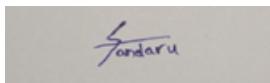


**IN
PARTNERSHIP
WITH
PLYMOUTH
UNIVERSITY**

Name: Sakaladhipathige Fernando

Student Reference Number: 10749110

Module Code: PUSL3122	Module Name: HCI, Computer Graphics, and Visualisation
Coursework Title: Project Report	
Deadline Date: 25 th of May 2023	Member of staff responsible for coursework: Dr. Alaa Alkhafaji
Programme: BSc (Hons) Software Engineering	
Please note that University Academic Regulations are available under Rules and Regulations on the University website www.plymouth.ac.uk/studenthandbook .	
Group work: please list all names of all participants formally associated with this work and state whether the work was undertaken alone or as part of a team. Please note you may be required to identify individual responsibility for component parts.	
Sakaladhipathige Fernando	10749110
Udugama Nuwanthika	10749139
Rajapaksha Rajapaksha	10749121
Merenna Amarasinghe	10749150
Bopage Muthumala	10749145
Randeera Withanage	10749185
<i>We confirm that we have read and understood the Plymouth University regulations relating to Assessment Offences and that we are aware of the possible penalties for any breach of these regulations. We confirm that this is the independent work of the group.</i>	
Signed on behalf of the group:	
Individual assignment: <i>I confirm that I have read and understood the Plymouth University regulations relating to Assessment Offences and that I am aware of the possible penalties for any breach of these regulations. I confirm that this is my own independent work.</i>	
Signed:	
Use of translation software: failure to declare that translation software or a similar writing aid has been used will be treated as an assessment offence. I *have used/not used translation software. If used, please state name of software.....	

Overall mark _____ % Assessors Initials _____ Date _____

SHAPELAND

(SHAPE LEARNING APPLICATION FOR PRIMARY SCHOOL CHILDREN)

PUSL3122 HCI, Computer Graphics,
and Visualisation .

Group No: 44



Name	Ply ID	Email
Udugama Nuwanthika	10749139	10749139@students.plymouth.ac.uk
Rajapaksha Rajapaksha	10749121	10749121@students.plymouth.ac.uk
Merenna Amarasinghe	10749150	10749150@students.plymouth.ac.uk
Sakaladhipathige Fernando	10749110	10749110@students.plymouth.ac.uk
Bopage Muthumala	10749145	10749145@students.plymouth.ac.uk
Randeera Withanage	10749185	10749185@students.plymouth.ac.uk



Table of Contents

Chapter 01 – Introduction	1
Chapter 02 – Background	2
Chapter 03 – Data Gathering	3
Chapter 04 – Design.....	4
4.1 – Story Board	4
4.2 – Low-Fidelity	5
4.3 – High-Fidelity.....	11
4.3.1 – Sign Up	11
4.3.2 – Login.....	12
4.3.3 – Settings Customization	13
4.3.4 – Dashboard	14
4.3.5 – Learn Shapes.....	15
4.3.6 – Match Shapes	17
4.3.7 – Draw with Shapes	19
4.3.8 – Color Shapes	20
4.3.9 – 3D Objects	21
4.3.10 – 3D Space	22
Chapter 05 – Implementation.....	24
5.1 – Login	24
5.2 – Sign Up	26
5.3 – Dashboard	27
5.4 – Learn Shapes	28
5.5 – Match Shapes	32
5.6 – Draw Shapes	35
5.7 – Color Shapes	36
5.8 – 3D Objects	38
5.9 – 3D Space	40
Chapter 06 – Evaluation.....	44
Chapter 07 – Summary	47
References.....	48
Appendices.....	49
Appendix 1 – Responses of Survey	49

Table of Figures

Figure 1: Story Board.....	4
Figure 2: LF-Sign Up.....	5
Figure 3: LF-Login	6
Figure 4: LF-Settings Customization.....	6
Figure 5: LF-Dashboard.....	7
Figure 6: LF-Learn Shapes	7
Figure 7: LF-Match Shapes (Ice Cream)	8
Figure 8: LF-Match Shapes (House).....	8
Figure 9: LF-Match Shapes (Elephant).....	9
Figure 10: LF-Draw with Shapes.....	9
Figure 11: LF-Colour Shapes.....	10
Figure 12: LF-3D Space.....	10
Figure 13: HF-Sign Up	11
Figure 14: HF-Login	12
Figure 15: HF-Settings Customization	13
Figure 16: HF-Dashboard	14
Figure 17: HF-Learn Shapes (Rectangle)	15
Figure 18: HF-Learn Shapes (Square)	15
Figure 19: HF-Learn Shapes (Circle)	16
Figure 20: HF-Learn Shapes (Triangle).....	16
Figure 21: HF-Match Shapes (Ice Cream)	17
Figure 22: HF-Match Shapes (House)	17
Figure 23: HF-Match Shapes (Elephant)	18
Figure 24: HF-Draw with Shapes	19
Figure 25: HF-Colour Shapes (Circle).....	20
Figure 26: HF-Colour Shapes (Rectangle)	20
Figure 27: HF-3D Objects (Sphere).....	21
Figure 28: HF-3D Objects (Cube)	21
Figure 29: HF-3D Space (Galaxy)	22
Figure 30: HF-3D Space (Earth).....	22
Figure 31: HF-3D Space (Mars)	23
Figure 32: HF-3D Space (Moon)	23
Figure 33: Login	24
Figure 34: Source Code of Login.....	25
Figure 35: Sign Up.....	26
Figure 36: Source Code of Sign Up.....	26
Figure 37: Dashboard.....	27
Figure 38: Source Code of Dashboard.....	27
Figure 39: Square	28
Figure 40: Source Code of Square	28
Figure 41: Circle	29
Figure 42: Source Code of Circle	29
Figure 43: Rectangle	30
Figure 44: Source Code of Rectangle	30

Figure 45: Triangle	31
Figure 46: Source Code of Triangle.....	31
Figure 47: Create Ice Cream	32
Figure 48: Source Code of Create Ice Cream	32
Figure 49: Create House	33
Figure 50: Source Code of Create House	33
Figure 51: Create Elephant	34
Figure 52: Source Code of Create Elephant	34
Figure 53: Draw Shapes.....	35
Figure 54: Source Code of Draw Shapes	35
Figure 55: Color Rectangle.....	36
Figure 56: Source Code of Color Rectangle	36
Figure 57: Color Circle	37
Figure 58: Source Code of Color Circle	37
Figure 59: Cube Object.....	38
Figure 60: Source Code of Cube Object.....	38
Figure 61: Sphere Object	39
Figure 62: Source Code of Sphere Object	39
Figure 63: 3D Galaxy.....	40
Figure 64: Source Code of 3D Galaxy.....	40
Figure 65: 3D Earth	41
Figure 66: Source Code of 3D Earth.....	41
Figure 67: 3D Mars	42
Figure 68: Source Code of 3D Mars	42
Figure 69: 3D Moon.....	43
Figure 70: Source Code of 3D Moon.....	43
Figure 71: Workload Matrix	46

Chapter 01 – Introduction

The increasing ubiquity of computing implies that applications are being utilized in diverse contexts by a vast user base. Consequently, the design and development of novel user interfaces must consider complex user profiles, interaction contexts, and multimodal information. The foundation of user interfaces lies in two relatively new fields: Human-Computer Interaction (HCI) and Computer Graphics. However, the integration of HCI and Computer Graphics is crucial in creating effective and visually appealing user interfaces. HCI principles and methods inform the design process by considering user needs, cognitive abilities, and interaction patterns. Computer Graphics techniques contribute to the visual presentation of information, enhancing the user experience and facilitating efficient communication of complex data. The convergence of HCI and Computer Graphics facilitates the development of user interfaces that not only deliver functional capabilities but also engage users through visually captivating and interactive designs. Therefore, integration of HCI and Computer Graphics in education is essential to equip children with the knowledge and skills necessary to design and develop user interfaces that meet the evolving needs of users in various domains. By emphasizing the importance of these fields in academic curricula, educators can prepare future professionals to tackle the challenges of creating user-centric and visually appealing computer applications. (Hascoët, 2015)

Nevertheless, this document presents the development of an educational application designed for primary school children to, with a primary objective on facilitating their understanding of shapes both 2D and 3D. The application aims to provide a comprehensive learning experience, integrating principles from HCI and computer graphics to create an intuitive and engaging user interface. By incorporating HCI principles, the application endeavours to establish an interface that is both user-friendly and accessible to children. The design process emphasizes the importance of user-centered approaches, considering the cognitive and developmental abilities of primary school children. HCI principles such as simplicity, consistency, and feedback mechanisms are employed to ensure an intuitive and seamless user experience throughout the application. Furthermore, by leveraging the power of computer graphics techniques, the application brings shapes to life in both 2D and 3D representations. Accurate rendering of dimensions, colours, and shading provides a realistic visual experience, aiding children in developing a deeper understanding of shape properties and spatial relationships.

Development Source Code - <https://github.com/Plymouth-University/main-coursework-phoenix>

Chapter 02 – Background

The development of ShapeLand application for primary school children to learn about shapes in 2D and 3D formats addresses the need for interactive and engaging tools that enhance the learning experience. This application provides a comprehensive platform for children to explore, visualize, and manipulate different shapes, enabling them to grasp fundamental concepts of geometry and spatial reasoning. The application caters to primary school children, typically aged 4 to 6, who are at a crucial stage of cognitive development and are acquiring foundational knowledge geometry. At this stage, it is important to foster a positive and enjoyable learning environment to promote active participation and retention of information. Traditionally, learning about shapes in classrooms has often relied on static textbooks and limited visual aids. The lack of hands-on experiences and interactive elements can hinder children's understanding and engagement with the subject matter. This ShapeLand application seeks to bridge this gap by providing dynamic and interactive tools that allow children to create, visualize, and manipulate shapes in both 2D and 3D formats.

The application aims to offer an array of features that enable children to not only understand the properties of shapes but also actively participate in their creation and modification. Users can create new shapes by inputting dimensions and witness their visual representation in real-time. The application supports 2D visualization, allowing children to examine shapes on a flat surface, and 3D visualization, offering a more immersive experience. This application incorporates functionalities such as scaling, shading, and colour customization, enabling children to experiment and personalize their shapes. ShapeLand includes a shape matching feature, where users are presented with incomplete pictures and are required to correctly place the corresponding shapes to complete the full picture. Additionally, it includes a 3D space feature where children can observe galaxy, earth, mars, and moon in 3D. Those engaging activities promote visual perception and shape recognition skills among primary school children.

Chapter 03 – Data Gathering

In the process of developing the ShapeLand application, various techniques were utilized to gather data. These techniques were carefully chosen to ensure a comprehensive and reliable collection of relevant information. The main techniques that utilized in this study as follows:

Extensive research was conducted to gather relevant information and insights on shape learning in primary school education. This research involved studying research papers, academic journals and educational resources related to shape geometry and pedagogical approaches. In there, primary goal was to gain a comprehensive understanding of established teaching methodologies, learning objectives, and effective strategies for engaging primary school students.

