Callum McLaughlin

24678805  COMPUTER GRAPHICS AND MODELLING

Assignment one

Table of Contents

[Computer Graphics and Modelling 2](#_Toc65817792)

[Report 2](#_Toc65817793)

[Portfolio 1 2](#_Toc65817794)

[Portfolio 2 5](#_Toc65817795)

[Portfolio 3 6](#_Toc65817796)

[Portfolio 4 9](#_Toc65817797)

[Portfolio 5 11](#_Toc65817798)

[Appendices 14](#_Toc65817799)

[Portfolio 1 14](#_Toc65817800)

[Portfolio 2 16](#_Toc65817801)

[Portfolio 3 17](#_Toc65817802)

[Portfolio 4 22](#_Toc65817803)

[Portfolio 5 28](#_Toc65817804)

[References 32](#_Toc65817805)

# Computer Graphics and Modelling

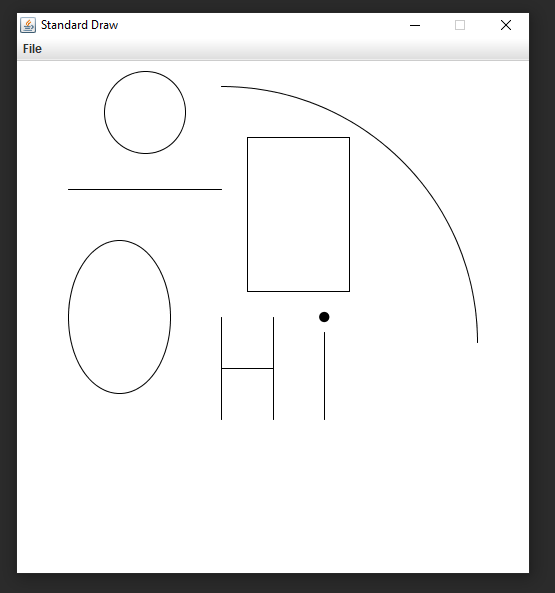
## Report

### Portfolio 1

#### Task 1

The first task was to draw the graphics primitives such as lines, rectangles, ellipses, curves, and circles. This was a simple task to complete and only required knowledge of the imported libraries and how to use them. For extra reading on the topic I looked online and found a useful website that explained the “Standard Libraries” in java and how they can be used. I didn’t have any problems with this exercise, and I found it very straightforward and easy to complete.

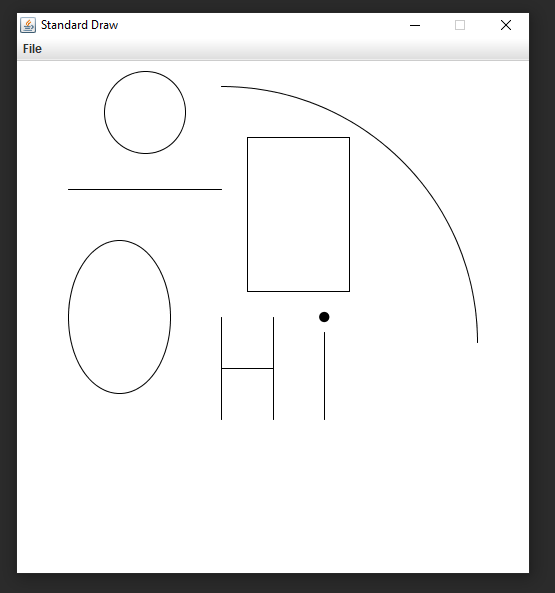
##### Output



#### Task 2

This next task was to create a word on screen using the graphics primitive. I was able to use lines and a filled circle to create the word “Hi” on screen as can be seen in the output picture below. This task was straight forward like the previous one and was not difficult at all, it just took some time to position each element so it would form the letters.

