```
    1: all: NBody
    2:
    3: NBody: main.o space.o mf.o
    4:         g++ main.o space.o mf.o -o NBody -lsfml-graphics -lsfml-window -lsfm
l-system
    5:
    6: NBody.o: main.cpp space.hpp
    7:         g++ -c main.cpp -Wall -Werror -ansi -pedantic -g
    8:
    9: Space.o: space.cpp space.hpp
   10:         g++ -c space.cpp -Wall -Werror -ansi -pedantic -g
   11:
   12: mf.o: mf.cpp space.hpp
   13:         g++ -c mf.cpp -Wall -Werror -ansi -pedantic -g
   14:
   15: clean:
   16:         rm *.o NBody *~ *# *.gch
```

```
 1: #include <iostream>
 2: #include <string>
 3: #include <vector>
 4: #include <SFML/Graphics.hpp>
 5: #include <SFML/System.hpp>
 6: #include <SFML/Window.hpp>
 7: #include "space.hpp"
 8:
 9: using namespace std ;
10: using namespace sf ;
11:
12: int main( int argc, char *argv[] ) {
13:
14:   int n, i, ws ; double spcrdi, result; vector< body* > space ;
15:
16:   int et = atoi(argv[1]) ;   int ct = atoi(argv[2]) ;
17:
18:   cin >> n ; cin >> spcrdi ; ws = 900 ;
19:
20:   for( i = 0 ; i < n ; i++ ) {
21:     body *planet= new body( spcrdi, ws ) ;
22:     cin >> planet;
23:     space.push_back( planet )  ;
24:    }
25:   result = 0 ;
26:   RenderWindow window( VideoMode( ws, ws), "ps3 AC" ) ;
27:   int test_num = 0 ;
28:
29:   while (window.isOpen()){
30:     test_num++ ;
31:     sf::Event event;
32:     while (window.pollEvent(event)) {
33:       if (event.type == sf::Event::Closed)
34:         window.close();
35:     }
36:     window.clear();
37:     for( i = 0 ; i < n ; i++ ) window.draw(*space[i]) ;
38:     window.display();
39:     result = pl(space, result , ct ) ;
40:     if( result == -1 ) return 0 ;
41:     if( result > et ) return 0 ;
42:   }
43:
44:   return 0 ;
45: }
```

```
    1: // Copyright 2015 <Zheondre Calcano>
    2: #include <iostream>
    3: #include <string>
    4: #include <vector>
    5: #include <SFML/Graphics.hpp>
    6: #include <SFML/System.hpp>
    7: #include <SFML/Window.hpp>
    8: #include "space.hpp"
    9: #include <math.h>
   10:
   11: using namespace std;
   12:
   13: double pl( vector< body* > &space, double ot, double deltat) {
   14:   int i, j;
   15:   double x, y, F, Fx, Fy, ax, ay, Vx, Vy, r;
   16:   x = y = F = Fx = Fy = ax = ay = Vx = Vy = r = 0;
   17:
   18:   for( i = 0 ; (unsigned)i < space.size() ; i++ ) {
   19:
   20:     space[i]->setfnx(0.0);
   21:     space[i]->setfny(0.0);
   22:     for( j = 0 ; (unsigned)j < space.size() ; j++ ) {
   23:
   24:       if( i == j ) continue ;
   25:       else{
   26:          x = space[j]->getposx() - space[i]->getposx() ;
   27:          y = space[j]->getposy() - space[i]->getposy() ;
   28:
   29:          r = sqrt( x * x + y * y ) ;
   30:          if( r < 1 ) return -1 ;
   31:          F = space[i]->grav( r, space[j]->getmass(), space[i]->getmass() ) ;

   32:          Fx = (F*x)/r ; Fy = (F*y)/r ;
   33:
   34:          space[i]->setfnx(space[i]->getfnx() + Fx) ;
   35:          space[i]->setfny(space[i]->getfny() + Fy) ;
   36:       }
   37:     }
   38:   }
   39:
   40:   for (i = 0; (unsigned)i < space.size(); i++) {
   41:
   42:     if( space[i]->getmass() < 1 ) return -1; // return error message throw e
xception
   43:     ax = (space[i]->getfnx() / space[i]->getmass());
   44:     ay = (space[i]->getfny() / space[i]->getmass());
   45:
   46:     Vx = space[i]->getxvel() + deltat*ax;
   47:     Vy = space[i]->getyvel() + deltat*ay;
   48:     space[i]->setV( Vx, Vy );
   49:     space[i]->setNotScaledPos(space[i]->getposx()+deltat*Vx,
   50:                               space[i]->getposy() + deltat*Vy);
   51:     space[i]->setpos(space[i]->getnsx()/(1e+9), space[i]->getnsy()/(1e+9));
   52:
   53:   }
   54:   return deltat + ot ;
   55: }
```

```
    1: #include <iostream>
    2: #include <math.h>
    3: #include "space.hpp"
    4: #include <vector>
    5: using namespace sf ;
    6:
    7: body::body(double ss, double ws) {
    8:   ceny = cenx = ss/(1e+9) + 200;
    9:   winsiz = ws;
   10: }
   11: void body::center(double x) {