To gather data for the development of the educational application, a survey was conducted in selected primary schools in Colombo, Gampaha and Kalutara districts. The purpose of the survey was to collect valuable insights and feedback from primary school teachers. It consisted of structured and open-ended questions that focused on topics such as current shape learning practices, challenges faced, preferences for educational technology, and desired features in a shape learning application. The questions were carefully crafted to ensure clarity and encourage meaningful responses. (Appendix 1)

User feedbacks played a crucial role in gathering data during the development of the application. Prototype versions of the application were tested with primary school children in controlled settings. Observations and interviews were conducted to gather feedback on usability, engagement, and the effectiveness of different features. This iterative process allowed for continuous improvement of the application based on user experiences and suggestions.

4.1 – Story Board

The following scenario unfolds within a classroom, featuring a teacher and a group of students engaged in a collaborative discussion. The primary objective of this interaction revolves around the students' collective endeavor to ascertain the shape corresponding to a particular name they have in mind. However, the teacher assumes the role of an instructor, imparting precise and accurate knowledge regarding the designated shape to the students' utilizing activities.

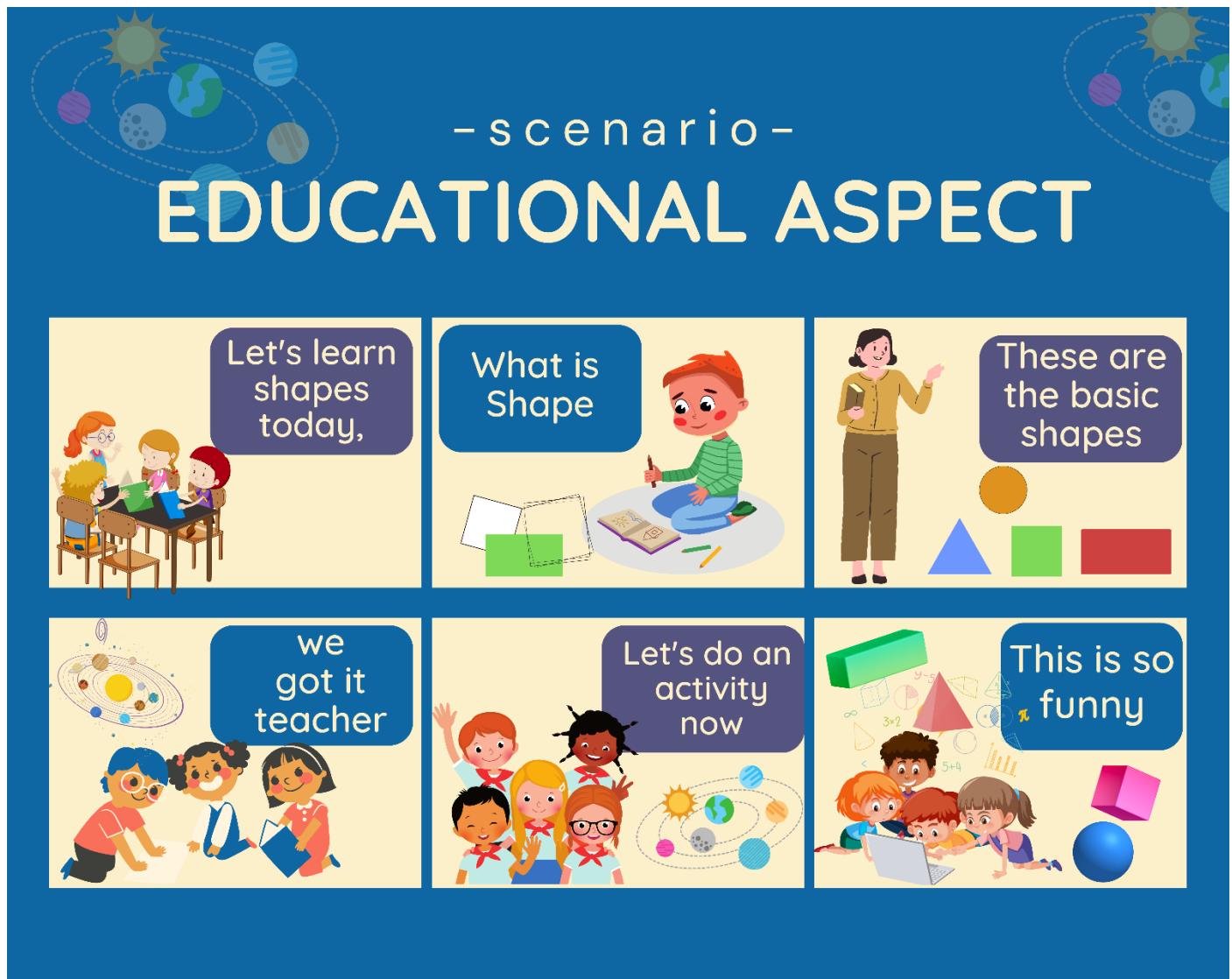


Figure 1: Story Board

4.2 – Low-Fidelity

The initial design iteration of ShapeLand involved creating low-fidelity prototypes using sketches. These rough sketches outlined the basic layout, placement of elements, and overall structure of the application. Since these low-fidelity prototypes provide a quick and cost-effective way to visualize the app's concept it has been utilized.

Sign up

Figure 2: LF-Sign Up

Log in

username

password

log in

Figure 3: LF-Login

Select your age	4	5	6
mode	easy	medium	hard
Language	english	french	spanish

Figure 4: LF-Settings Customization

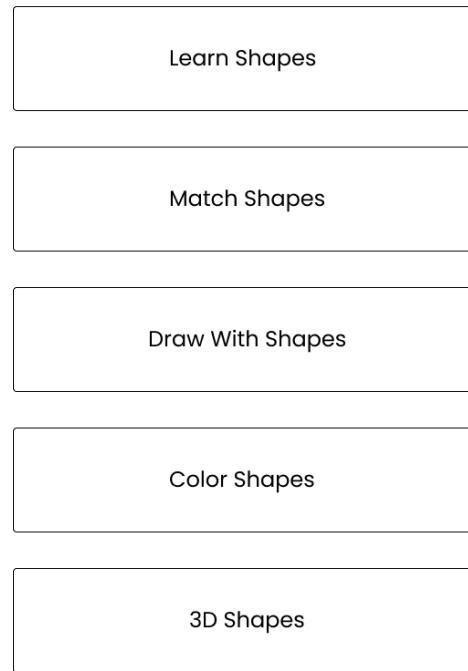
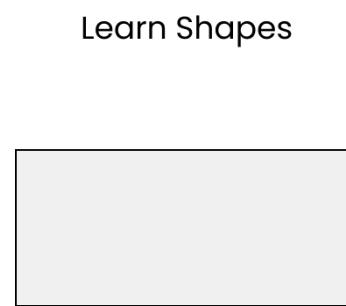


Figure 5: LF-Dashboard



I'm a Rectangle

Figure 6: LF-Learn Shapes

Match shapes

Draw an Ice Cream

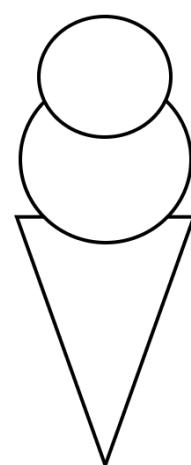


Figure 7: LF-Match Shapes (Ice Cream)

Match shapes

Draw a House

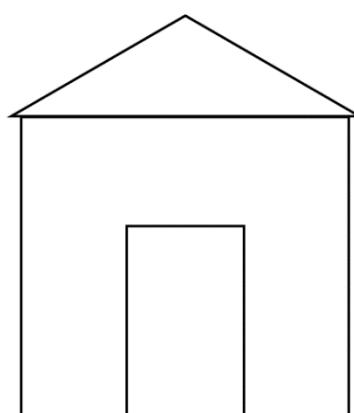


Figure 8: LF-Match Shapes (House)

Match shapes

Draw an Elephant

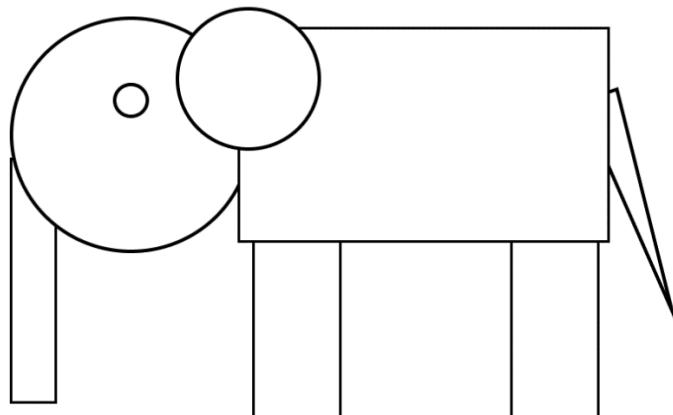


Figure 9: LF-Match Shapes (Elephant)

Draw with Shapes

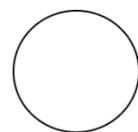


Figure 10: LF-Draw with Shapes

Color Shapes

Color the rectangle in Red

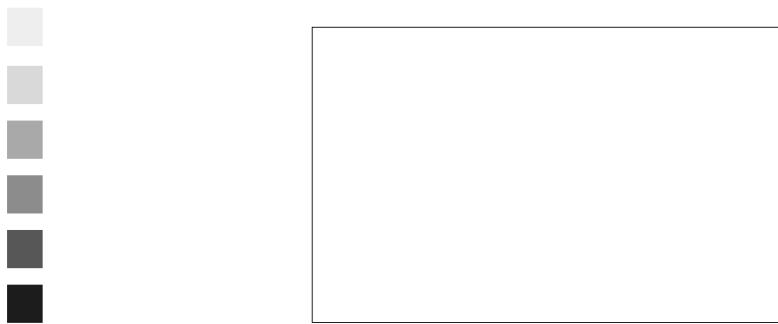


Figure 11: LF-Colour Shapes

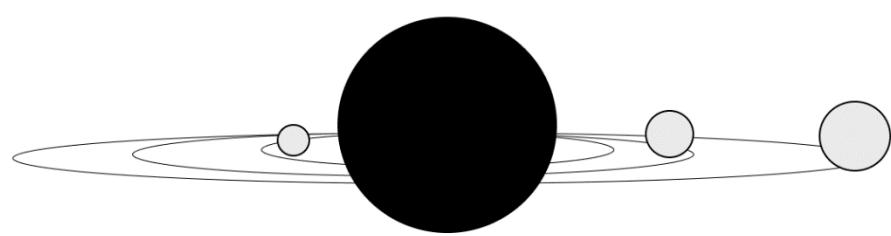
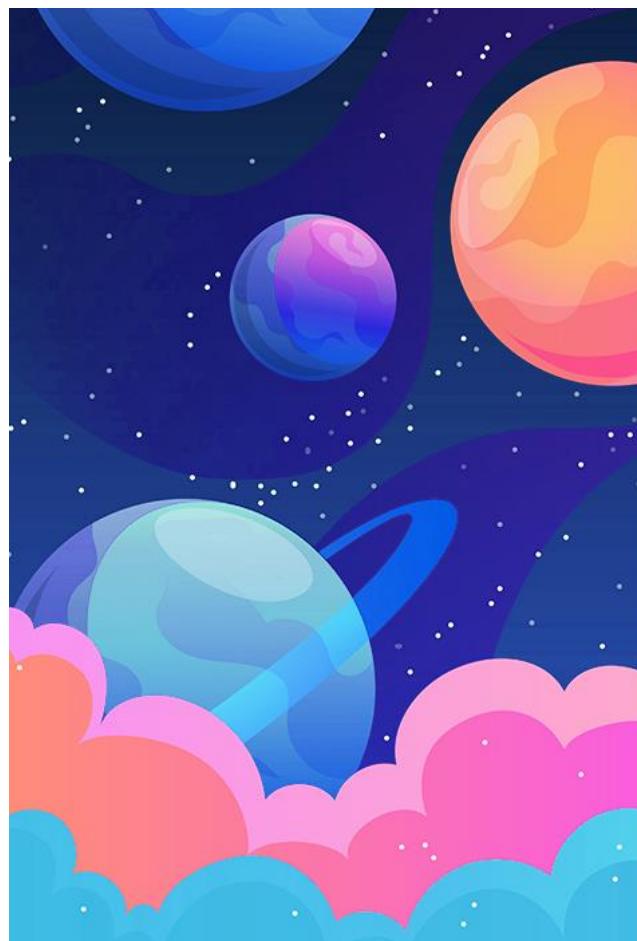


Figure 12: LF-3D Space

4.3 – High-Fidelity

The design process progressed to the creation of a high-fidelity prototype using Figma. The main objective of designing a high-fidelity prototype is to replicate the app's interface and interactions with a higher level of detail and visual fidelity. It incorporated realistic colors, typography, and interactive elements to simulate the user experience more accurately. These prototypes were utilized for comprehensive usability testing evaluations to gather information.

