Callum McLaughlin

coursework two

Table of Contents

[Computer Graphics and Modelling Report 3](#_Toc70001814)

[**Introduction** 3](#_Toc70001815)

[**Research** 3](#_Toc70001816)

[**The Logic** 3](#_Toc70001817)

[The Planet Class 3](#_Toc70001818)

[Creating the Planets 4](#_Toc70001819)

[Creating the Background 4](#_Toc70001820)

[The Animation Loop & Orbiting the Planes 4](#_Toc70001821)

[Custom Camera 4](#_Toc70001822)

[Implementing the Lighting 5](#_Toc70001823)

[Frame Counter and Show/Clear functions 5](#_Toc70001824)

[**Testing & Evaluation** 6](#_Toc70001825)

[**Bibliography** 7](#_Toc70001826)

[**Appendix** 8](#_Toc70001827)

[CourseworkTwoMain.java 8](#_Toc70001828)

[Planet.java 15](#_Toc70001829)

[StdDraw3D.java 18](#_Toc70001830)

# Computer Graphics and Modelling Report

## **Introduction**

The goal of coursework two was to create a Our Solar System using the StdDraw libraries that were used in the class exercises. The guidelines asked that the solar system include all of the main planets and moons as well as the background being stars. The planets should orbit the sun as accurately as possible and be in the correct order in distance, from the sun. The sun should also have a light source to give light to the planets and moons. The guidelines also asked that a camera be programmed just outside the earths orbit but unfortunately, I could not get this to work properly so it is not included in the programme.

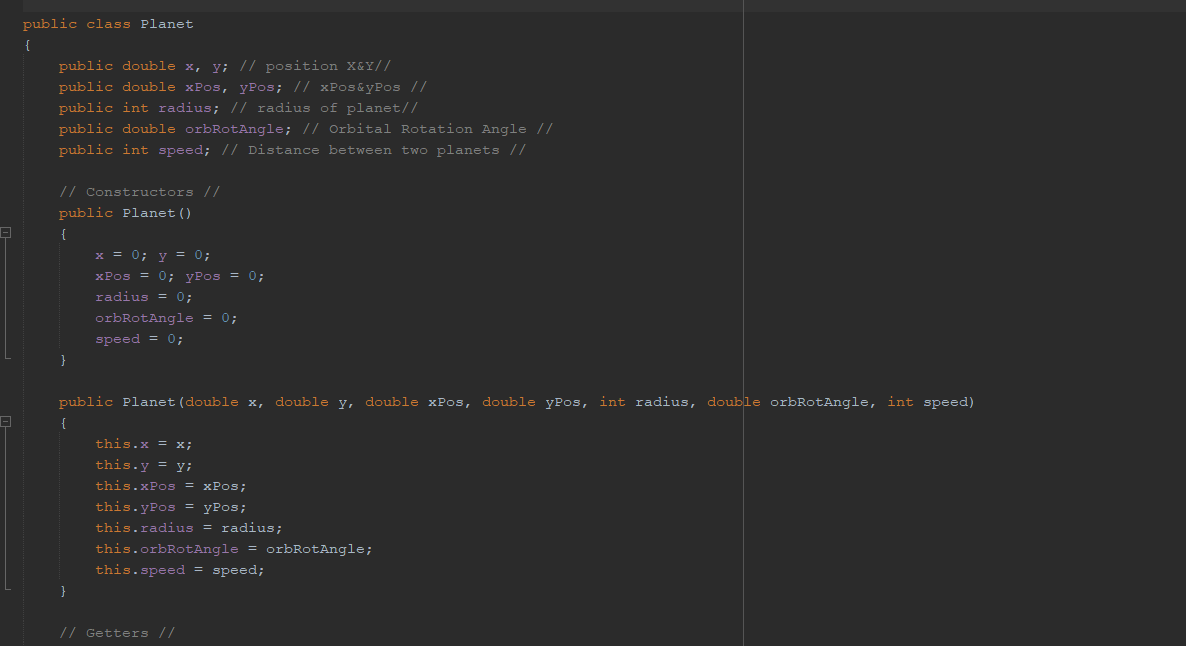
## **Research**

Research was conducted before beginning on the project, examples of how others did similar programmes were researched and the final project takes inspiration from multiple different solutions. The main research that was used was a video titled “Coding Challenge #7: Solar System in Processing - Part 1 (2D)”, (Shiffman, 2016). A lot of ideas were inspired using this video and the mathematics and logic used were a lot of help to figuring out the final solution. Other sources used were Stackoverflow (GigaNova, 2016)and UCL.ac.uk (Sherman IP , 2010).

