Section 1 – Approach

James M. Mutry

UMGC

**Section 1 – Approach**

My approach to this project wasn’t over thought in any kind of way. This wasn’t my first time creating a GUI. I initially planned how I wanted it to look by looking at some examples online. Next, was figuring out how I would create shapes and expand their size withing the GUI. I looked at some reddit post and overdrive and found some ideas. The ideas were mainly around the use of Java FX’s Canvas. After that these two, I got to coding.

# Section 2 – Assumptions

My assumptions about this project were not stressful at all. I felt that the instructions were very simple and direct. I assumed this project would be very simple. My biggest trouble would be implementing shapes with canvas which was relatively hard when it came to drawing 3-D shapes. I had basically implement my own mathematical formulas to combine 2-D shapes into 3-D shapes.

Section 3 – Not Implemented

The only thing not implemented was using images of 3-D shapes. I was able to create 3-D shapes using canvas besides a torus and a cone. I could have implemented the images to do these, but I got caught trying to create them with canvas and ran out of time. The figures ended up not looking great, but I was still able to do what the project instructed.

**Section 4 – User Guide**

1. Download the program
2. Unzip the project and place into a folder.
3. Open a new project in preferred java software.
4. Select to start project with the downloaded project that was placed into the folder.

**Section 5 – Lessons Learned**

The lessons I learned during this project would probably be to just listen to the instructions of the professor. While creating the other shapes with Canvas in Java FX, I spent about to 2-3 days trying to implement a 3-D torus and cone. I failed and the instructor had already said to use something that would render images for the 3-D shapes. I was just very determined since I had already gotten the other shapes to load.

**Section 6 – Possible Improvements**

The only improvement I would make would be to implement the image renderer for the torus and cone. The shapes I drew up with the Canvas for these two are not adequate. I would just simply add a image render and only apply it for those two shapes.

**Section 7 – UML**

A diagram of a computer program

Description automatically generated with medium confidence

**Section 8 – Source Code**

package com.example.cmsc335p2;  
import javafx.application.Application;  
import javafx.geometry.Insets;  
import javafx.scene.Scene;  
import javafx.scene.canvas.Canvas;  
import javafx.scene.canvas.GraphicsContext;  
import javafx.scene.control.Button;  
import javafx.scene.control.ComboBox;  
import javafx.scene.layout.VBox;  
import javafx.stage.Stage;  
  
public class ShapeDrawer extends Application {  
  
 private Canvas canvas;  
  
 public static void main(String[] args) {  
 *launch*(args);  
 }  
  
 @Override  
 public void start(Stage primaryStage) {  
 primaryStage.setTitle("Shape Drawer");  
  
 // ComboBox for shape selection  
 ComboBox shapeComboBox = new ComboBox<>();  
 shapeComboBox.getItems().addAll("Circle", "Square", "Rectangle", "Triangle", "Cube", "Cylinder", "Sphere", "Torus", "Cone");  
  
 // ComboBox for size selection  
 ComboBox sizeComboBox = new ComboBox<>();  
 sizeComboBox.getItems().addAll(50, 100, 150, 200);  
  
 // Button to draw the shape  
 Button drawButton = new Button("Draw Shape");  
  
 // Canvas to draw the shape  
 canvas = new Canvas(400, 400);  
  
 // Event handler for the draw button  
 drawButton.setOnAction(e -> drawShape((String) shapeComboBox.getValue(), (Integer) sizeComboBox.getValue()));  
  
 // Layout  
 VBox layout = new VBox(10);  
 layout.setPadding(new Insets(20));  
 layout.getChildren().addAll(shapeComboBox, sizeComboBox, drawButton, canvas);  
  
 Scene scene = new Scene(layout, 450, 500);  
 primaryStage.setScene(scene);  
 primaryStage.show();  
 }  
  
 private void drawShape(String shape, Integer size) {  
 if (shape == null || size == null) {  
 return;  
 }  
  