4.3.1 – Sign Up



Let's Start your Journey

Join Us ! Let's Start Learning

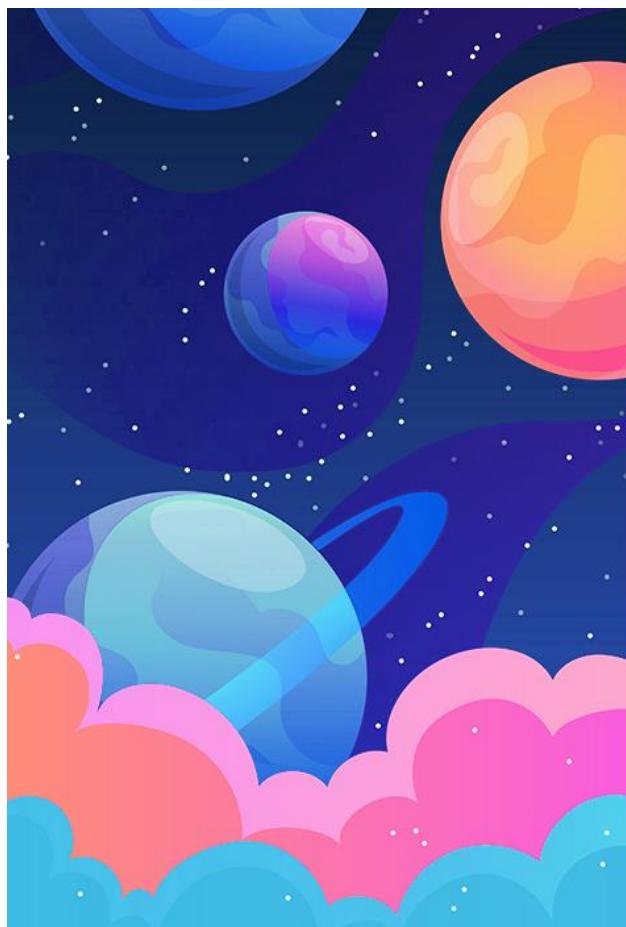
username
email
password

Login

[Log into Existing account](#)

Figure 13: HF-Sign Up

4.3.2 – Login



Hello There

Welcome back! let's Start Learning

Login

[Create New Account](#)