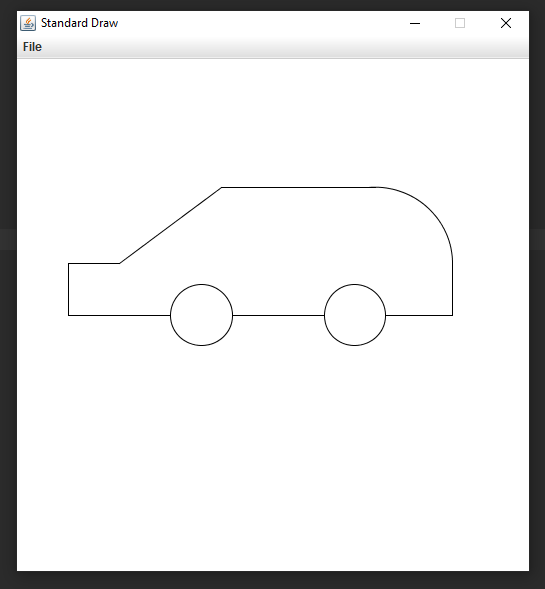
##### Output



#### Task 3

The third task was to write a graphics programme that would draw a car as shown in the exercise sheet. I was able to create the car using lines, arcs, and circles and again, found it pretty straight forward to do it just took a bit of time to get everything placed right. I didn’t encounter any problems when attempt the solution for this task.

##### Output

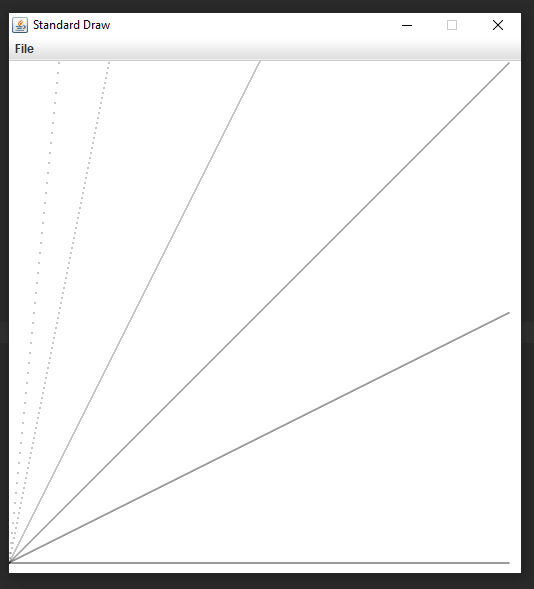


### Portfolio 2

#### Task 1

The first task of the second portfolio was to implement a brute force line drawing algorithm for which the pseudo code was shown on the exercise sheet. The method was to be implemented in its given general form and the method should be executed to draw six different lines for the given values shown on the exercise sheet. With a bit of help from my lecturer I was able to complete this task relatively easily without much difficulty and I did not feel the need to look into more research on the topic.

##### Output



#### Task 2

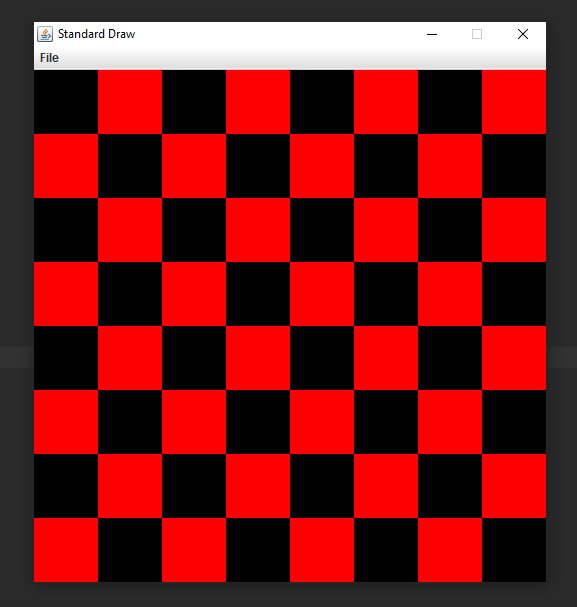
The second task asked the explain any observations made about these line drawings in task 1 for increasing values of m. To put it simply, when m is increased, the line becomes steeper towards the left side of the screen. Increasing values of m move the endpoint of the line higher which in turn makes the lines come closer together.

### Portfolio 3

#### Task 1

The first task of portfolio 3 was to create a checkboard and the number of red and black squares would be determines by the variable N. The lower left square must be considered as always red. This exercise took some thinking and planning ahead as it was a bit more complex than the tasks before it. The solution was to create a nested for loop, one to handles the rows and one to handles the columns of the checkboard, to start all squares would be coloured red and then if statements would be used to work out which squares should be coloured black by stating that if the row/2 leaves no remainder (if it is even) then paint the square black, otherwise keep it red, the same idea for the columns with another if statement. In the end all the odd columns become black as well as the even rows and the rest are red.