## **The Logic**

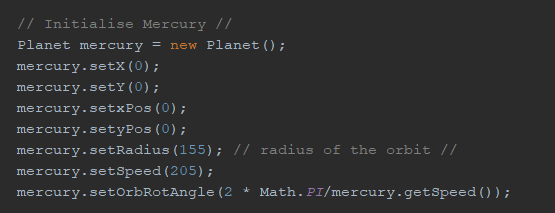
### The Planet Class

To start the programme a class was created called “Planet”. This class would be used to instantiate each planet with each of the required attribute like radius, xPos, yPos etc. The class variables and constructors can be seen below:



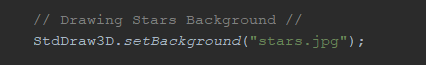
### Creating the Planets

Using this class each planet was created, an example of Mercury being created is shown below:



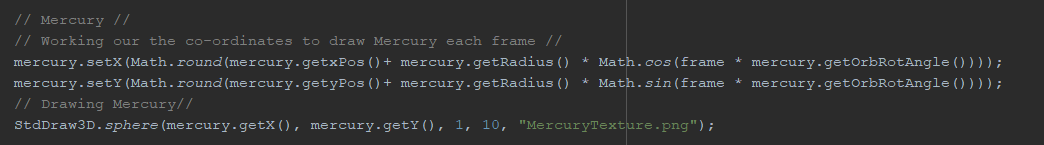
### Creating the Background

After each planet had been created, the background was set using a image texture for stars in space as seen below:



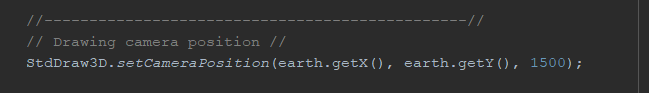
### The Animation Loop & Orbiting the Planes

Next the animation loop was created which would move the planets around the sun frame by frame by redrawing them each iteration of the loop. This would be done by working out the orbital angle and angle the planets around the sun be defining a radius for the complete rotation which would be edited to change the distance between planets. The formulas shown below would work out where the next coordinates would be using the formulas for finding orbital angle. An example of how Mercury was moved is shown below:



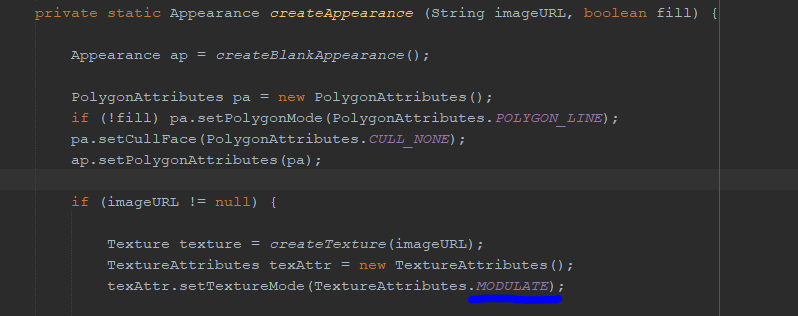
### Custom Camera

The custom camera was achieved by setting the cameras X and Y coordinates to the same as the earths so that it would follow the earth, by then changing the z coordinate, a camera following the earths orbit of the sun was achieved. The code, which is placed inside the animation loop, can be seen below:



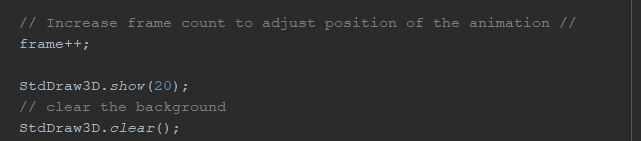
### Implementing the Lighting

The next part of the programme was to get the point light working correctly so that the planets would have a dark side and a light side, only receiving light from the side facing the sun. In order to achieve this the StdDraw3D library would have to be edited. Within the library I edited the line shown below using MODULATE instead of REPLACE. The code can be seen below:



### Frame Counter and Show/Clear functions

The last part of the programme would increase the frame counter which was used to adjust the position of the planets in the animation loop. The other 2 lines of code is showing the animation and clearing the animation after on iteration of the loop so when the new planets are drawn there are no duplicates, giving the impression of a moving object rather than one that is being deleted and redrawn.



