Simulating Gravity

AiS Challenge
Final Report
April 7th, 2004

Team 28
Farmington High School

Team Members
Leonard Biemer
Michael Blount
Brian Geistwhite
Kasra Manavi
Amelia Symmonds

Teachers
Mr. Mike DeField
Mrs. Shirley Maurer

EXECUTIVE SUMMARY

The goal of our project was to create a program that could accurately and efficiently simulate gravity. Such a program would be very useful in investigating a wide array of physical phenomena. We first created an algorithm that closely approximates gravity. Then, we researched the equations necessary to describe the algorithm in mathematical terms, and created a skeleton C++ program that could model a system of two bodies. This program was not run, but instead acted as a model for a more powerful Java program. Upon completion, the Java program was tested using a scenario that included the Sun, the Earth, and the Moon. The program failed the test, outputting data that shows the Earth and the Moon flying past the sun and out of the solar system. Further, the data shows the Earth, Sun, and Moon accelerating in directions where there is no gravitational attraction. We believe that the program is close to working correctly, and that a detailed analysis of the program could point out the flaw and yield an accurately working program.

INTRODUCTION

Gravity, one of the most basic forces of astrophysics, stretches beyond all barriers, binding every atom to every other atom in the universe. On the astrophysical scale, it dominates every other force; in fact, in most cases astrophysical systems can be modeled accurately while completely neglecting the other forces. Thus, a program that can simulate gravity would be extremely useful in modeling everything from moons orbiting Jupiter to galaxy clusters orbiting each other. The goal of our project was to create such a program that would operate accurately and efficiently.

DESCRIPTION

The model for the program is a follows: A number of astronomical bodies (stars, planets, etc.) are created as points in the coordinate plane. In addition to their coordinates, each body has the properties of mass and velocity. A length of time is chosen that will be the length of each time step, and then the actual simulation begins, in which each body undergoes the following procedure: First, the acceleration due to gravity on the body from each other body is calculated. The body's accelerations are then summed to find a resultant acceleration. This resultant acceleration is multiplied by the length of the time step to obtain a velocity change. This velocity change is then added to the body's previous velocity to result in a new velocity. The new velocity is then multiplied by the time step to obtain a position change. The position change is then added to the body's previous coordinates. Thus, we have the program moving through a single time step, with each body's position and velocity changed slightly. The program then repeats these time steps a specified number of times, at which point the simulation is completed.

These numbers are calculated and simulated by system of basic mathematical and physical equations. Our program was a compilation of the following equations.

$F = G * M_1 * M_2 / D^2$

$F * M = A$

$A = G * M_2 / D^2$

$A^2 = A_x^2 + A_y^2 + A_z^2$

$A_x / A = (X_2 – X_1) / D$

$A_y / A = (Y_2 – Y_1) / D$

$A_z / A = (Z_2 – Z_1) / D$

$A * T = \Delta V$

$V * T = \Delta P$

F – Force on body
G – Gravitational Constant = $6.67300 * 10^{-11}$ m^3 kg^{-1} s^{-2}
M_1 – Mass of current body
M_2 – Mass of attracting body
D – Distance between current body and attracting body
A – Acceleration
A_x – Acceleration's X component
A_y – Acceleration's Y component
A_z – Acceleration's Z component
X_1 – Current body's X coordinate
X_2 – Attracting body's X coordinate
Y_1 – Current body's Y coordinate
Y_2 – Attracting body's Y coordinate
Z_1 – Current body's Z coordinate
Z_2 – Attracting body's Z coordinate
T = Time

The original draft of the program was written in C++. It can be seen in Appendix A.

The mathematical formulas were stripped from the C++ program, and laid the foundation for the Java program. The Java program uses inheritance and methods, to simplify the process of having multiple bodies. To see the Java program, please refer to Appendix A.