 GraphicsContext gc = canvas.getGraphicsContext2D();  
 gc.clearRect(0, 0, canvas.getWidth(), canvas.getHeight());  
  
 double x = (canvas.getWidth() - size) / 2;  
 double y = (canvas.getHeight() - size) / 2;  
  
 switch (shape) {  
 case "Circle":  
 gc.strokeOval(x, y, size, size);  
 break;  
 case "Square":  
 gc.strokeRect(x, y, size, size);  
 break;  
 case "Rectangle":  
 gc.strokeRect(x, y, size, size / 2.0);  
 break;  
 case "Triangle":  
 double[] xPoints = {x, x + size / 2.0, x + size};  
 double[] yPoints = {y + size, y, y + size};  
 gc.strokePolygon(xPoints, yPoints, 3);  
 break;  
 case "Cube":  
 double offset = size / 4.0;  
 // Front face  
 gc.strokeRect(x, y, size, size);  
 // Top face  
 gc.strokeLine(x, y, x + offset, y - offset);  
 gc.strokeLine(x + size, y, x + size + offset, y - offset);  
 gc.strokeLine(x + offset, y - offset, x + size + offset, y - offset);  
 // Side face  
 gc.strokeLine(x + size, y, x + size + offset, y - offset);  
 gc.strokeLine(x + size, y + size, x + size + offset, y + size - offset);  
 gc.strokeLine(x + size + offset, y - offset, x + size + offset, y + size - offset);  
 break;  
 case "Cylinder":  
 // Draw the top ellipse  
 gc.strokeOval(x, y, size, size / 2.0);  
 // Draw the bottom ellipse  
 gc.strokeOval(x, y + size, size, size / 2.0);  
 // Draw the sides  
 gc.strokeLine(x, y + size / 4.0, x, y + size + size / 4.0);  
 gc.strokeLine(x + size, y + size / 4.0, x + size, y + size + size / 4.0);  
 break;  
 case "Sphere":  
 // Draw the main circle  
 gc.strokeOval(x, y, size, size);  
 // Draw the horizontal ellipse  
 gc.strokeOval(x, y + size / 4.0, size, size / 2.0);  
 // Draw the vertical ellipse  
 gc.strokeOval(x + size / 4.0, y, size / 2.0, size);  
 break;  
 case "Torus":  
 drawTorus(gc, x, y, size);  
 break;  
 case "Cone":  
 drawCone(gc, x, y, size);  
 break;  
 }  
 }  
 private void drawCone(GraphicsContext gc, double x, double y, int size) {  
 double baseWidth = size;  
 double baseHeight = size / 4;  
 double height = size;  
  
 // Draw the base ellipse  
 gc.strokeOval(x, y + height - baseHeight / 2, baseWidth, baseHeight);  
  
 // Draw the sides of the cone  
 gc.strokeLine(x, y + height / 2, x + baseWidth / 2, y);  
 gc.strokeLine(x + baseWidth, y + height / 2, x + baseWidth / 2, y);  
 }  
 private void drawTorus(GraphicsContext gc, double x, double y, int size) {  
 double outerRadius = size / 2;  
 double innerRadius = size / 4;  
  
 // Draw the outer circle  
 gc.strokeOval(x, y, size, size);  
  
 // Draw the inner circle  
 gc.strokeOval(x + outerRadius - innerRadius, y + outerRadius - innerRadius, innerRadius \* 2, innerRadius \* 2);  
  
 // Draw concentric circles to simulate the 3D effect  
 int numCircles = 10;  
 for (int i = 1; i < numCircles; i++) {  
 double ratio = (double) i / numCircles;  
 double currentRadius = outerRadius - (outerRadius - innerRadius) \* ratio;  
 gc.strokeOval(x + outerRadius - currentRadius, y + outerRadius - currentRadius, currentRadius \* 2, currentRadius \* 2);  
 }  
 }  
}

**Section 9 – Test Plans**

|  |  |  |  |
| --- | --- | --- | --- |
| TTest # | Description | Screenshot | Pass/Fail |
| 11 | Circle | A screenshot of a computer  Description automatically generated | Pass |
| 22 | Testing for Rectangle | A screenshot of a computer  Description automatically generated | Pass |
| 33 | Testing for Square | A screenshot of a computer  Description automatically generated | Pass |
| 44 | Testing for Triangle | A screenshot of a computer  Description automatically generated | Pass |
| 55 | Testing for Sphere | A screenshot of a computer  Description automatically generated | Pass |
| 66 | Testing for Cube |  | Pass |
| 77 | Testing for Cone | A screenshot of a computer  Description automatically generated | Pass |
| 88 | Testing for Cylinder | A screenshot of a computer  Description automatically generated | Pass |
| 99 | Testing for Torus | A screenshot of a computer  Description automatically generated | Pass |
| 1 |  |  | Pass |