##### Output



#### Task 2

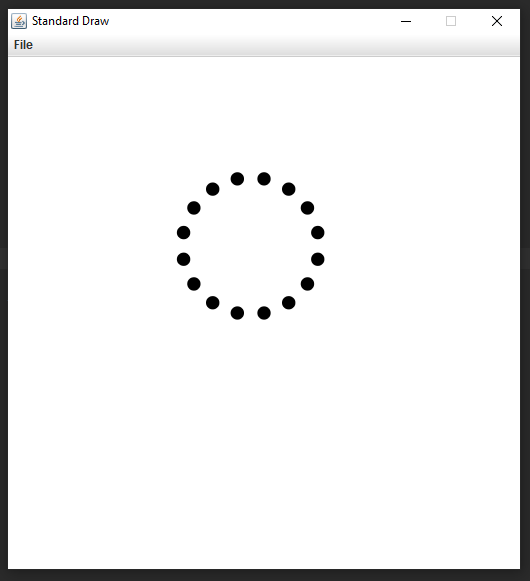
The next task was to write a programme that considers a variable N = 16 equally spaced points on the circumference of a circle with a radius of 200 and a centre position x = 200 and y =200. In order to find N equally spaced points, first the angle between any two consecutive points needed to be found which was done using the formula (2\*Math.PI)/N. I then used a for loop run from 1 to N to generate the N points with coordinates and the formulas used were:

and

.

I was able to complete this exercise after a while but there were a few times I needed to look around for solutions and ask for help from my lecturer.

##### Output

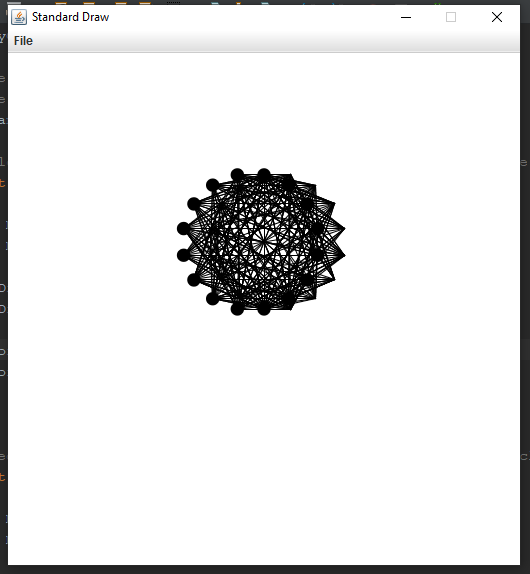


#### Task 3

The next task was to extend the previous programme by drawing a grey line between some pairs of points and what would determine which lines would be drawn and which would not, a probability variable was used, P. A line would be drawn between two points, dependant on the value of P.

In order to do this a double nested for loop was required to check each circle against the 15 others, generate a random number and if that number was <= p, draw the line between the two points. For this to work 2 arrays were also needed to store the x and y coordinates of each circle. The output is seen below. Although It seems at first glance like the programme is working as intended, for some reason there is also smaller circles being drawn inside the bigger one and the lines are offset for some reason. I haven’t been able to figure out why this is happening, so I decided to leave it as is in order to focus on the other tasks.

##### Output



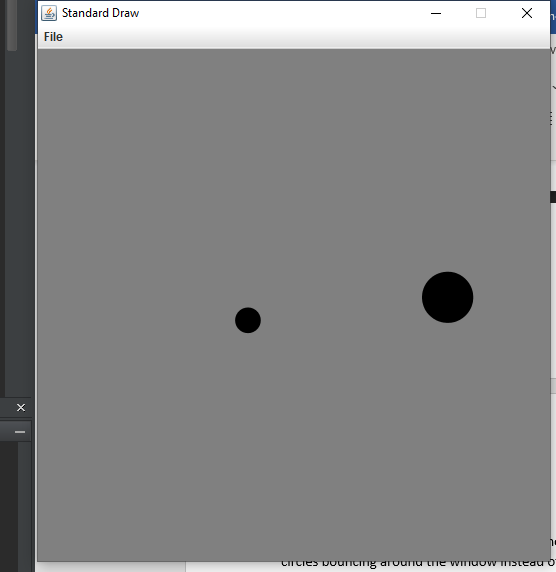
### Portfolio 4

#### Task 1

The next task was to take the code given at the start of exercise two and update it to have two circles bouncing around the window instead of 1. Different initial positions were required for the second circle and when the balls hit the side of the screen, they should have collision detection, allowing them to seemingly bounce off the side of the screen when they come into contact with it.