The Java program was tested with a scenario that modeled the Sun, the Earth, and the Moon. The Sun was set at the origin and had no initial velocity. The Earth was set above sun on the positive Y axis and was given a velocity in the positive X direction. The moon was set above the Earth on the positive X axis. It was given its own velocity plus the Earth's velocity in the positive X direction.

The following information was take from NASA's Goddard Space Flight Center site.

	Sun	Earth	Moon
Mass (kg)	1.9891 E30	5.9736 E24	7.349 E22
Orbital Radius (m)	N/A	1.496 E11	3.844 E8
Mean Orbital Velocity (m/s)	N/A	2.978 E4	1.023 E3

Number of time steps: 365 (1 year)
Length of time steps: 86400 seconds (1 day)

RESULTS

The actual output of the test scenario can be seen in Appendix B. However, through analysis of the data, what was happening became quite clear. The Earth and the Moon both maintained their velocity in the X direction, and accelerated quite rapidly in the negative Y direction toward the sun. After passing the sun, slightly to its right side due to their x velocity, they continued to accelerate in the negative Y direction, even though there was no body to accelerate them in that direction anymore. Likewise, the Sun moved and accelerated in the positive X and Y directions during the entire test, even after the Moon and Earth had long left that region of space.

CONCLUSION

A program that could successfully model gravity was not created. The output obtained in the test scenario was nowhere near tracing the orbits of the Earth and Moon. Even if the information obtained about the Sun, Moon, and Earth were faulty, this would not explain the acceleration of the bodies in a direction where there clearly could have been no gravity pulling. There is clearly a significant problem either in the program model or in the way that the model is implemented in the program.

RECOMMENDATIONS

Clearly, there is room to expand. First of all, the program needs to be debugged such that it will correctly model gravity. Afterwards, several improvements can be made. The current output displays the coordinates of the bodies, which is effective but difficult to interpret and tedious. A graphic out put that illustrates the behavior of the bodies would be much easier to analyze. Once finalized there will be many ways to apply the program. One application is to model solar systems and how they will change over periods of time. A simpler application would be to use it to model a planet, such as Earth, and show how an asteroid or other body coming towards it would behave.