Figure 14: HF-Login

4.3.3 – Settings Customization

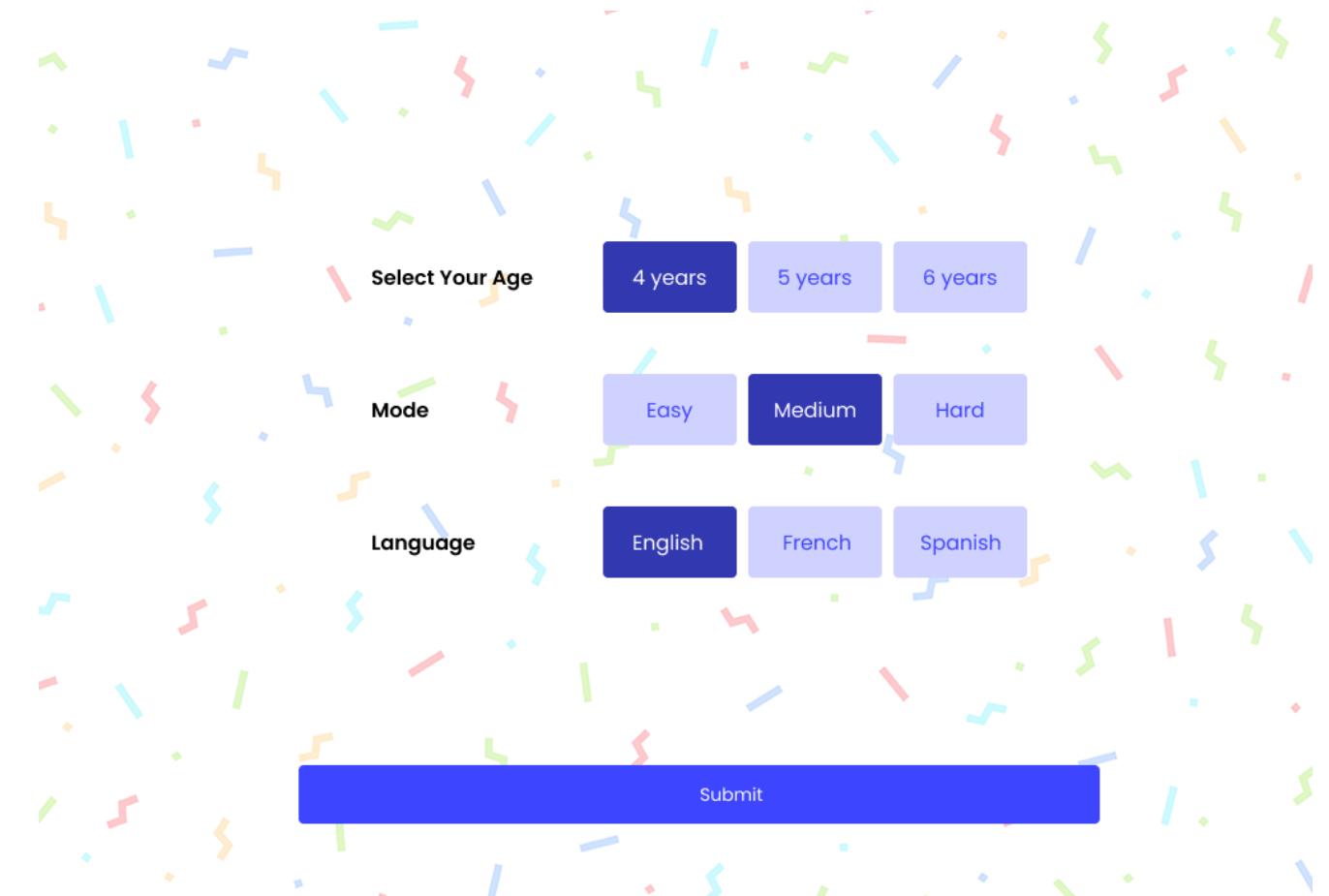


Figure 15: HF-Settings Customization

4.3.4 – Dashboard

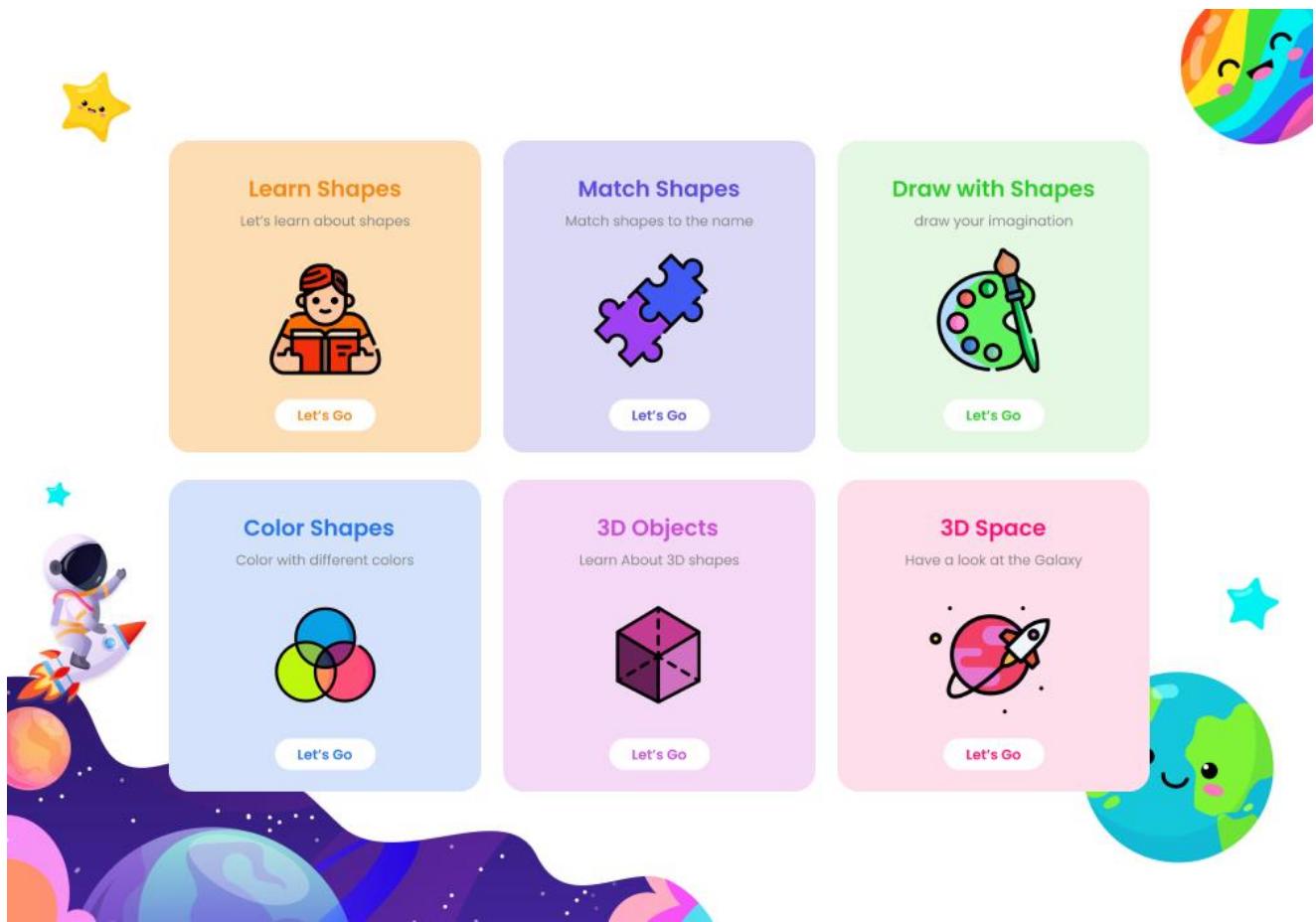


Figure 16: HF-Dashboard

4.3.5 – Learn Shapes

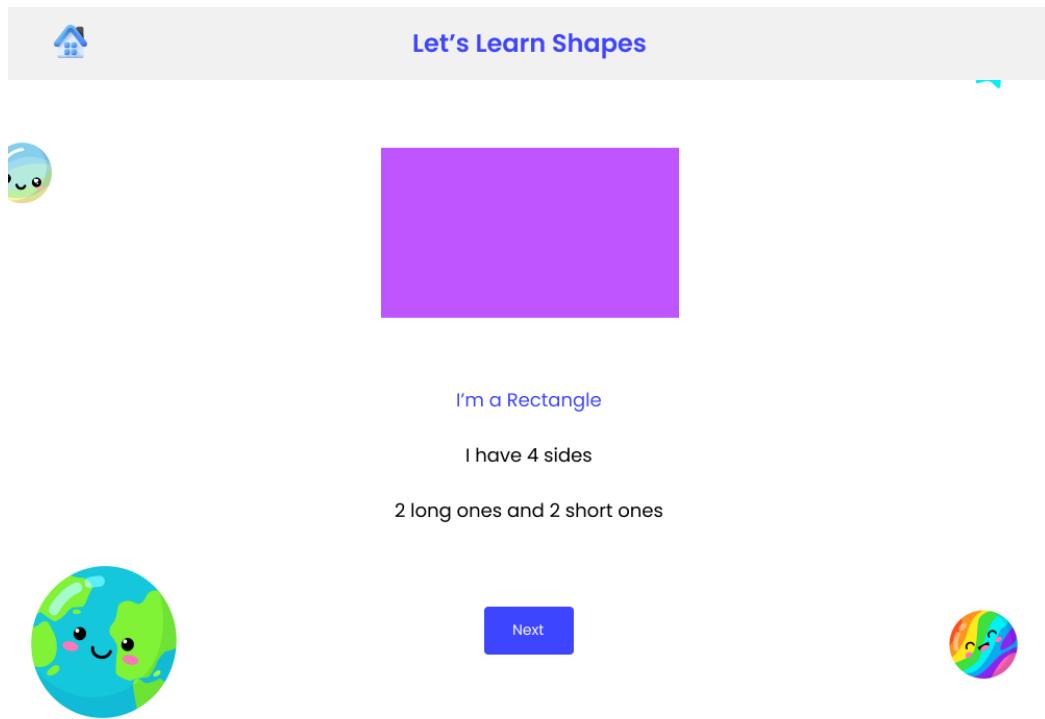


Figure 17: HF-Learn Shapes (Rectangle)

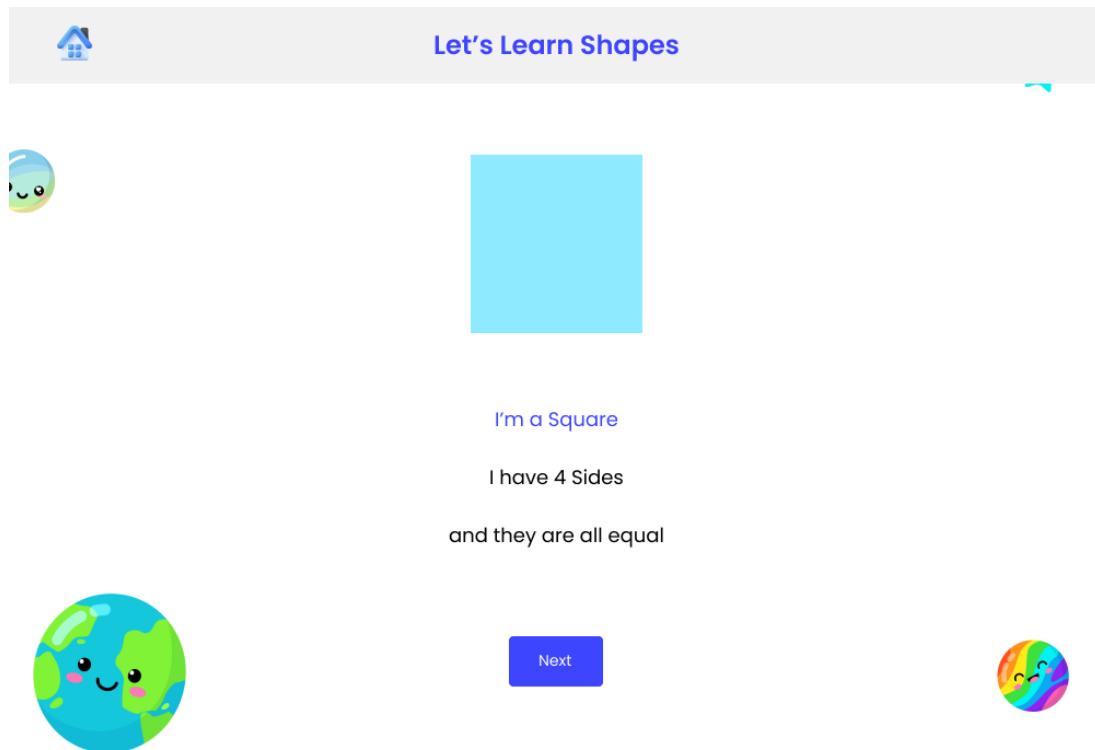


Figure 18: HF-Learn Shapes (Square)

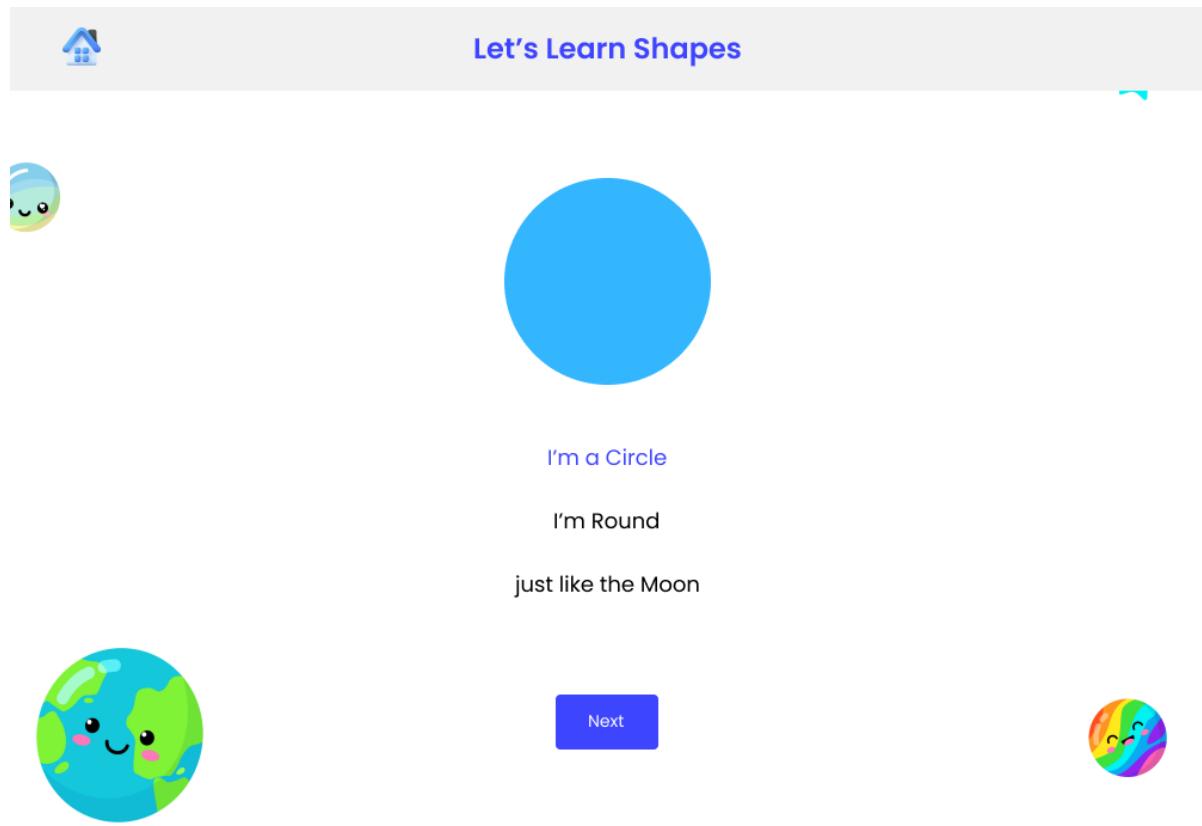


Figure 19: HF-Learn Shapes (Circle)

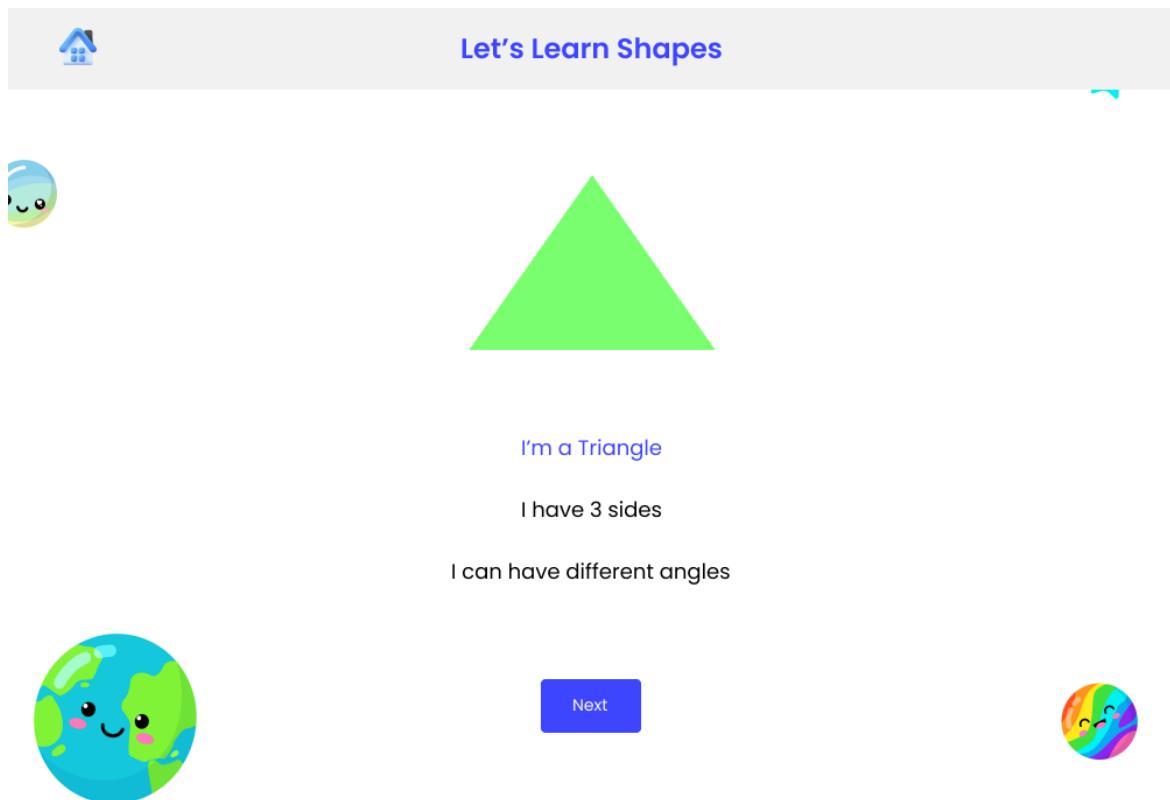


Figure 20: HF-Learn Shapes (Triangle)