## **Testing & Evaluation**

While testing the programme it was discovered that the code for the earth’s moon orbiting the earth is not working as intended, as it is still orbiting the sun but following closely with the earth. I have been able to fix this as I was using the wrong variable assignment which was affecting the calculation. I have tested that each of the planets have the correct textures and the light source is also tested and working as intended, placed inside the sun.

I tested the earths moon by increasing its speed to check if it moved away from the earth, but it did not, it kept orbiting the earth even with the speed increased to a very high number.

The only other issue I had was setting up the camera perspective from the earth’s orbit, but I was able to create it after some research, so that it follows the orbit of the earth around the sun with the other planets in view, I had the same error as before where I used the wrong X and Y variables in the calculation.

The only thing I was unable to do was drawing the rings around Saturn as I prioritised the more important features of the programme and left that to be added in if I had time but with other assignments to complete, I decided it should not be too much of a problem to leave it out.

# **Bibliography**

GigaNova, 2016. *Stack Overflow.* [Online]   
Available at: https://stackoverflow.com/questions/35103994/java-planetary-orbit-simulation-centering-planets  
[Accessed 12 04 2021].

Sherman IP , 2010. *UCL.ac.uk.* [Online]   
Available at: https://www.ucl.ac.uk/~zcapg66/work/COMP4/simulations/orbit/orbit.html  
[Accessed 25 03 2021].

Shiffman, D., 2016. *Youtube.* [Online]   
Available at: https://www.youtube.com/watch?v=l8SiJ-RmeHU&t=89s  
[Accessed 19 04 2021].

## **Appendix**

### CourseworkTwoMain.java

import java.awt.Color;

public class CourseworkTwoMain

{

public static void main (String[] args)

{

StdDraw3D.setScale(-500,500);

int frame = 0;

// Initialise Mercury //

Planet mercury = new Planet();

mercury.setX(0);

mercury.setY(0);

mercury.setxPos(0);

mercury.setyPos(0);

mercury.setRadius(155); // radius of the orbit //

mercury.setSpeed(205);

mercury.setOrbRotAngle(2 \* Math.PI/mercury.getSpeed());

// Initialise Venus //

Planet venus = new Planet();

venus.setX(0);

venus.setY(0);

venus.setxPos(0);

venus.setyPos(0);

venus.setRadius(205); // radius of the orbit //

venus.setSpeed(200);

venus.setOrbRotAngle(2 \* Math.PI/venus.getSpeed());

// Initialise Earth //

Planet earth = new Planet();

earth.setX(0);

earth.setY(0);

earth.setxPos(0);

earth.setyPos(0);

earth.setRadius(260); // radius of the orbit //

earth.setSpeed(195);

earth.setOrbRotAngle(2 \* Math.PI/earth.getSpeed());

// Initialise Mars //

Planet mars = new Planet();

mars.setX(0);

mars.setY(0);

mars.setxPos(0);

mars.setyPos(0);

mars.setRadius(300); // radius of the orbit //

mars.setSpeed(200);

mars.setOrbRotAngle(2 \* Math.PI/mars.getSpeed());

// Initialise Jupiter //

Planet jupiter = new Planet();

jupiter.setX(0);

jupiter.setY(0);

jupiter.setxPos(0);

jupiter.setyPos(0);

jupiter.setRadius(405); // radius of the orbit //

jupiter.setSpeed(275);

jupiter.setOrbRotAngle(2 \* Math.PI/jupiter.getSpeed());

// Initialise Saturn //

Planet saturn = new Planet();

saturn.setX(0);

saturn.setY(0);

saturn.setxPos(0);

saturn.setyPos(0);

saturn.setRadius(530); // radius of the orbit //

saturn.setSpeed(225);

saturn.setOrbRotAngle(2 \* Math.PI/saturn.getSpeed());