APPENDIX A

C++ PROGRAM

```
#include <iostream>
#include <stdlib.h>
#include <cmath>
#include <fstream>
//Programmed by Brian Geistwhite and Michael Blount.
using namespace std;

int main()
{
ofstream of;
of.open("Outputs.txt");
cout << "Program is now running…." << endl;
of << fixed << showpoint;
double G=(6.67*(10^(-11)));
double o1, o2;                                  //o=object, x,y=x,y coordinate
double o1x, o1y, o1z, o2x, o2y, o2z;
double o1v, o2v;                                //v=velocity, a=acceleration, m = mass
double o1xv, o1yv, o1zv, o2xv, o2yv, o2zv;
double o1a, o2a;
double o1xa, o1ya, o1za, o2xa, o2ya, o2za;
double o1m, o2m;
double d1_2, d2_1;                              //d=distance between

double t=0;                     //how long each each step of the program takes (seconds)
o1x=0,  o1y=0,  o1z=0,  o2x=0,  o2y=0,  o2z=0;      //the starting
o1xv=0, o1yv=0, o1zv=0, o2xv=0, o2yv=0, o2zv=0;     //parameters of each object
o1m=0,  o2m=0;
int iterations = 0;

for (iterations = 0;iterations < 0;iterations++)
{

 /* The line of code below is better seen in this diagram
                (o1x-o2x)    (o1y-o2y)    (o1z-o2z)
               (       ^2)+(       ^2)+(       ^2)
              (                                  )
       (                                        ) */
 d1_2= sqrt((((o1x-o2x)* (o1x-o2x))+((o1y-o2y) * (o1y-o2y))+((o1z-o2z)*(o1z-o2z))));
 o1a=(G*o2m)/((d1_2) * (d1_2));          //calculates the x and y component
 o1xa=(o1a)*((o1x-o2x)/(d1_2));          //acceleration vectors for object 1
 o1ya=(o1a)*((o1y-o2y)/(d1_2));          //Notice, it will have to be changed to take
 o1za=(o1a)*((o1z-o2z)/(d1_2));          //into account multiple attracting bodies
```

```
 d2_1= sqrt((((o2x-o1x)*(o2x-o1x))+((o2y-o1y)*(o2y-o1y))+((o2z-o1z)*(o2z-o1z))));
 o2a=(G*o1m)/((d2_1)*(d2_1));
 o2xa=(o2a)*((o2x-o1x)/(d2_1));
 o2ya=(o2a)*((o2y-o1y)/(d2_1));          //calculates the x and y component
 o2za=(o2a)*((o2z-o1z)/(d2_1));          //acceleration vectors for object 2

 o1xv=o1xv+o1xa*t;       // acceleration*time=velocity,
 o1yv=o1yv+o1ya*t;       //velocity initial + velocity added from gravity
 o1zv=o1zv+o1za*t;       //= new velocity
 o2xv=o2xv+o2xa*t;
 o2yv=o2yv+o2ya*t;
 o2zv=o2zv+o2za*t;

 o1x=o1x+o1xv*t;         //same as above, but this time from velocity to position
 o1y=o1y+o1yv*t;
 o1z=o1z+o1zv*t;
 o2x=o2x+o2xv*t;
 o2y=o2y+o2yv*t;
 o2z=o2z+o2zv*t;
 of << "Step: " << (iterations + 1) << endl;
 of << "Sun: \nX = " << o1x << "\nY = " << o1y << "\nZ = " << o1z;
 of << "\nEarth: \nX = " << o2x << "\nY = " << o2y << "\nZ = " << o2z << endl << endl;
}

    system("PAUSE");
    of.close();
    return 0;
}
```

JAVA PROGRAM

Body.java:

```
public class Body{
public String name;
public double x;
public double y;
public double z;
public double xvel;
public double yvel;
public double zvel;
public double mass;
public double gravity;
public long time;
//Programmed by Michael Blount
```

```
 public Body() { }
 public Body(String nm, double x1, double y1, double z1, double xvel1,
           double yvel1, double zvel1, double mass1) {
  name = nm;
  x = x1;
  y = y1;
  z = z1;
  xvel = xvel1;
  yvel = yvel1;
  zvel = zvel1;
  mass = mass1;
 }
  public String getName()
  {
    return name;
  }
  public void setTime(int tim)
  {
        time = tim;
  }
  public double getX()
  {
    return x;
  }
  public void setX()
  {
    x = x + xvel * time;
  }
  public double getY()
  {
    return y;
  }
  public void setY()
  {
    y = y + yvel * time;
  }
  public double getZ()
  {
    return z;
  }
  public void setZ()
  {
    z = z + zvel * time;
  }
  public double getvelX()
{
```

```
  return xvel;
}
public void setvelX(double xaccel)
{
  xvel = xvel + xaccel * time;
}
public double getvelY()
{
  return yvel;
}
public void setvelY(double yaccel)
{
  yvel = yvel + yaccel * time;
}
public double getvelZ()
{
  return zvel;
}
public void setvelZ(double zaccel)
{
  zvel = zvel + zaccel * time;
}
public double getMass()
{
  return mass;
}
public double getGravity()
{
  return gravity;
}
public void setGravity(double grav)
    {
      gravity = grav;
    }
    public String toString()
    {
      return ("Name: " + name + "\n" + "X Position: " + x + "\n" +
                    "Y Position: " + y + "\n" + "Z Position: " + z);
    }
  }
```

