```
    1: #include "CelestialBody.h"
    2: // constructor will initialize private data memberst
    3: // as well as load the image into new texture obj, then make a sprite with t
hat
    4: // texture
    5:
    6: CelestialBody::CelestialBody() { return; }
    7:
    8: CelestialBody::CelestialBody(double x, double y, double xv, double yv, doubl
e m,
    9:                                double rad, string name) {
   10:   xPos = x;
   11:   yPos = y;
   12:   xVel = xv;
   13:   yVel = yv;
   14:   mass = m;
   15:   radius = rad;
   16:   filename = name;
   17:
   18:   if (!img.loadFromFile(filename)) // check to make sure file opens
   19:   {
   20:     return;
   21:   }
   22:
   23:   tex.loadFromImage(img);
   24:
   25:   spr.setTexture(tex);
   26:
   27:   spr.setPosition(sf::Vector2f(xPos, yPos));
   28: }
   29:
   30: void CelestialBody::draw(sf::RenderTarget &target,
   31:                          sf::RenderStates states) const {
   32:   target.draw(spr);
   33: }
   34:
   35: void CelestialBody::setRadius(double rad) { radius = rad; }
   36:
   37: void CelestialBody::setPosition() {
   38:   double xPosScreen = ((xPos / radius) * (winWidth / 2)) + (winWidth / 2);
   39:   double yPosScreen = - ((yPos / radius) * (winHeight / 2)) + (winHeight / 2
);
   40:   spr.setPosition(sf::Vector2f(xPosScreen, yPosScreen));//only change the po
sition on screen
   41: }
   42:
   43: void CelestialBody::setVelocity(double x, double y) {
   44:   xVel = x;
   45:   yVel = y;
   46: }
   47:
   48: void CelestialBody::setForces(double x, double y) {
   49:   xForce = x;
   50:   yForce = y;
   51: }
   52:
   53: void CelestialBody::step(double seconds) {
   54:   xAccel = xForce / mass;
   55:   yAccel = yForce / mass;
   56:   xVel = xVel + (xAccel * seconds);
   57:   yVel = yVel + (yAccel * seconds);
```

```
   58:   xPos = xPos + (xVel * seconds);
   59:   yPos = yPos + (yVel * seconds);
   60: }
   61:
   62: // operator to read in universe file and set up the celestial bodies
   63: istream &operator>>(istream &input, CelestialBody &bod) {
   64:   // take input in order of the txt file
   65:   input >> bod.xPos >> bod.yPos;
   66:   input >> bod.xVel >> bod.yVel;
   67:   input >> bod.mass >> bod.filename;
   68:
   69:   if (!bod.img.loadFromFile(
   70:           bod.filename)) // exit with input if file does not load
   71:   {
   72:     return input;
   73:   }
   74:
   75:   bod.tex.loadFromImage(bod.img); // repeat same steps as in constructor
   76:   bod.spr.setTexture(bod.tex);
   77:   bod.spr.setPosition(sf::Vector2f(bod.xPos, bod.yPos));
   78:
   79:   return input;
   80: }
   81:
   82: ostream &operator<<(ostream &output, CelestialBody &bod) {
   83:   //output << "Filename: " << bod.filename << std::endl;
   84:   //output << "Pos X: " << bod.xPos << std::endl;
   85:   //output << "Pos Y: " << bod.yPos << std::endl;
   86:   //output << "Vel X: " << bod.xVel << std::endl;
   87:   //output << "Vel Y: " << bod.yVel << std::endl;
   88:
   89:   output << setw(12) << bod.xPos << " " << setw(12) << bod.yPos << " ";
   90:   output << setw(10) << bod.xVel << " " << setw(10) << bod.yVel << " ";
   91:   output << setw(12) << bod.mass << " " << setw(12) << bod.filename;
   92:   return output;
   93: }
   94: //finds x component of force between 2 bodies
   95: double getForceX(CelestialBody &bod1, CelestialBody &bod2) {
   96:   double distX = bod2.xPos - bod1.xPos;
   97:   double distY = bod2.yPos - bod1.yPos;
   98:   double r2 = pow(distX, 2) + pow(distY, 2);
   99:   double r = sqrt(r2);
  100:   double force = (gravity * bod1.mass * bod2.mass) / r2;
  101:   double forceX = force * (distX / r);
  102:   return forceX;
  103: }
  104: //finds y component of force
  105: double getForceY(CelestialBody &bod1, CelestialBody &bod2) {
  106:   double distX = bod2.xPos - bod1.xPos;
  107:   double distY = bod2.yPos - bod1.yPos;
  108:   double r2 = pow(distX, 2) + pow(distY, 2);
  109:   double r = sqrt(r2);
  110:   double force = (gravity * bod1.mass * bod2.mass) / r2;
  111:   double forceY = force * (distY / r);
  112:   return forceY;
  113: }
```