4.3.6 – Match Shapes

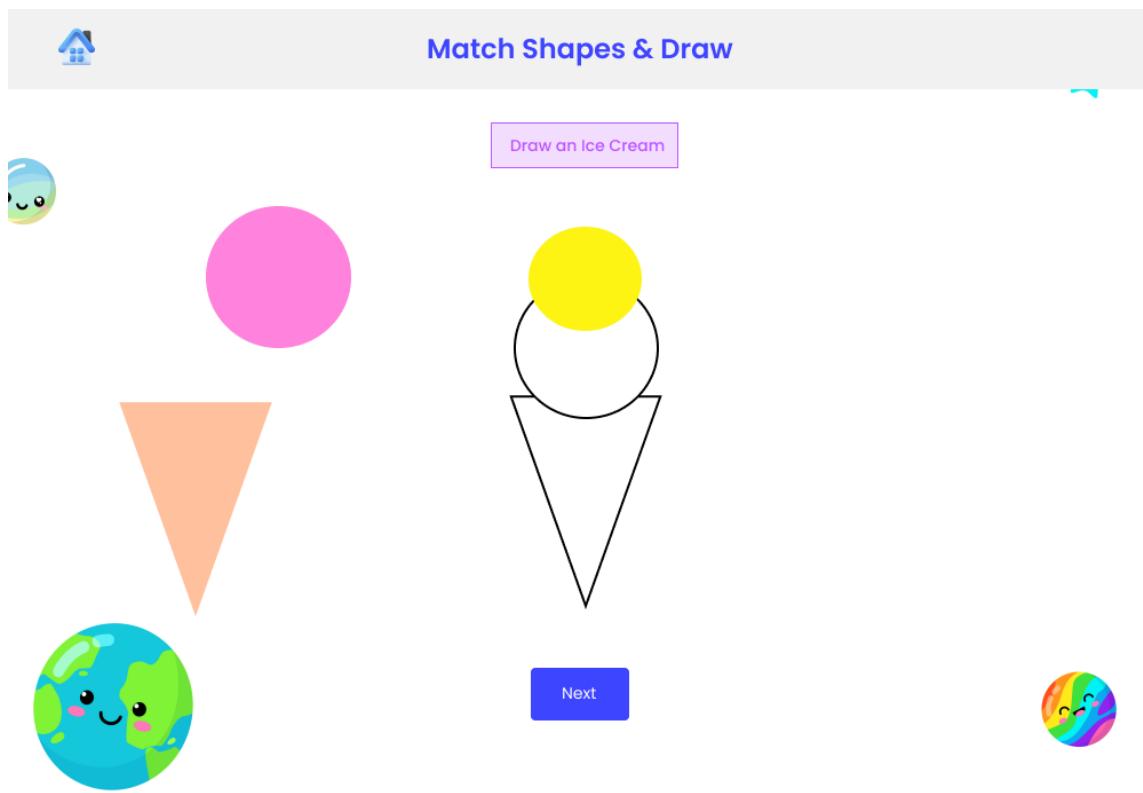


Figure 21: HF-Match Shapes (Ice Cream)

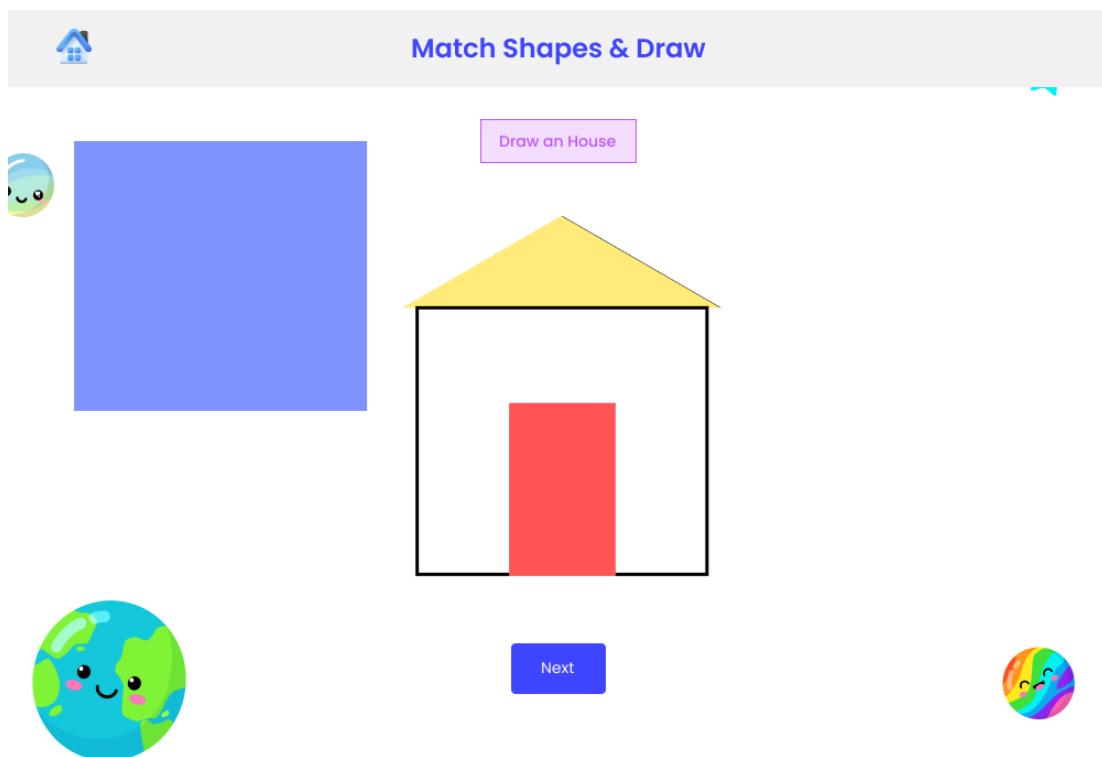


Figure 22: HF-Match Shapes (House)

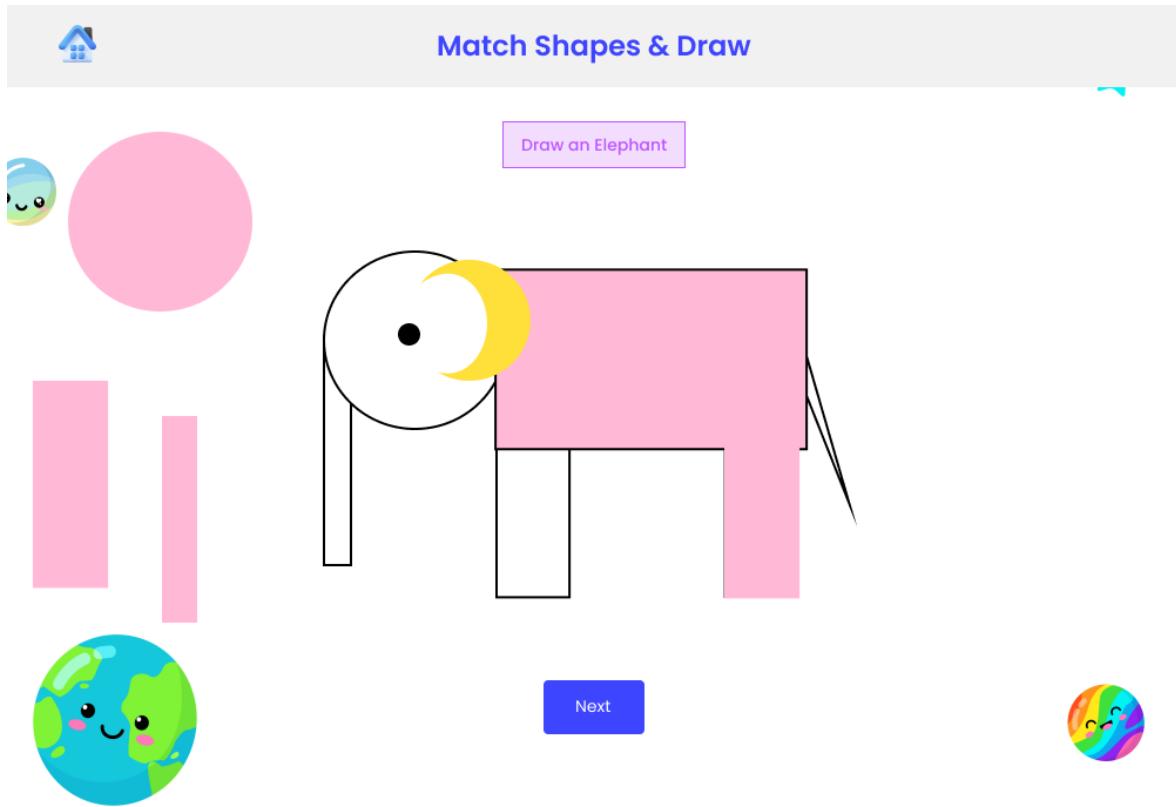


Figure 23: HF-Match Shapes (Elephant)