Main.java:

```
import TerminalIO.KeyboardReader;
import java.io.FileOutputStream;
import java.io.BufferedOutputStream;
import java.io.PrintStream;
```

```
import java.io.*;
import java.*;
//Programmed By Michael Blount
public class Main
{
   public static void main(String [] args){
     Body[] bodarray;
     KeyboardReader reader = new KeyboardReader();
     int numBody = 0;
     String name = "";
     double x = 0;
     double steps = 0;
     int time = 0;
     double cur = 0;
     final double G = (6.672 * Math.pow(10, -11));
     double[][] distance;
     double[][] accel;
     double[][] xaccel, yaccel, zaccel;
   double y = 0;
   double z = 0;
   double xvel = 0;
   double yvel = 0;
   double zvel = 0;
   double mass = 0;
   double[] xac;
   double[] yac;
   double[] zac;
   File dstFile = new File("outputfile.txt");
   try{
         PrintWriter out = new PrintWriter(new BufferedWriter(new FileWriter(dstFile)));


   numBody = reader.readInt("Enter number of bodies: ");
   bodarray = new Body[numBody];
   distance = new double[numBody][numBody];
   accel = new double[numBody][numBody];
   xaccel = new double[numBody][numBody];
   yaccel = new double[numBody][numBody];
   zaccel = new double[numBody][numBody];
   xac = new double[numBody];
   yac = new double[numBody];
   zac = new double[numBody];
   for (int i = 0; i < numBody; i++)
   {
     name = reader.readLine("Body " + (i + 1) + " Name: ");
     x = reader.readDouble("Body " + (i + 1) + " X Cordinate: ");
```

```
  y = reader.readDouble("Body " + (i + 1) + " Y Cordinate: ");
  z = reader.readDouble("Body " + (i + 1) + " Z Cordinate: ");
  xvel = reader.readDouble ("Body " + (i + 1) + " X Velocity: ");
  yvel = reader.readDouble ("Body " + (i + 1) + " Y Velocity: ");
  zvel = reader.readDouble ("Body " + (i + 1) + " Z Velocity: ");
  mass = reader.readDouble ("Body " + (i + 1) + " Mass: ");
  bodarray[i] = new Body(name, x, y, z, xvel, yvel, zvel, mass);
  System.out.println();
}

steps = reader.readInt("Number of Time Steps: ");
time = reader.readInt("Enter length of time steps: ");
for (int g = 0; g < numBody; g++)
{
	bodarray[g].setTime(time);
}
out.println("Step: 0");
for (int v = 0; v < numBody; v++)
{
	out.println(bodarray[v]);
	}
for(int s = 1; s <= steps; s++)
{
	out.println();
	out.println("Step: " + s);
	for(int w = 0; w < numBody; w++)
	{
		for (int q = 0; q < numBody; q++)
		{
			distance[w][q] = Math.sqrt(Math.pow((bodarray[q].getX() -
			bodarray[w].getX()),2) + Math.pow((bodarray[q].getY() -
			bodarray[w].getY()),2) + Math.pow((bodarray[q].getZ() -
			bodarray[w].getZ()),2));
		}
	}
	for (int w = 0; w < numBody; w++)
	{
		for (int q = 0; q < numBody; q++)
		{
			if(w != q){
	//Distance to Accel.
	accel[w][q] = G * bodarray[q].getMass() / Math.pow(distance[w][q],2);
	//Separate Accel to Comps.
		xaccel[w][q] = (accel[w][q] * (bodarray[q].getX() - bodarray[w].getX())) /
			distance[w][q];
		yaccel[w][q] = (accel[w][q] * (bodarray[q].getY() - bodarray[w].getY())) /
```

```
                    distance[w][q];
                zaccel[w][q] = (accel[w][q] * (bodarray[q].getZ() - bodarray[w].getZ())) /
                    distance[w][q];
        }
}
}
                //Set New Velocities.
                for(int e = 0; e < numBody; e++)
                {
                    for(int u = 0; u < numBody; u++)
                    {
                        xac[e] += xaccel[e][u];
                        yac[e] += yaccel[e][u];
                        zac[e] += zaccel[e][u];
                    }
                }
                for (int v = 0; v < numBody; v++)
                {
                bodarray[v].setvelX(xac[v]);
                bodarray[v].setvelY(yac[v]);
                bodarray[v].setvelZ(zac[v]);
                //Set New Coords.
                bodarray[v].setX();
                bodarray[v].setY();
                bodarray[v].setZ();
                //Show new Coords
                out.println(bodarray[v]);
        }
        }
                out.close();
 }

 catch (Exception e) {
            System.err.println(e); // print the exception to warn.
          }

        }
}
```