// Initialise Uranus //

Planet uranus = new Planet();

uranus.setX(0);

uranus.setY(0);

uranus.setxPos(0);

uranus.setyPos(0);

uranus.setRadius(620); // radius of the orbit //

uranus.setSpeed(215);

uranus.setOrbRotAngle(2 \* Math.PI/uranus.getSpeed());

// Initialise Neptune //

Planet neptune = new Planet();

neptune.setX(0);

neptune.setY(0);

neptune.setxPos(0);

neptune.setyPos(0);

neptune.setRadius(700); // radius of the orbit //

neptune.setSpeed(245);

neptune.setOrbRotAngle(2 \* Math.PI/neptune.getSpeed());

// Initialise Earths Moon //

Planet moon = new Planet();

moon.setX(earth.getX());

moon.setY(earth.getY());

moon.setxPos(earth.getxPos());

moon.setyPos(earth.getyPos());

moon.setRadius(20); // radius of the orbit //

moon.setSpeed(3000);

moon.setOrbRotAngle(2 \* Math.PI/moon.getSpeed());

// Drawing Stars Background //

StdDraw3D.setBackground("stars.jpg");

while (true)

{

// Loop must be used if doing animations //

// Clears Drawings //

StdDraw3D.clear();

// Sun //

// Drawing the Sun //

StdDraw3D.sphere(0, 0, 0, 110, "sunTexture.jpg");

StdDraw3D.pointLight(0, 0, 0, Color.WHITE);

// Mercury //

// Working our the co-ordinates to draw Mercury each frame //

mercury.setX(Math.round(mercury.getxPos()+ mercury.getRadius() \* Math.cos(frame \* mercury.getOrbRotAngle())));

mercury.setY(Math.round(mercury.getyPos()+ mercury.getRadius() \* Math.sin(frame \* mercury.getOrbRotAngle())));

// Drawing Mercury//

StdDraw3D.sphere(mercury.getX(), mercury.getY(), 1, 10, "MercuryTexture.png");

// Venus //

// Working our the co-ordinates to draw Venus each frame //

venus.setX(Math.round(venus.getxPos()+ venus.getRadius() \* Math.cos(frame \* venus.getOrbRotAngle())));

venus.setY(Math.round(venus.getyPos()+ venus.getRadius() \* Math.sin(frame \* venus.getOrbRotAngle())));

// Drawing Venus //

StdDraw3D.sphere(venus.getX(), venus.getY(), 1, 15, "VenusTexture.jpg");

// Earth //

// Working our the co-ordinates to draw Earth each frame //

earth.setX(Math.round(earth.getxPos()+ earth.getRadius() \* Math.cos(frame \* earth.getOrbRotAngle())));

earth.setY(Math.round(earth.getyPos()+ earth.getRadius() \* Math.sin(frame \* earth.getOrbRotAngle())));

// Drawing Earth //

StdDraw3D.sphere(earth.getX(), earth.getY(), 1, 15, "EarthTexture.jpg");

//-----------------------------------------------//

// Drawing camera position //

StdDraw3D.setCameraPosition(earth.getX(), earth.getY(), 1500);

//------------------------------------------------------------------------------------------------//

// Earths Moon //

// Working our the co-ordinates to draw Earths Moon each frame //

moon.setX(Math.round(earth.getX()+ moon.getRadius() \* Math.cos(frame \* moon.getOrbRotAngle())));

moon.setY(Math.round(earth.getY()+ moon.getRadius() \* Math.sin(frame \* moon.getOrbRotAngle())));

// Drawing Earth //

StdDraw3D.sphere(moon.getX(), moon.getY(), 1, 10, "EarthTexture.jpg");

// Mars //

// Working our the co-ordinates to draw Mars each frame //

mars.setX(Math.round(mars.getxPos()+ mars.getRadius() \* Math.cos(frame \* mars.getOrbRotAngle())));

mars.setY(Math.round(mars.getyPos()+ mars.getRadius() \* Math.sin(frame \* mars.getOrbRotAngle())));

// Drawing Mars //

StdDraw3D.sphere(mars.getX(), mars.getY(), 1, 12, "MarsTexture.jpg");