This exercise was quite simple. After reading through the code and understanding it I was able to add the second circle by simply copying what was needed for the first circle and just changing the initial starting position of it and the radius to make it a bit bigger.

##### Output

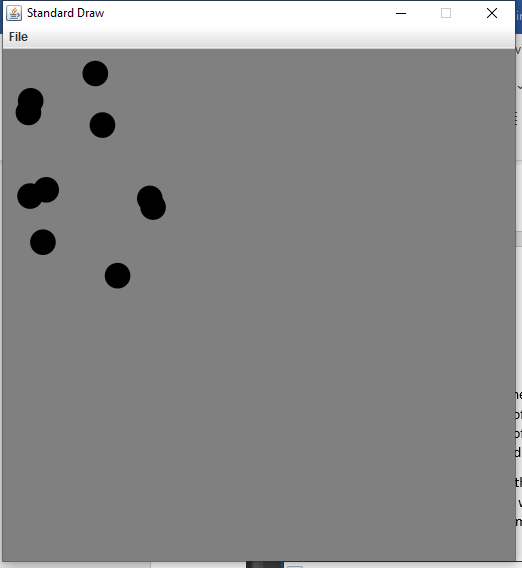


#### Task 2

The next task was to extend the previous programme with a variable number of ball (e.g. n = 10). In order to complete this exercise I created an array of structures using a class I created, “Ball” and creating an array of Balls which would store the position, velocity and other information about the ball. After creating the array of balls I had to create the balls so I used a for loop that would run to the numBalls and it would instantiate a ball with each iteration, also setting the x and y coordinates of that ball using Math.random() each time to ensure their starting positions are different.

After that it was simple enough to create the main part of the code, a for loop inside the animation loop, which would update the position of each ball in the array, make sure they bounced off the edge of the window when came into contact with it and also to redraw the balls on the screen within the loop with each frame..

##### Output

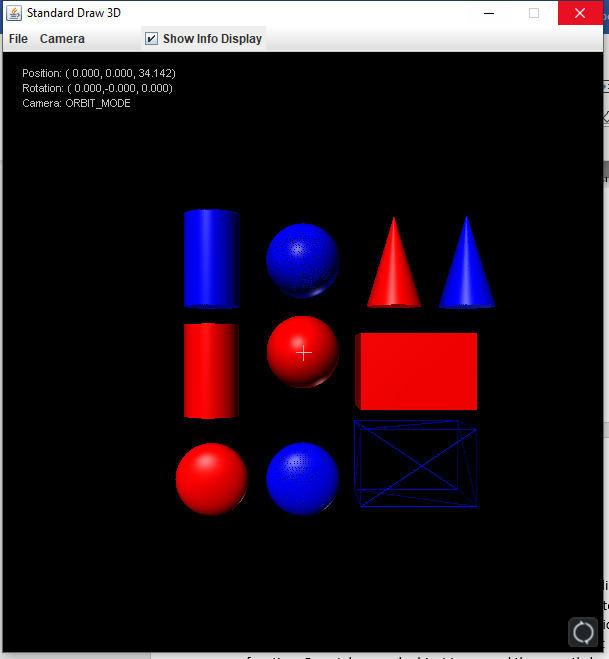


### Portfolio 5

#### Task 1

The first task of portfolio 5 was to use the Java3D imported libraries to display the primitive 3D objects such as spheres, cones, cylinders, boxes, ellipsoids etc. The task asked that the programme show the filled version of the object and the wireframe version. This task was quite easy to complete as it only required basic knowledge of the libraries and what each parameter was for each draw function. Once I drew each object, I arranged them neatly beside one another as seen below.