APPENDIX B

This is only the first ten steps, and the last five steps. This is representative of the overall data. The entire output file would be greater than 100 pages.

Step: 0

Name: Sun
X Position: 0.0
Y Position: 0.0
Z Position: 0.0

Name: Earth
X Position: 0.0
Y Position: 1.496E11
Z Position: 0.0

Name: Moon
X Position: 0.0
Y Position: 1.499844E11
Z Position: 0.0

Step: 1

Name: Sun
X Position: 0.0
Y Position: 134.56736472930618
Z Position: 0.0

Name: Earth
X Position: 2.572992E9
Y Position: 1.495559810306567E11
Z Position: 0.0

Name: Moon
X Position: 8.83872E7
Y Position: 1.4992022487530225E11
Z Position: 0.0

Step: 2

Name: Sun
X Position: 2.2884224432434803
Y Position: 538.2900846957089
Z Position: 0.0

Name: Earth
X Position: 5.145216573624697E9
Y Position: 1.4942367084938217E11

Z Position: 0.0

Name: Moon
X Position: 1.7721523703438085E8
Y Position: 1.4974772843464813E11
Z Position: 0.0

Step: 3

Name: Sun
X Position: 11.447491093929031
Y Position: 1345.8179649387303
Z Position: 0.0

Name: Earth
X Position: 7.715147093417442E9
Y Position: 1.4915877713380875E11
Z Position: 0.0

Name: Moon
X Position: 2.6655160988748395E8
Y Position: 1.4942272357355093E11
Z Position: 0.0

Step: 4

Name: Sun
X Position: 34.36945885466203
Y Position: 2691.9834012671795
Z Position: 0.0

Name: Earth
X Position: 1.0280488879990656E10
Y Position: 1.4871694905486087E11
Z Position: 0.0

Name: Moon
X Position: 3.563706897809796E8
Y Position: 1.4890083682708325E11
Z Position: 0.0

Step: 5

Name: Sun
X Position: 80.29135349040331
Y Position: 4712.002410459961
Z Position: 0.0

Name: Earth
X Position: 1.2838167115529636E10
Y Position: 1.480537118593704E11
Z Position: 0.0

Name: Moon
X Position: 4.4659572845821476E8
Y Position: 1.4813738465395248E11
Z Position: 0.0

Step: 6

Name: Sun
X Position: 160.85637811339336
Y Position: 7541.757943104693
Z Position: 0.0

Name: Earth
X Position: 1.5384306261455685E10
Y Position: 1.4712437438996405E11
Z Position: 0.0

Name: Moon
X Position: 5.371100017820476E8
Y Position: 1.4708722241775812E11
Z Position: 0.0

Step: 7

Name: Sun
X Position: 290.2110367515787
Y Position: 11318.169658448081
Z Position: 0.0

Name: Earth
X Position: 1.7914197680672897E10
Y Position: 1.458839082375317E11
Z Position: 0.0

Name: Moon
X Position: 6.277597935202211E8
Y Position: 1.4570455899661542E11
Z Position: 0.0

Step: 8

Name: Sun
X Position: 485.1486723574615
Y Position: 16179.657372722875
Z Position: 0.0

Name: Earth
X Position: 2.042225184677165E10
Y Position: 1.442867962051194E11
Z Position: 0.0