   12:   ceny = cenx = x/(1e+9) + 200;
   13: }
   14: void body::setpos(){
   15:   xpos = radfromsun + cenx;
   16:   ypos = ypos/(1e+9) + ceny;
   17: }
   18: void body::setNotScaledPos(double x, double y) {
   19:   nsx = x;
   20:   nsy = y;
   21: }
   22: void body::setpos(double x, double y) {
   23:   xpos = x;
   24:   ypos = y;
   25:   sprite.setPosition(xpos, ypos) ;
   26: }
   27: void body::setfnx( double val ){ fnx = val ; }
   28: void body::setfny( double val ){ fny = val ; }
   29:
   30: double body::gspx(){return xpos ; }
   31: double body::gspy(){ return ypos ; }
   32: double body::getfnx(){ return fnx ; }
   33: double body::getfny(){ return fny ; }
   34: double body::getposx(){ return xpos*(1e+9); }  //return the unscaled x & y
   35: double body::getposy(){ return ypos*(1e+9); }
   36: double body::getnsx(){ return nsx; }
   37: double body::getnsy(){ return nsy; }
   38: void body::setV(double x, double y ){ xvel = x; yvel = y; }
   39: double body::getxvel(){ return xvel; }
   40: double body::getyvel(){ return yvel; }
   41: double body::getmass(){ return mass; }
   42: void body::setrSun(){ radfromsun = xpos/(1e+9) ; }
   43: double body::getrad(){ return radfromsun ; }
   44: string body::getfname(){ return fname; }
   45: void body::setImage(){
   46:   texture.loadFromFile(fname) ;
   47:   sprite.setTexture(texture) ;
   48:   sprite.setPosition(xpos, ypos) ; // needs to be called every time position
 is changed.
   49: }
   50: void body::newpos(){
   51:
   52:   if( xpos > cenx && ypos <= ceny ) {
   53:     xpos = xpos - xvel;
   54:     ypos = ypos - yvel ;
   55:   }
   56:   if( xpos < cenx && ypos <= ceny ) {
   57:     xpos = xpos - xvel;
   58:     ypos = ypos + yvel ;
   59:   }
   60:   if( xpos >= cenx && ypos < ceny ) {
```

```
   61:     xpos = xpos + xvel;
   62:     ypos = ypos + yvel ;
   63:   }
   64:   if( xpos >= cenx && ypos > ceny ) {
   65:     xpos = xpos + xvel;
   66:     ypos = ypos - yvel ;
   67:   }
   68: }
   69:
   70: double body::grav(double r, double m2, double m1){
   71:
   72:   if(r < 1) return -1;
   73:   return ((6.67e-11)*m1*m2)/(r*r);
   74: }
   75:
```

```
 1: #ifndef _space
 2: #define _space
 3:
 4: #include <SFML/Graphics.hpp>
 5: #include <SFML/System.hpp>
 6: #include <SFML/Window.hpp>
 7: #include <iostream>
 8: #include <string>
 9: #include <vector>
10:
11: using namespace std;
12: using namespace sf ;
13:
14: class body : public Drawable {
15:   double nsx, nsy;
16:   double xpos, ypos, xvel, yvel, mass, radfromsun;
17:   double unir, cenx, ceny, winsiz, fnx, fny;
18:   string fname;
19:   Sprite sprite;
20:   Texture texture;
21:
22: public :
23:   body(double spacer, double winsize);
24:   body():xpos(0), ypos(0), xvel(0), yvel(0), mass(0), fname("") {};
25:
26:   void center(double);
27:   void setrSun();
28:   void setfnx(double);
29:   void setfny(double);
30:   void setnsx(double);
31:   void setnsy(double);
32:   void setNotScaledPos(double, double);
33:   void setV(double, double);
34:
35:   double grav(double, double, double);
36:   double getfnx();
37:   double getfny();
38:
39:   double gspx();
40:   double gspy();
41:
42:   double getnsx();
43:   double getnsy();
44:   double getposx();
45:   double getposy();
46:
47:   double getxvel();
48:   double getyvel();
49:
50:   double getmass();
51:   double getrad();
52:   string getfname();
53:
54:   void newpos();
55:   void setpos();
56:   void setpos(double, double);
57:   void setImage();
58:
59:   friend istream &operator >> (istream &input, body *S) {
60:     input >> S->xpos >> S->ypos >> S->xvel >> S->yvel >> S->mass >> S->fname
;
```

```
   61:     S->setrSun();
   62:     S->setpos();
   63:     S->setImage();
   64:     return input;
   65:   }
   66: private:
   67:   virtual void draw(RenderTarget& target, RenderStates states) const{
   68:     target.draw(sprite, states);
   69:   };
   70: };
   71:
   72: double pl(vector< body* > &x, double y, double z);
   73: #endif
```