##### Output

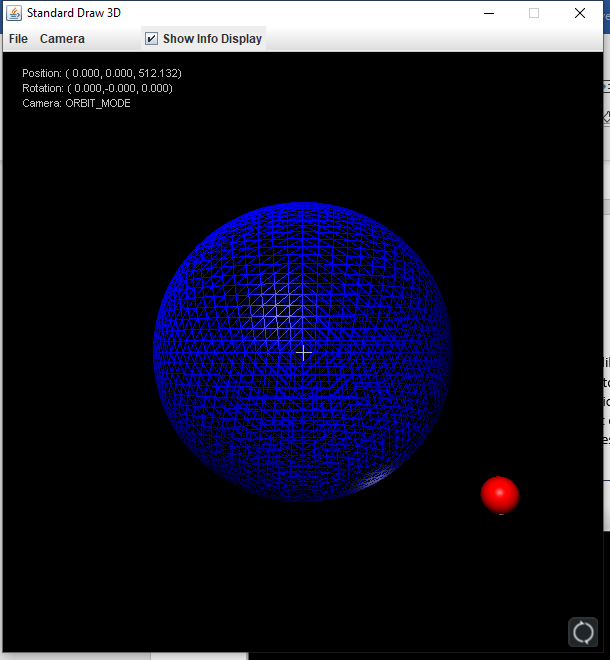


#### Task 2

The second task was to create a programme that would display the sun as a wireframe sphere in the middle and a planet as a solid sphere should be rotating around the sun in a circular orbit. This exercise took some time but one of the previous exercises that I had completed had code that was similar to what was needed here so I looked back at that which helped me along with creating this solution for this task.

First, I created the animation loop and the basic variables such as x and y for the coordinates, r for the radius of the circle, n was used to set how fast or slow the animation would draw the outer circle and also the angle at which the sphere would move. I also needed two more variables xPos and yPos for the centre position of where the outer circle would orbit, this would the centre point of the wireframe sphere. Inside the animation loop I clear the screen so that the circle is redrawn each frame without the last being there, I draw the blue wireframe sphere (sun), use the formulas from a previous task to work out the x and y coordinates which would act as the centre point for the next circle to be drawn, set the pen colour to red and draw the next circle in the next frame. The I increment the frame each time.

##### Output



## Appendices

### Portfolio 1

#### **Task 1 & 2**

public class Portfolio1

{

public static void main(String[] args)

{

StdDraw.setPenColor(StdDraw.BLACK);

// Draw Circle //

StdDraw.circle(0.25, 0.9, 0.08);

// Draw Line //

StdDraw.line(0.1, 0.75, 0.4, 0.75);

// Draw Rectangle //

StdDraw.rectangle( 0.55, 0.70, 0.1, 0.15);

// Draw Ellipsis //

StdDraw.ellipse(0.20, 0.50, 0.1, 0.15);

// Draw Curve //

StdDraw.arc(0.40, 0.45, 0.5, 0, 90);

// Draw "Hi" //

StdDraw.line(0.4, 0.3, 0.4, 0.5);

StdDraw.line(0.4, 0.4, 0.5, 0.4);

StdDraw.line(0.5, 0.3, 0.5, 0.5);

StdDraw.line(0.6, 0.3, 0.6, 0.47);

StdDraw.filledCircle(0.6, 0.5, 0.010);

}

}

#### **Task 3**

public class Portfolio1Car

{

public static void main(String[] args)

{

StdDraw.setPenColor(StdDraw.BLACK);

// Draw Lines //

StdDraw.line(0.1, 0.6, 0.1, 0.50);

StdDraw.line(0.1, 0.5, 0.3, 0.50);

StdDraw.line(0.2, 0.6, 0.1, 0.6);

StdDraw.line(0.4, 0.75, 0.2, 0.6);

StdDraw.line(0.7, 0.75, 0.4, 0.75);

StdDraw.line(0.85, 0.6, 0.85, 0.5);

StdDraw.line(0.85, 0.5, 0.72, 0.5);

StdDraw.line(0.60, 0.5, 0.42, 0.5);

// Draw Curve //

StdDraw.arc(0.7, 0.6, 0.15, 0, 95);

// Draw Circles //

StdDraw.circle(0.36, 0.5, 0.06);

// Draw Circles //

StdDraw.circle(0.66, 0.5, 0.06);

}

}

### Portfolio 2

#### **Task 1**

public class Portfolio2

{

public static void bruteForceLine(float x1, float x2,float m, float c)

{

// for each column of pixels (x) //

for (int x = Math.round(x1); x <= x2; x++)

{

// Compute y

float y = m \* x + c;

// Draw One pixel

// Have to round the y because it could be a decimal and we need an int //

StdDraw.point(x, Math.round(y));

}

}

public static void main(String[] args)

{

StdDraw.setScale(0, 512);

bruteForceLine(0, 500, 0, 10);

bruteForceLine(0, 500, 0.5f, 10);

bruteForceLine(0, 500, 1, 10);

bruteForceLine(0, 500, 2, 10);

bruteForceLine(0, 500, 5, 10);

bruteForceLine(0, 500, 10, 10);