Name: Moon
X Position: 7.183540095611552E8
Y Position: 1.4394273050776352E11
Z Position: 0.0

Step: 9

Name: Sun
X Position: 765.312701763178
Y Position: 22266.70948838321
Z Position: 0.0

Name: Earth
X Position: 2.2901930701008373E10
Y Position: 1.422868464275959E11
Z Position: 0.0

Name: Moon
X Position: 8.086616972547472E8
Y Position: 1.4175392424506775E11

Z Position: 0.0

Step: 10

Name: Sun
X Position: 1153.476845253696
Y Position: 29722.57318561556
Z Position: 0.0

Name: Earth
X Position: 2.5345654394364983E10
Y Position: 1.3983696671542432E11
Z Position: 0.0

Name: Moon
X Position: 8.984077045477442E8
Y Position: 1.390888401310828E11
Z Position: 0.0

…….

Step: 361

Name: Sun
X Position: 4.222629736796239E8
Y Position: 1.7591089443201065E8
Z Position: 0.0

Name: Earth
X Position: -1.3864231161690973E14
Y Position: -5.1154744647922234E13
Z Position: 0.0

Name: Moon
X Position: -8.407462701908761E13
Y Position: -5.908518787906925E14
Z Position: 0.0

Step: 362

Name: Sun

X Position: 4.247656719127483E8
Y Position: 1.7688869569084302E8
Z Position: 0.0

Name: Earth
X Position: -1.3946695459929895E14
Y Position: -5.1437475451316766E13
Z Position: 0.0

Name: Moon
X Position: -8.457337807801992E13
Y Position: -5.94335669194637E14
Z Position: 0.0

Step: 363

Name: Sun
X Position: 4.272757651356814E8
Y Position: 1.778691996325837E8
Z Position: 0.0

Name: Earth
X Position: -1.402940418331094E14
Y Position: -5.172098022720944E13
Z Position: 0.0

Name: Moon
X Position: -8.507360408509052E13
Y Position: -5.978296991476091E14
Z Position: 0.0

Step: 364

Name: Sun
X Position: 4.297932533482983E8
Y Position: 1.7885240625718656E8
Z Position: 0.0

Name: Earth
X Position: -1.411235733182995E14
Y Position: -5.200525897558492E13
Z Position: 0.0

Name: Moon
X Position: -8.557530504029902E13
Y Position: -6.013339686496062E14
Z Position: 0.0

Step: 365

Name: Sun
X Position: 4.323181365504757E8
Y Position: 1.7983831556460604E8
Z Position: 0.0

Name: Earth
X Position: -1.4195554905482816E14
Y Position: -5.229031169642808E13
Z Position: 0.0

Name: Moon
X Position: -8.607848094364503E13
Y Position: -6.048484777006256E14
Z Position: 0.0

WORKS CITED

Serway, Raymond A. and Jerry S. Faughn. College Physics Fort Worth, TX: Saunders College Publishing, 1999.

Weisstein, Eric W. http://scienceworld.wolfram.com/physics/GravitationalConstant.html. v. 3/3/04

Williams, Dr. David R. http://nssdc.gsfc.nasa.gov/planetary/factsheet/earthfact.html. 27 Oct., 2003. v. 2/19/04

ACKNOWLEDGEMENTS

We would like to thank all the people who helped out with the project, and contributed in many ways. First of all, we would like to thank our sponsors, Mr. DeField and Mrs. Maurer. They helped out whenever possible, and were ready and willing to do anything they could to help us out in any and all dimensions of the project. We would also like to thank the evaluators that judged our project and provided pointers to direct our success in the project. Their info was quite useful and provided helpful tips to further our project. Lastly, we would like to thank all the people who take the time to put on the Supercomputing Challenge, and all the hard work they do, as well as the judges who review the projects. They don't have to do this and we all appreciate it very much. Thanks!