// Jupiter //

// Working our the co-ordinates to draw Jupiter each frame //

jupiter.setX(Math.round(jupiter.getxPos()+ jupiter.getRadius() \* Math.cos(frame \* jupiter.getOrbRotAngle())));

jupiter.setY(Math.round(jupiter.getyPos()+ jupiter.getRadius() \* Math.sin(frame \* jupiter.getOrbRotAngle())));

// Drawing Jupiter //

StdDraw3D.sphere(jupiter.getX(), jupiter.getY(), 1, 50, "JupiterTexture.jpg");

// Saturn //

// Working our the co-ordinates to draw Saturn each frame //

saturn.setX(Math.round(saturn.getxPos()+ saturn.getRadius() \* Math.cos(frame \* saturn.getOrbRotAngle())));

saturn.setY(Math.round(saturn.getyPos()+ saturn.getRadius() \* Math.sin(frame \* saturn.getOrbRotAngle())));

// Drawing Saturn //

StdDraw3D.sphere(saturn.getX(), saturn.getY(), 1, 50, "SaturnTexture.jpg");

StdDraw3D.setPenColor(StdDraw3D.WHITE);

// StdDraw.circle(saturn.getX(), saturn.getY(), 300);

// Uranus //

// Working our the co-ordinates to draw Uranus each frame //

uranus.setX(Math.round(uranus.getxPos()+ uranus.getRadius() \* Math.cos(frame \* uranus.getOrbRotAngle())));

uranus.setY(Math.round(uranus.getyPos()+ uranus.getRadius() \* Math.sin(frame \* uranus.getOrbRotAngle())));

// Drawing Uranus //

StdDraw3D.sphere(uranus.getX(), uranus.getY(), 1, 30, "UranusTexture.jpg");

StdDraw3D.setPenColor(StdDraw3D.WHITE);

// StdDraw.circle(uranus.getX(), uranus.getY(), 300);

// Neptune //

// Working our the co-ordinates to draw Neptune each frame //

neptune.setX(Math.round(neptune.getxPos()+ neptune.getRadius() \* Math.cos(frame \* neptune.getOrbRotAngle())));

neptune.setY(Math.round(neptune.getyPos()+ neptune.getRadius() \* Math.sin(frame \* neptune.getOrbRotAngle())));

// Drawing Neptune //

StdDraw3D.sphere(neptune.getX(), neptune.getY(), 1, 30, "NeptuneTexture.jpg");

StdDraw3D.setPenColor(StdDraw3D.WHITE);

// Increase frame count to adjust position of the animation //

frame++;

StdDraw3D.show(20);

// clear the background

StdDraw3D.clear();

}

}

}

### Planet.java

public class Planet

{

public double x, y; // position X&Y//

public double xPos, yPos; // xPos&yPos //

public int radius; // radius of planet//

public double orbRotAngle; // Orbital Rotation Angle //

public int speed; // Distance between two planets //

// Constructors //

public Planet()

{

x = 0; y = 0;

xPos = 0; yPos = 0;

radius = 0;

orbRotAngle = 0;

speed = 0;

}

public Planet(double x, double y, double xPos, double yPos, int radius, double orbRotAngle, int speed)

{

this.x = x;

this.y = y;

this.xPos = xPos;

this.yPos = yPos;

this.radius = radius;

this.orbRotAngle = orbRotAngle;

this.speed = speed;

}

// Getters //

public double getX() {

return x;

}

public double getY() {

return y;

}

public double getxPos() {

return xPos;

}

public double getyPos() {

return yPos;

}

public int getRadius() {

return radius;

}

public double getOrbRotAngle() {

return orbRotAngle;

}

public int getSpeed() {

return speed;

}

// Setters //

public void setX(double x) {

this.x = x;

}

public void setY(double y) {

this.y = y;

}

public void setxPos(double xPos) {

this.xPos = xPos;

}

public void setyPos(double yPos) {

this.yPos = yPos;

}

public void setRadius(int radius) {

this.radius = radius;

}

public void setOrbRotAngle(double orbRotAngle) {

this.orbRotAngle = orbRotAngle;

}

public void setSpeed(int speed) {

this.speed = speed;

}

}

### StdDraw3D.java

The StandardDraw3D libraries were used to complete this coursework. Due to the length of the class I have not pasted it in the appendix as it is on Blackboard. The only different in my code is what I have mentioned above in the report in relation to the “.MODULATE” method.