}

}

### Portfolio 3

#### **Task 1**

public class Portfolio3 {

public static void checkerboard(int N)

{

StdDraw.setScale(0, N);

// for each row number r //

for (int r = 0; r < N; r++)

{

// for each column number c

for (int c = 0; c < N; c++)

{

// Find coords (x,y) // need to add 0.5 to get the centre point of the square //

double x = c + 0.5;

double y = r + 0.5;

// choose the colour //

// default red

StdDraw.setPenColor(StdDraw.RED);

//on even rows

if (r % 2 == 0)

{

// odd colums are black

if (c%2 != 0)

{

StdDraw.setPenColor(StdDraw.BLACK);

}

}

if (r%2 != 0)

{

//even rows are black

if (c%2 == 0)

{

StdDraw.setPenColor(StdDraw.BLACK);

}

}

// Draw Element at (x,y)

StdDraw.filledSquare(x, y, 0.5);

}

}

}

public static void main(String[] args)

{

checkerboard(8);

}

}

#### **Task 2**

public class Portfolio3b

{

public static void main (String[] args)

{

StdDraw.setScale(-1920,1920);

//total number of equally spaced points

int n = 16;

// radius of circle

int r = 200;

// Coordinates //

double x = 0, y = 0, dotPosX = 0, dotPosY = 0;

//centre the circle at x = 200, y = 200

// angle between two points//

double angle = 2 \* Math.PI/n;

// for loop to draw circle at co-ordinate then increment the angle and draw again until n circles are drawn forming the larger circle //

for (int i = 1; i <= n; i++)

{

x = Math.round(x + r \* Math.cos(i \* angle));

y = Math.round(y + r \* Math.sin(i \* angle));

StdDraw.setPenColor(StdDraw.BLACK);

StdDraw.filledCircle(x, y, 50);

}

}

}

#### **Task 3**

public class Portfolio3c

{

public static void main (String[] args)

{

StdDraw.setScale(-1920,1920);

//total number of equally spaced points

int n = 16;

// radius of circle

int r = 200;

// Coordinates //

double x = 0, y = 0;

double p = 0.1;

// Arrays for x and y coordinates - tried 2d array and was too problematic //

double xCoord [] = new double [17];

double yCoord [] = new double [17];

//centre the circle at x = 200, y = 200

// angle between two points//

double angle = 2 \* Math.PI/n;

// for loop to draw circle at co-ordinate then increment the angle and draw again until n circles are drawn forming the larger circle //

for (int i = 1; i <= n; i++)

{

x = Math.round(x + r \* Math.cos(i \* angle));

y = Math.round(y + r \* Math.sin(i \* angle));

StdDraw.setPenColor(StdDraw.BLACK);

StdDraw.filledCircle(x, y, 50);

xCoord[i] = x;

yCoord[i] = y;

}

// Nested for loop that checks the second loop againast every circle //

for (int c = 0; c <= 16; c++)

{

x = Math.round(x + r \* Math.cos(c \* angle));

y = Math.round(y + r \* Math.sin(c \* angle));

// nested for loop that does probability and draws line from one circle to each other circle if Mat.random <= proability //

for (int i = 0; i <= 16; i++)

{

for (int j = 0; j <= 15; j++)

{

if (Math.random() <= p)

{

StdDraw.line(xCoord[j+1], yCoord [j+1], x, y);

}

}

}

}

}

}

### Portfolio 4

#### **Task 1**

public class Portfolio4a

{

public static void main(String[] args)

{

// set the scale of the coordinate system

StdDraw.setXscale(-1.0, 1.0);

StdDraw.setYscale(-1.0, 1.0);

// initial values for first circle

double rx = 0.480, ry = 0.860; // position

double vx = 0.015, vy = 0.023; // velocity

double radius = 0.05; // radius

// initial values for second circle

double rx2 = 0.880, ry2 = 0.450; // position

double vx2 = 0.035, vy2 = 0.015; // velocity

double radius2 = 0.1; // radius

// main animation loop

while (true) {

// bounce off wall according to law of elastic collision

// first circle //

if (Math.abs(rx + vx) > 1.0 - radius) vx = -vx;

if (Math.abs(ry + vy) > 1.0 - radius) vy = -vy;

