         for (std::vector<Body>::iterator iter = planets.begin() ; iter != pl
anets.end() ; iter++ ){
  145:           iter->step(time_step);
  146:           window.draw(*iter);
  147:         }
  148:         std::cout << current << std::endl;
  149:         timestamp.setString(std::to_string(current));
  150:         current += time_step;
  151:       }
  152:     window.draw(timestamp);
  153:     window.display();
  154:   }
  155:   return 0 ;
  156: }
  157:
```

```
    1: CC = g++
    2: OFLAGS = -c -Wall -ansi -pedantic -Werror -std=c++0x -lboost_unit_test_frame
work
    3: CFLAGS = -Wall -ansi -pedantic -Werror -std=c++0x -lboost_unit_test_framewor
k
    4: LFLAGS = -lsfml-window -lsfml-graphics -lsfml-system -lsfml-audio
    5:
    6: all:    NBody
    7:
    8: NBody: Body.o main.o
    9:         $(CC) Body.o main.o $(CFLAGS) $(LFLAGS)  -o NBody
   10:
   11:
   12: main.o: main.cpp
   13:         $(CC) $(OFLAGS) $(LFLAGS) main.cpp
   14:
   15: Body.o: Body.cpp Body.hpp
   16:         $(CC) $(OFLAGS) Body.cpp
   17:
   18:
   19: clean:
   20:         \rm -f *.o  *~ NBody main Body
```

```
 1: // Copyright [2015] Justin Nguyen
 2:
 3: #include <SFML/Graphics.hpp>
 4: #include <SFML/Window.hpp>
 5: #include <iostream>
 6: #include "Body.hpp"
 7:
 8: Body::Body(int univ_size, double univ_radius, sf::Vector2f window_size) {
 9:   setUniverseSize(univ_size);
10:   setUniverseRadius(univ_radius);
11:   setWindowSize(window_size);
12: }
13:
14: void Body::setName(sf::String name) {
15:   _planet_name = name;
16: }
17:
18: void Body::setFile(sf::String file) {
19:   _filename = file;
20: }
21:
22: void Body::setPosition(sf::Vector2f pos) {
23:   _pos = pos;
24: }
25:
26: void Body::setVelocity(sf::Vector2f velo) {
27:   _velo = velo;
28: }
29:
30: void Body::setMass(double mass) {
31:   _mass = mass;
32: }
33:
34: void Body::setAccel(sf::Vector2f accel) {
35:   _accel = accel;
36: }
37:
38: void Body::setUniverseSize(double univ_size) {
39:   _univ_size = univ_size;
40: }
41:
42: void Body::setUniverseRadius(double univ_radius) {
43:   _univ_radius = univ_radius;
44: }
45:
46: void Body::setWindowSize(sf::Vector2f window_size) {
47:   _window_size = window_size;
48: }
49:
50: void Body::setnormalPos(sf::Vector2f normalPos) {
51:   _normalPos = normalPos;
52: }
53:
54: void Body::setFnet(sf::Vector2f force) {
55:   _Fnet = force;
56: }
57:
58: sf::Vector2f Body::getVelocity() {
59:   return _velo;
60: }
61:
```

```
   62: double Body::getMass() {
   63:   return _mass;
   64: }
   65:
   66: sf::String Body::getName() {
   67:   return _planet_name;
   68: }
   69:
   70: sf::String Body::getFile() const {
   71:   return _filename;
   72: }
   73:
   74: double Body::getUniverseRadius() {
   75:   return _univ_radius;
   76: }
   77:
   78: double Body::getUniverseSize() {
   79:   return _univ_size;
   80: }
   81:
   82: sf::Vector2f Body::getWindowSize() {
   83:   return _window_size;
   84: }
   85:
   86: sf::Vector2f Body::getPos() const {
   87:   return _pos;
   88: }
   89:
   90: sf::Vector2f Body::getAccel() const {
   91:   return _accel;
   92: }
   93:
   94: sf::Vector2f Body::getnormalPos() const {
   95:   return _normalPos;
   96: }
   97:
   98: sf::Vector2f Body::getFnet() const {
   99:   return _Fnet;
  100: }
  101:
  102: std::istream& operator >> (std::istream& in, Body& body) {
  103:   sf::Vector2f pos;
  104:   sf::Vector2f vel;
  105:   sf::Vector2f accel;
  106:   double _mass;
  107:   in >> (pos.x) >> (pos.y) >> (vel.x) >> (vel.y) >> _mass;
  108:   body.setPosition(pos);
  109:   body.setVelocity(vel);
  110:   body.setMass(_mass);
  111:   return in;
  112: }
  113:
  114: void Body::setUpPos() {
  115:   sf::Vector2f velo;
  116:   sf::Vector2f pos;
  117:   (pos.x) = (((getWindowSize().x) / 2) +
  118:              ((((getPos()).x) / getUniverseRadius()) *
  119:               ((getWindowSize().x) / 2)));
  120:   (pos.y) = (((getWindowSize().y) / 2) - (
  121:                (((getPos()).y) / getUniverseRadius()) *
  122:                ((getWindowSize().y) / 2)));
```