4.3.7 – Draw with Shapes

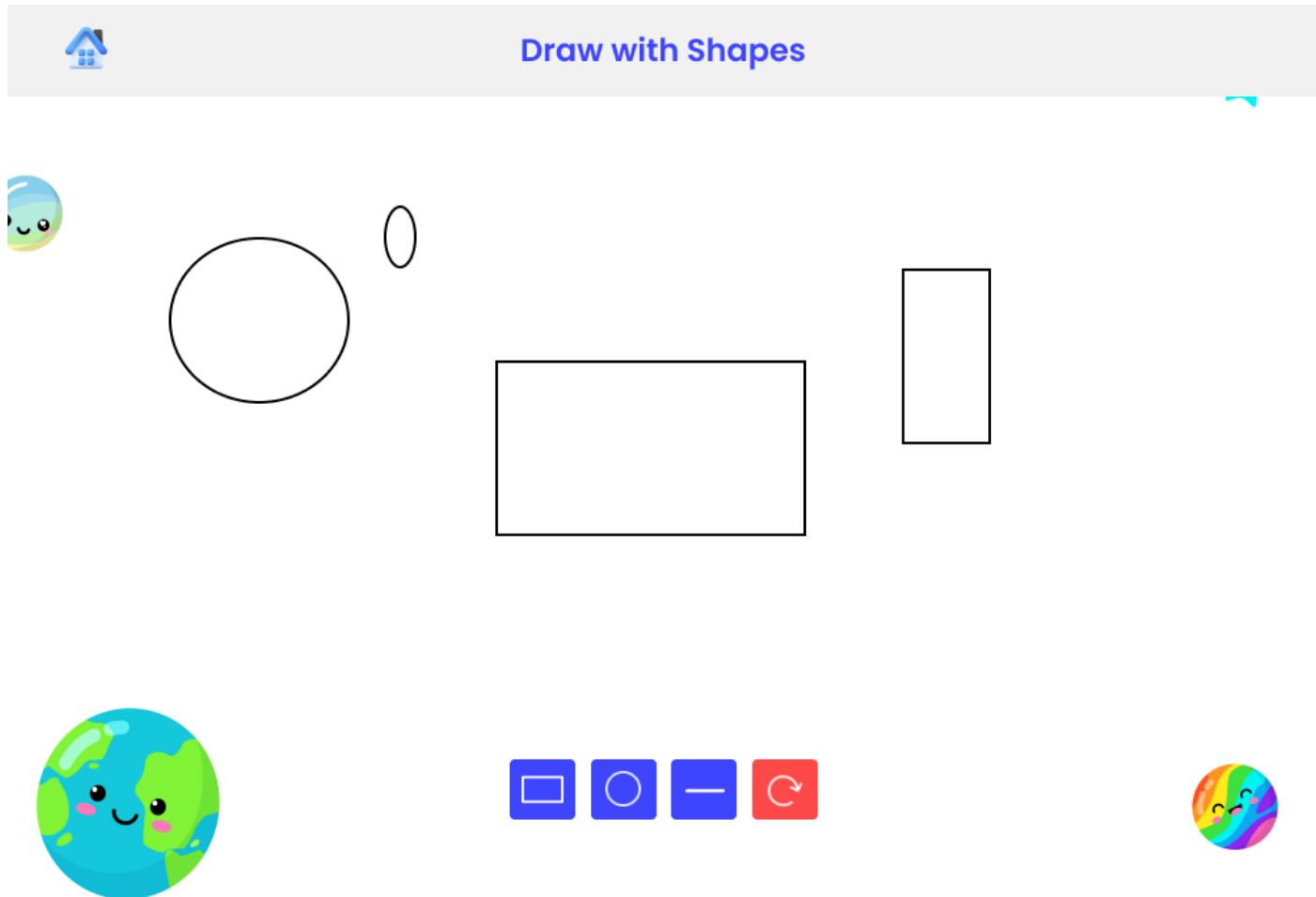


Figure 24: HF-Draw with Shapes

4.3.8 – Color Shapes

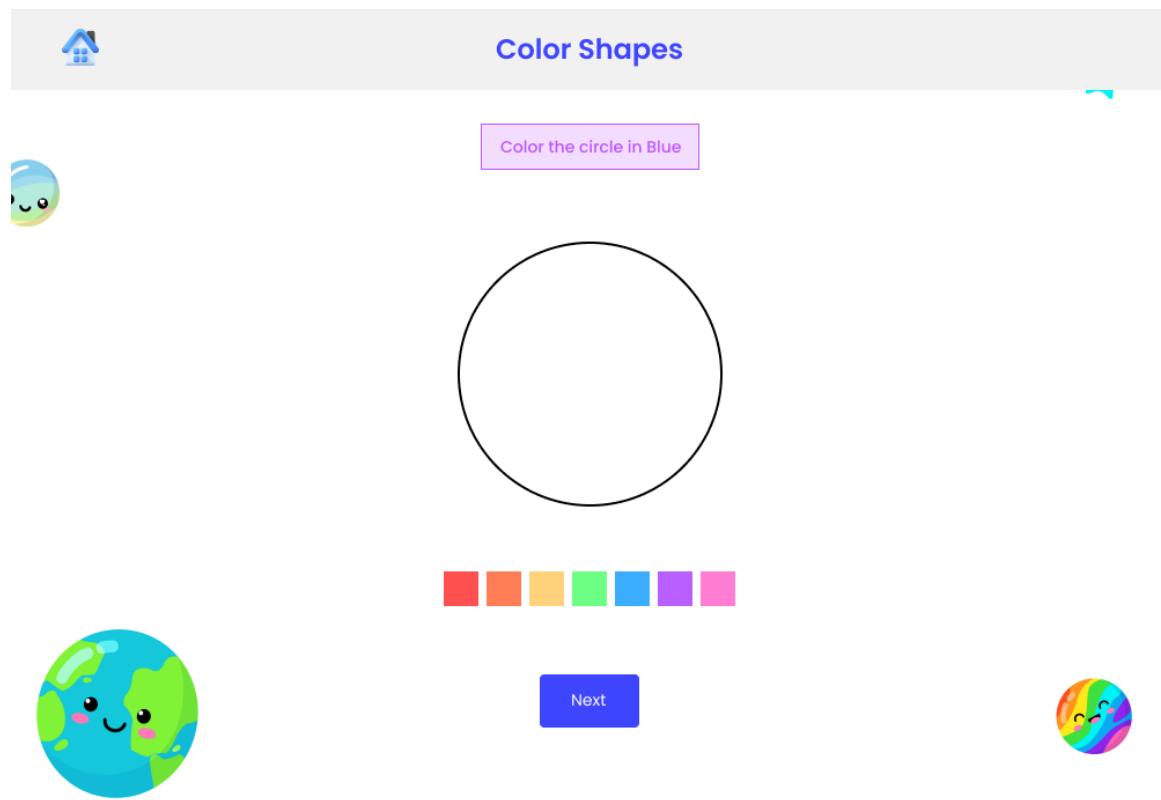


Figure 25: HF-Colour Shapes (Circle)

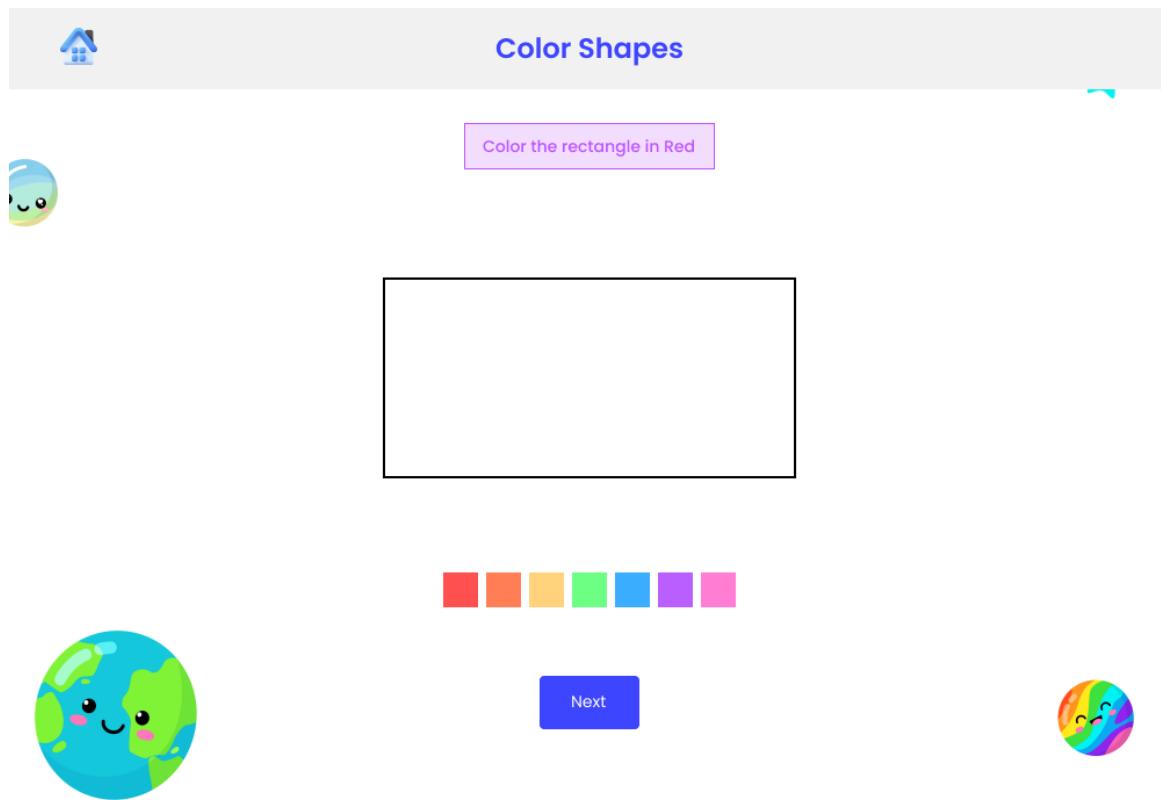


Figure 26: HF-Colour Shapes (Rectangle)

4.3.9 – 3D Objects

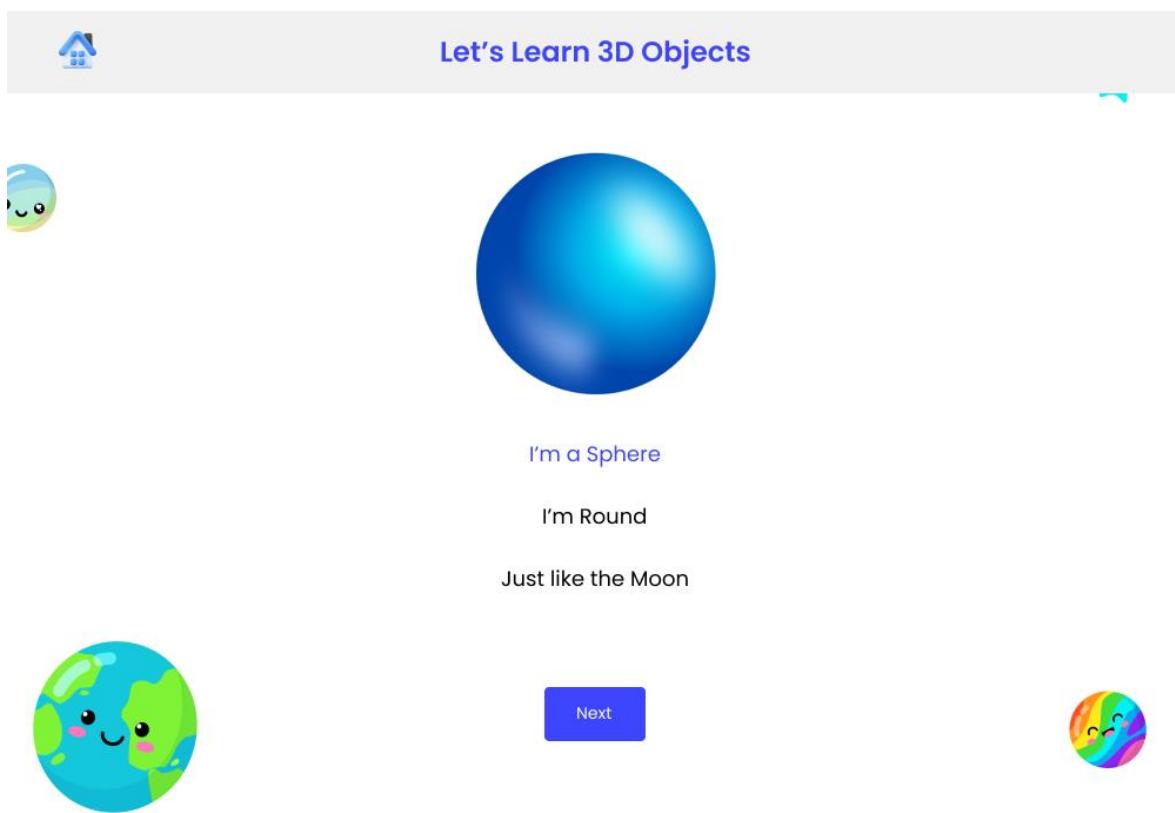


Figure 27: HF-3D Objects (Sphere)