// second circle //

if (Math.abs(rx2 + vx2) > 1.0 - radius2) vx2 = -vx2;

if (Math.abs(ry2 + vy2) > 1.0 - radius2) vy2 = -vy2;

// update position

rx = rx + vx;

ry = ry + vy;

// update position

rx2 = rx2 + vx2;

ry2 = ry2 + vy2;

// clear the background

StdDraw.clear(StdDraw.GRAY);

// draw first ball on the screen

StdDraw.setPenColor(StdDraw.BLACK);

StdDraw.filledCircle(rx, ry, radius);

// draw second ball on the screen

StdDraw.setPenColor(StdDraw.BLACK);

StdDraw.filledCircle(rx2, ry2, radius2);

// display and pause for 20 ms

StdDraw.show(20);

}

}

}

#### **Task 2**

public class Portfolio4b

{

public static void main(String[] args)

{

// set the scale of the coordinate system

StdDraw.setXscale(-1.0, 1.0);

StdDraw.setYscale(-1.0, 1.0);

int numBalls = 10;

// Array of Balls //

Ball[] ballArray = new Ball[numBalls];

// for loop to initialise each ball //

for (int i = 0; i < numBalls; i++)

{

// Instantiated new ball //

ballArray[i] = new Ball();

// Setting random starting position for ball when initialised //

ballArray[i].setRx(Math.random());

ballArray[i].setRy(Math.random());

}

// main animation loop

while (numBalls > 0)

{

for (int i = 0; i < numBalls; i++)

{

if (Math.abs(ballArray[i].getRx() + ballArray[i].getVx()) > 1.0 - ballArray[i].radius) ballArray[i].vx = -ballArray[i].vx;

if (Math.abs(ballArray[i].getRy() + ballArray[i].getVy()) > 1.0 - ballArray[i].radius) ballArray[i].vy = -ballArray[i].vy;

// Update position //

ballArray[i].setRx(ballArray[i].getRx() + ballArray[i].getVx());

ballArray[i].setRy(ballArray[i].getRy() + ballArray[i].getVy());

// draw balls on the screen

StdDraw.setPenColor(StdDraw.BLACK);

StdDraw.filledCircle(ballArray[i].getRx(), ballArray[i].getRy(), ballArray[i].radius);

}

// display and pause for 20 ms

StdDraw.show(20);

// clear the background

StdDraw.clear(StdDraw.GRAY);

}

}

}

#### **Ball Class**

public class Ball

{

public double rx, ry; // position X&Y//

public double vx; // velocity X&Y//

public double vy; // velocity X&Y//

public double radius; // radius of ball//

// Constructors //

public Ball()

{

rx = 0.50; ry = 0.750; // position of ball X&Y//

vx = 0.015; vy = 0.025; // velocity of ball X&Y//

radius = 0.05; // radius of ball//

}

public Ball(double rx, double ry, double vx, double vy, double radius)

{

rx = this.rx;

ry = this.ry;

vx = this.vx;

vy = this.vy;

radius = this.radius;

}

// Getters //

public double getRx() {

return rx;

}

public double getRy() {

return ry;

}

public double getVx() {

return vx;

}

public double getVy() {

return vy;

}

public double getRadius() {

return radius;

}

// Setters //

public void setRx(double rx) {

this.rx = rx;

}

public void setRy(double ry) {

this.ry = ry;

}

public void setVx(double vx) {

this.vx = vx;

}

public void setVy(double vy) {

this.vy = vy;

}

public void setRadius(double radius) {

this.radius = radius;

}

}

### Portfolio 5

#### **Task 1**

public class Portfolio5a

{

public static void main (String[] args)

{

StdDraw3D.setScale(-10,10);

while (true)

{

// Loop must be used if doing animations //

// Clears Drawings //

StdDraw3D.clear();

// Drawing Red Sphere //

StdDraw3D.setPenColor(StdDraw.RED);

StdDraw3D.sphere(0, 0, 0, 2);

// Drawing Blue Wireframe Sphere //

StdDraw3D.setPenColor(StdDraw.BLUE);

StdDraw3D.wireSphere(0, 5, 0, 2);

// Drawing Red Rectangle //

StdDraw3D.setPenColor(StdDraw.RED);

StdDraw3D.box(6, -1, 0, 3, 2, 2);

// Drawing Blue Wireframe Box //

StdDraw3D.setPenColor(StdDraw.BLUE);