```
  123:   setnormalPos(pos);
  124: }
  125:
  126: void Body::draw(sf::RenderTarget &target, sf::RenderStates state) const {
  127:   graphics(target, state);
  128: }
  129:
  130: void Body::graphics(sf::RenderTarget &target, sf::RenderStates state) const
{
  131:   sf::Sprite sprite;
  132:   sf::Image image;
  133:   sf::Texture texture;
  134:   image.loadFromFile(getFile());
  135:   texture.loadFromImage(image);
  136:   sprite.setTexture(texture);
  137:   sprite.setPosition((getnormalPos()).x, (getnormalPos()).y);
  138:   target.draw(sprite, state);
  139: }
  140:
  141: void Body::setNewVelo(double seconds) {
  142:   sf::Vector2f velo;
  143:   velo.x = ((getVelocity()).x + (seconds * (getAccel()).x));
  144:   velo.y = ((getVelocity()).y + (seconds * (getAccel()).y));
  145:   setVelocity(velo);
  146: }
  147:
  148: void Body::setNewPos(double seconds) {
  149:   sf::Vector2f pos;
  150:   pos.x = ((getPos()).x + (seconds * (getVelocity()).x));
  151:   pos.y = ((getPos()).y + (seconds * (getVelocity()).y));
  152:   setPosition(pos);
  153: }
  154:
  155: void Body::setNewAccel() {
  156:   sf::Vector2f accel;
  157:   accel.x = ((getFnet()).x / (getMass()));
  158:   accel.y = ((getFnet()).y / (getMass()));
  159:   setAccel(accel);
  160: }
  161:
  162: void Body::step(double seconds) {
  163:   setNewAccel();
  164:   setNewVelo(seconds);
  165:   setNewPos(seconds);
  166:   setUpPos();
  167: }
```

```
    1: // Copyright [2015] Justin Nguyen
    2:
    3: #ifndef N_BODY_HPP_
    4: #define N_BODY_HPP_
    5:
    6: #include <SFML/Graphics.hpp>
    7: #include <SFML/Window.hpp>
    8:
    9: class Body : public sf::Drawable {
   10:  private:
   11:   sf::Vector2f _window_size, _pos, _velo, _accel, _Fnet, _normalPos;
   12:   double _univ_size, _univ_radius, _mass;
   13:   sf::String _filename;
   14:   sf::String _planet_name;
   15:   virtual void draw( sf::RenderTarget &target, sf::RenderStates state ) cons
t ;
   16:
   17:  public:
   18:   Body(int univ_size, double univ_radius, sf::Vector2f window_size);
   19:   void setName(sf::String name) ;
   20:   void setFile(sf::String file);
   21:   void setPosition(sf::Vector2f pos);
   22:   void setVelocity(sf::Vector2f velo);
   23:   void setMass(double mass);
   24:   void setUniverseSize(double univ_size);
   25:   void setUniverseRadius(double univ_radius);
   26:   void setWindowSize(sf::Vector2f window_size);
   27:   void setnormalPos(sf::Vector2f normalPos);
   28:   void setAccel(sf::Vector2f accel);
   29:   void setFnet(sf::Vector2f force);
   30:   double getUniverseSize();
   31:   double getMass();
   32:   double getUniverseRadius();
   33:   sf::Vector2f getVelocity();
   34:   sf::Vector2f getAccel() const;
   35:   sf::Vector2f getWindowSize();
   36:   sf::Texture getTexture() const;
   37:   sf::Sprite getSprite() const;
   38:   sf::String getName();
   39:   sf::String getFile() const;
   40:   sf::Vector2f getPos() const;
   41:   sf::Vector2f getnormalPos() const;
   42:   sf::Vector2f getFnet() const;
   43:   friend std::istream& operator >> (std::istream& in, Body& body);
   44:   void step(double seconds);
   45:   void setNewAccel();
   46:   void setNewVelo(double seconds);
   47:   void setNewPos(double seconds);
   48:   void setUpPos();
   49:   void graphics( sf::RenderTarget &target, sf::RenderStates state ) const;
   50: };
   51:
   52: #endif
```

```
    1: 5
    2: 2.50e+11
    3:  1.4960e+11  0.0000e+00  0.0000e+00  2.9800e+04  5.9740e+24    earth.gif
    4:  2.2790e+11  0.0000e+00  0.0000e+00  2.4100e+04  6.4190e+23     mars.gif
    5:  5.7900e+10  0.0000e+00  0.0000e+00  4.7900e+04  3.3020e+23  mercury.gif
    6:  0.0000e+00  0.0000e+00  0.0000e+00  0.0000e+00  1.9890e+30      sun.gif
    7:  1.0820e+11  0.0000e+00  0.0000e+00  3.5000e+04  4.8690e+24    venus.gif
    8: [!] Your program should not read this line, as all 5 planets were already li
sted
    9: and there is no further information to be specified. So you can put comments
 here.
   10: This file contains the inner 4 planets of our solar system, and the sun.
```