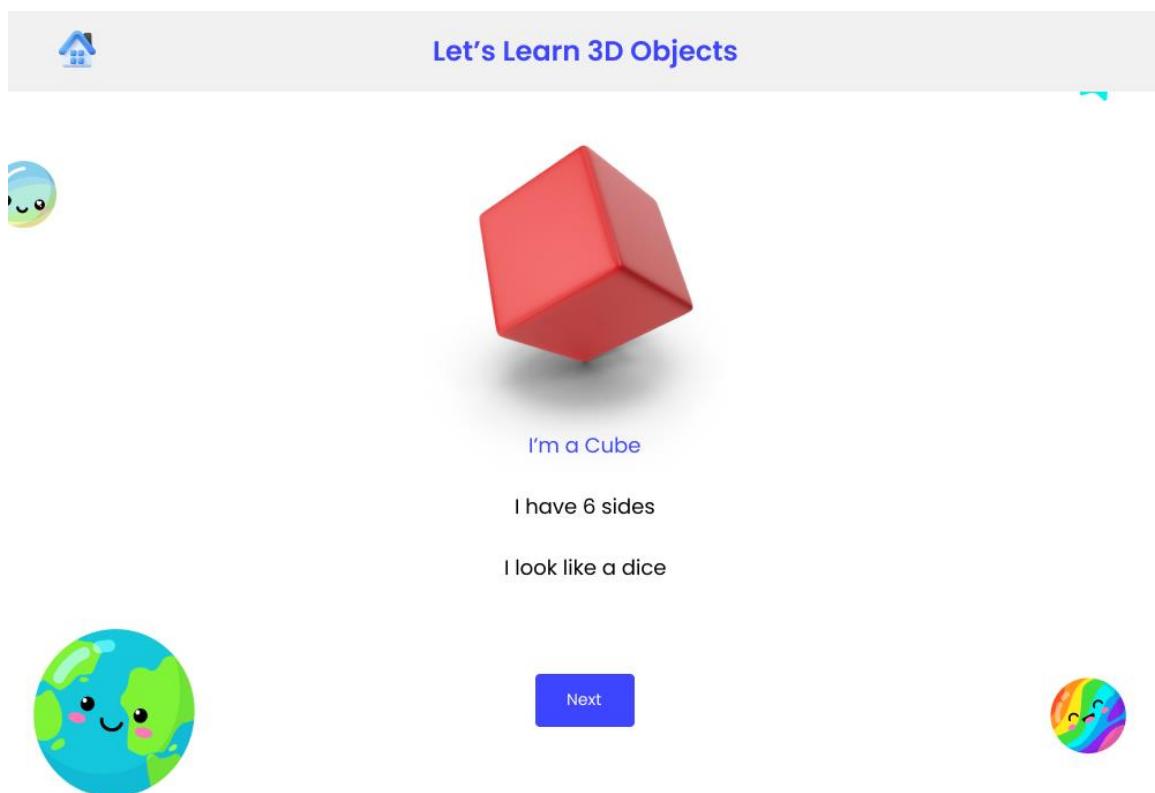


Figure 28: HF-3D Objects (Cube)

4.3.10 – 3D Space

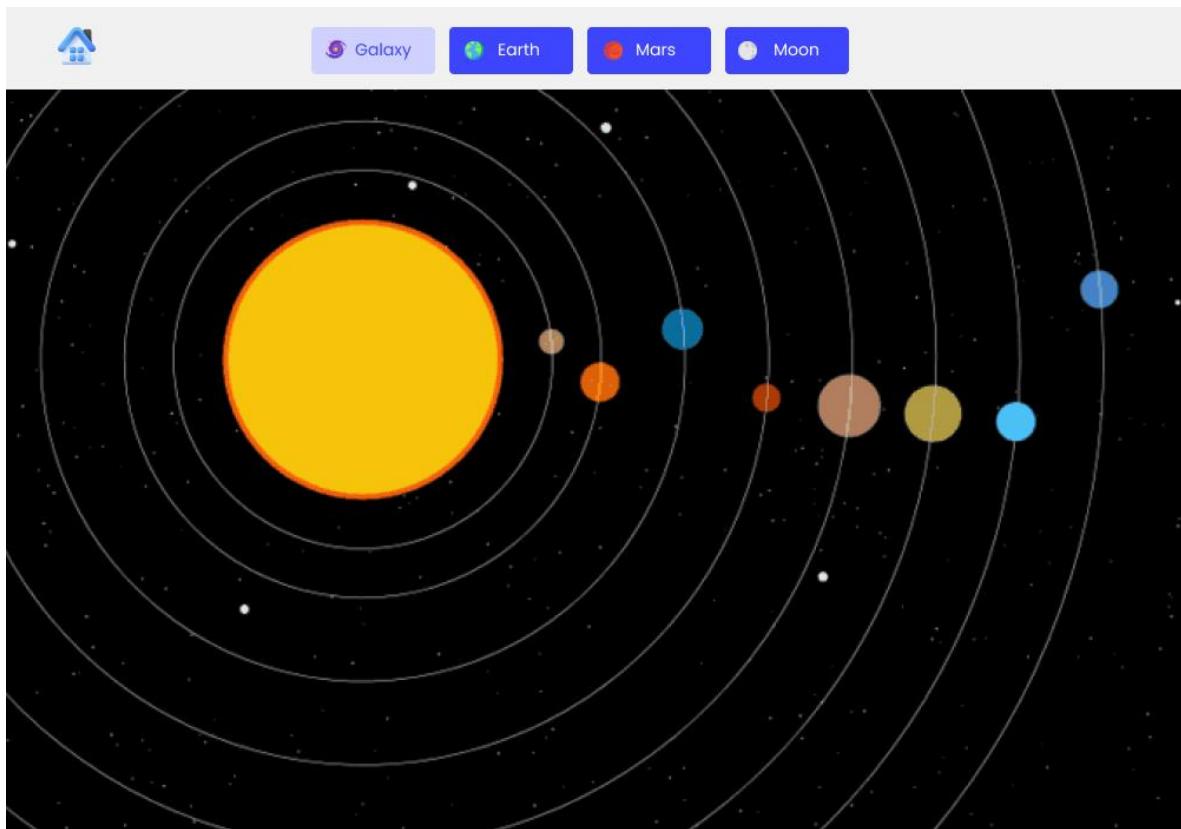


Figure 29: HF-3D Space (Galaxy)

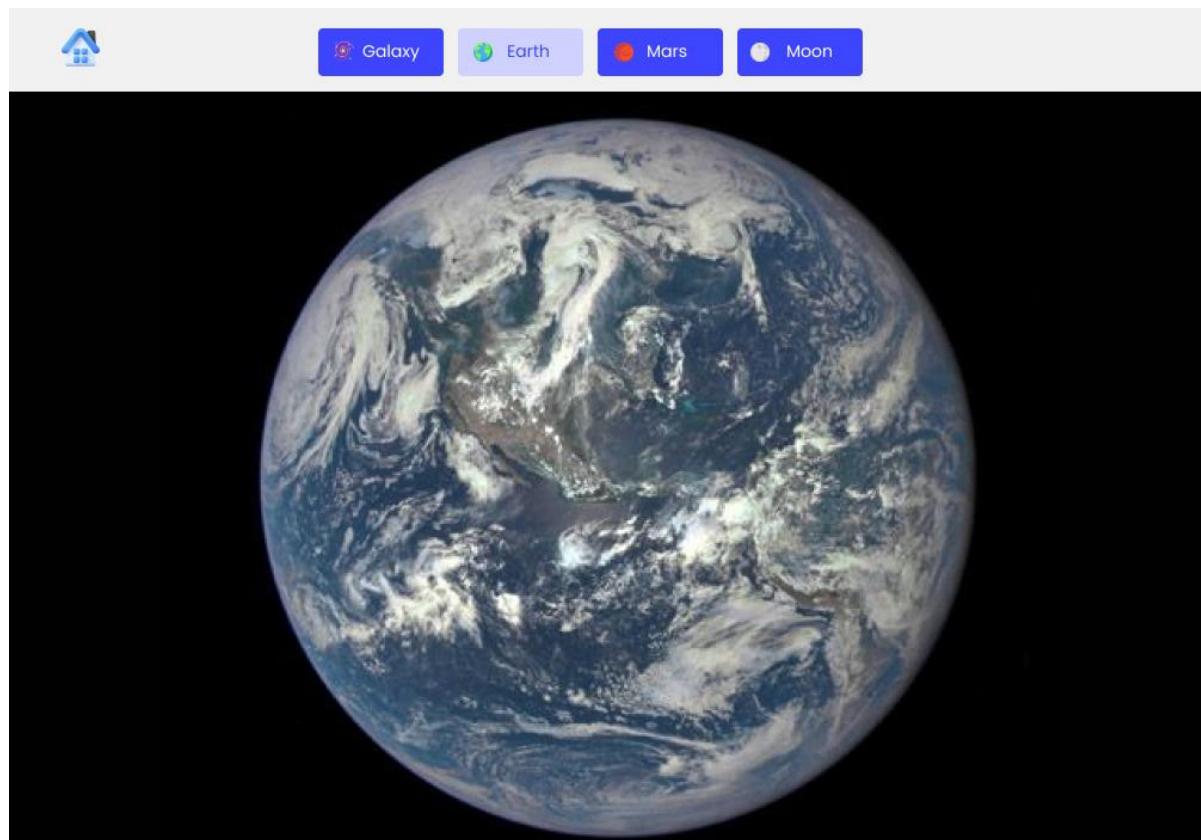


Figure 30: HF-3D Space (Earth)