StdDraw3D.wireBox(6, -6, 0, 3, 2, 2);

// Drawing Red Cone //

StdDraw3D.setPenColor(StdDraw.RED);

StdDraw3D.cone(5, 5, 0, 1.5, 5);

// Drawing Blue Wireframe Cone //

StdDraw3D.setPenColor(StdDraw.BLUE);

StdDraw3D.wireCone(9, 5, 0, 1.5, 5);

// Drawing Red Cylinder //

StdDraw3D.setPenColor(StdDraw.RED);

StdDraw3D.cylinder(-5, -1, 0, 1.5, 5);

// Drawing Blue Wireframe Cylinder //

StdDraw3D.setPenColor(StdDraw.BLUE);

StdDraw3D.wireCylinder(-5, 5, 0, 1.5, 5);

// Drawing Red Ellipsoid //

StdDraw3D.setPenColor(StdDraw.RED);

StdDraw3D.ellipsoid(-5, -7, 0, 2, 2, 1);

// Drawing Blue Wireframe Ellipsoid //

StdDraw3D.setPenColor(StdDraw.BLUE);

StdDraw3D.wireEllipsoid(0, -7, 0, 2, 2, 1);

StdDraw3D.show(20);

}

}

}

#### **Task 2**

public class Portfolio5b

{

public static void main (String[] args)

{

StdDraw3D.setScale(-150,150);

int frame = 0;

int n = 200;

// radius of circle

int r = 200;

// Coordinates //

double x = 0, y = 0;

double xPos = 0, yPos = 0;

//centre the circle at x = 200, y = 200

// angle between two points//

double angle = 2 \* Math.PI/n;

while (true)

{

// Loop must be used if doing animations //

// Clears Drawings //

StdDraw3D.clear();

// Drawing Blue Wireframe Sphere represents the sun //

StdDraw3D.setPenColor(StdDraw.BLUE);

StdDraw3D.wireSphere(0, 0, 0, 120);

x = Math.round(xPos + r \* Math.cos(frame \* angle));

y = Math.round(yPos + r \* Math.sin(frame \* angle));

// Drawing Red Sphere that represents a planet//

StdDraw3D.setPenColor(StdDraw.RED);

StdDraw3D.wireSphere(x, y, 1, 15);

frame++;

StdDraw3D.show(20);

// clear the background

StdDraw3D.clear();

}

}

}

## References

Robert Sedgewick and Kevin Wayne, “Standard Libraries” (June 2020) Available at: <https://introcs.cs.princeton.edu/java/stdlib/> (Accessed 17/02/2021)

Javin Paul, “How to declare an Array List with Values” (2015) Available at: <https://www.java67.com/2015/10/how-to-declare-arraylist-with-values-in-java.html> ( Accessed 01/03/2021)

Dawid09239y2, “How to create an array for storing x and y coordinates in python” (2020) Available at: <https://stackoverflow.com/questions/62121071/how-to-create-an-array-for-storing-x-and-y-coordinates-in-python> (Accessed 17/02/2021)

SkyeBoniwell, “Proper way of getting a variable from another class” (2012) Available at: <https://stackoverflow.com/questions/9212861/proper-way-of-getting-variable-from-another-class> (Accessed 25/02/2021)

Bernhard Barker, “Non-Static variable cannot be referenced from a static context” (2011) Available at: <https://stackoverflow.com/questions/2559527/non-static-variable-cannot-be-referenced-from-a-static-context> (Accessed 26/02/2021)

W3Schools, “Java Inheritance” (2021) Available at: <https://www.w3schools.com/java/java_inheritance.asp> (Accessed 02/02/2021)

W3Schools, “Java Contructors” (2021) Available at: <https://www.w3schools.com/java/java_constructors.asp> (Accessed 08/02/2021)

Phil Brown, “Is there a way to automatically generate getters and setters in Eclipse?” (2012) Available at: <https://stackoverflow.com/questions/7221691/is-there-a-way-to-automatically-generate-getters-and-setters-in-eclipse> (Accessed 12/02/2021)

TheOddy Linux, “How does Math.Random exactly work in java?” (2018) Available at: <https://stackoverflow.com/questions/47833269/how-does-math-random-exactly-work-in-java> (Accessed 27/02/2021)

JavaTPoint, “Java Math.Random() method” (2020) Available at: <https://www.javatpoint.com/java-math-random-method> (Acessed 04/03/2021)