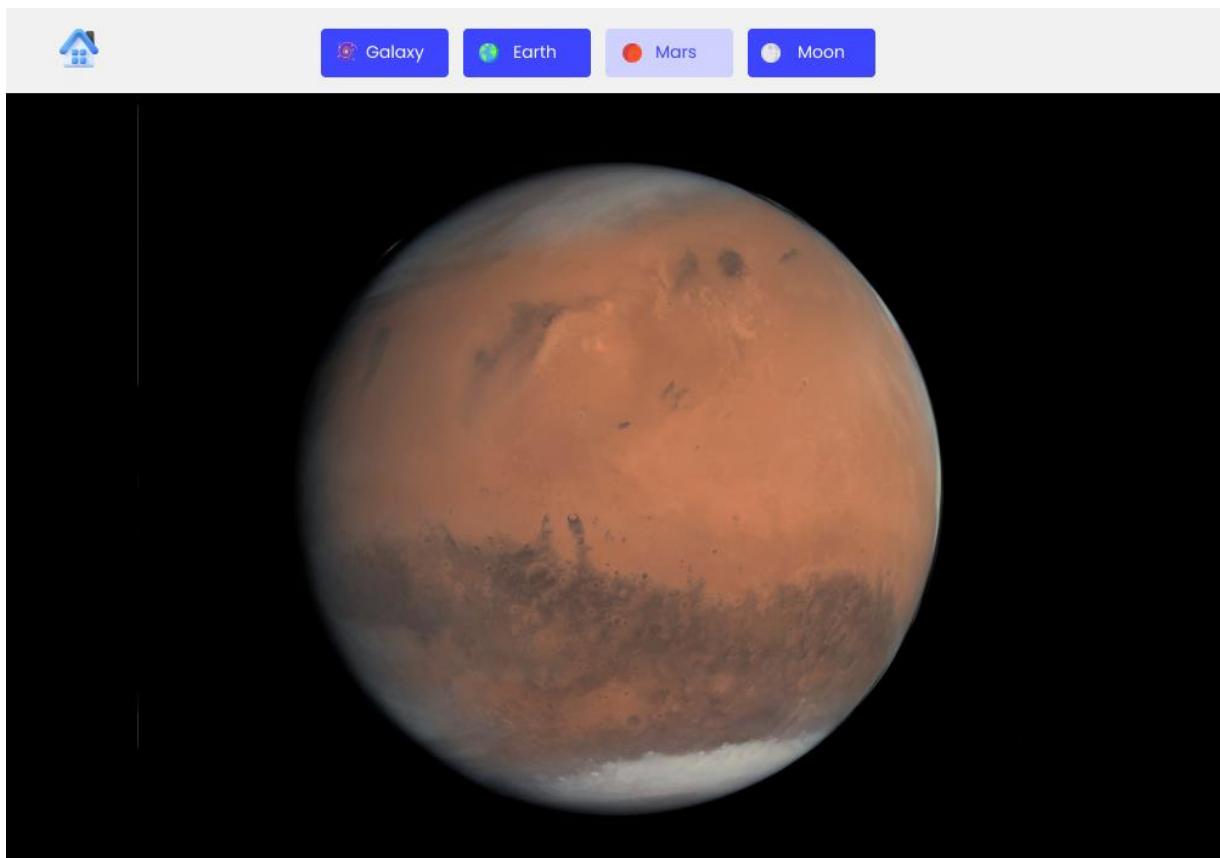


Figure 31: HF-3D Space (Mars)

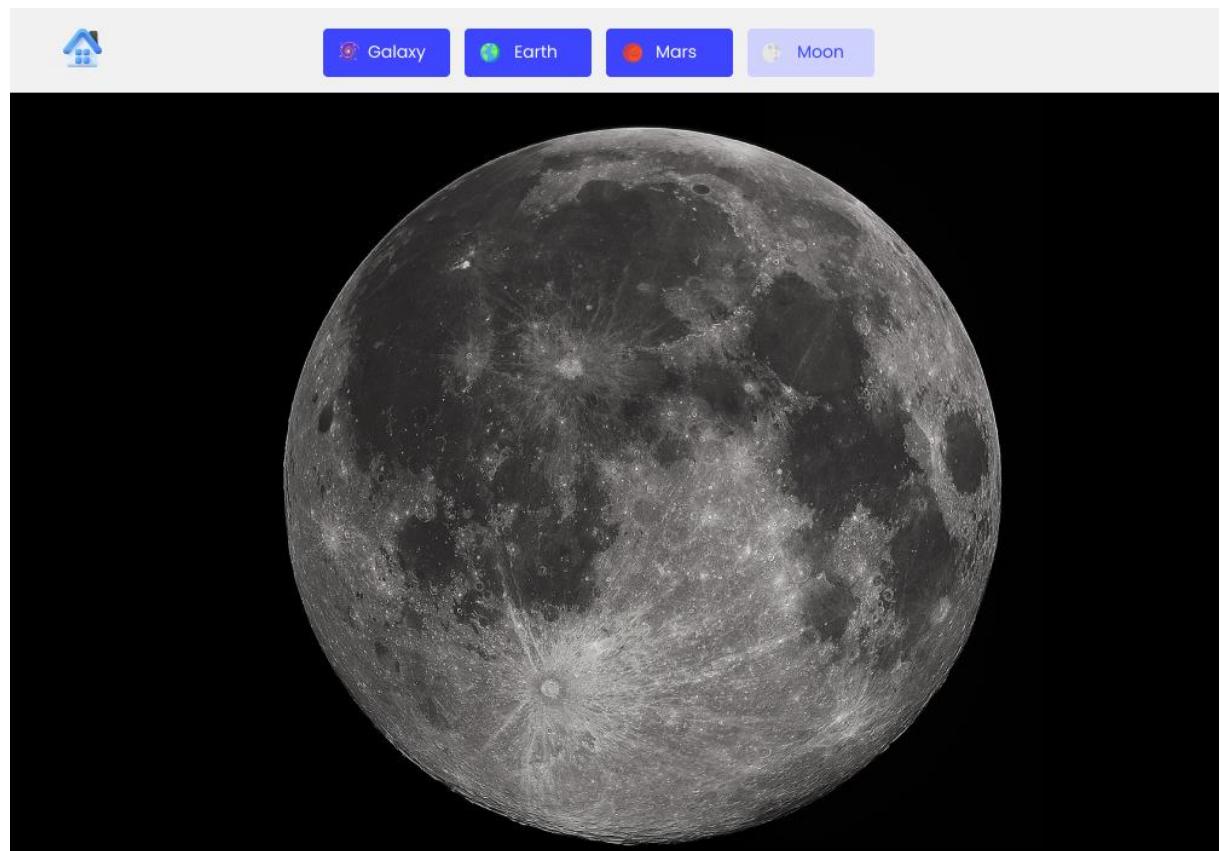


Figure 32: HF-3D Space (Moon)

Chapter 05 – Implementation

The implementation of the ShapeLand application was executed utilizing the Java Swing framework, which is a robust and widely adopted technology, as the development platform for this application. By leveraging the capabilities of Java Swing, the application's user interface (UI) was meticulously designed to provide an intuitive and engaging learning experience for primary school children. The UI elements, including buttons, menus, dialog boxes, and graphical representations of shapes, were created using Java Swing's extensive library of components.

GitHub Repository Link - <https://github.com/Plymouth-University/main-coursework-phoenix>

5.1 – Login

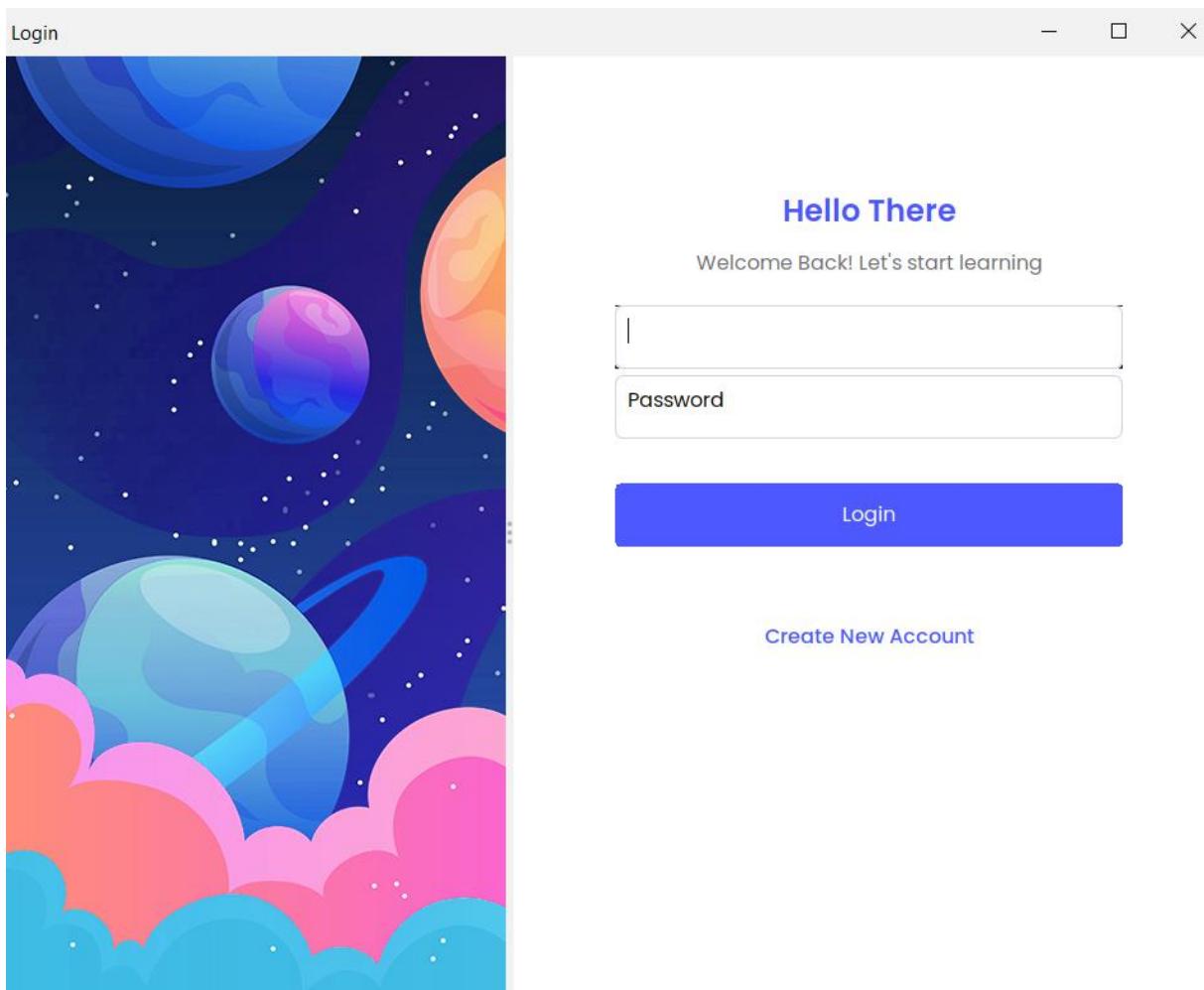


Figure 33: Login

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Tree:** On the left, the project structure is shown under the "Project" tab. It includes packages like "Clicked", "Draw", "Gradient", "Home", "Mars", "Moon", "MyRender", "NameLabel", "RoundedTextfield", and "Styled". Under the "Login" package, there are files: "Login", "Login.java", "Login.form", and "Mars".
- Code Editor:** The main window displays the content of the "Login.java" file. The code is as follows:

```
import ...
public class Login {
    private JPanel jPanel;
    private BufferedImage backgroundImage;
    public static void main(String[] args) {
        FlatLightLaf.setup();
        SwingUtilities.invokeLater(new Runnable() {
            public void run() { createAndShowGUI(); }
        });
    }
    public static void createAndShowGUI() {
        JFrame frame = new JFrame("Login");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Login login = new Login();
        login.loadBackgroundImage("Images/b1.png");
        JSplitPane splitPane = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT);
        splitPane.setResizeWeight(0.4);
    }
}
```

- Toolbars and Status Bar:** The top bar shows standard menu items (File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, Git, Window, Help) and the title "main-coursework-phoenix - LoginJava". The bottom status bar shows the time (21:14), file format (CRLF), encoding (UTF-8), spaces (4 spaces), and the user's name (Sandaru).

Figure 34: Source Code of Login

5.2 – Sign Up

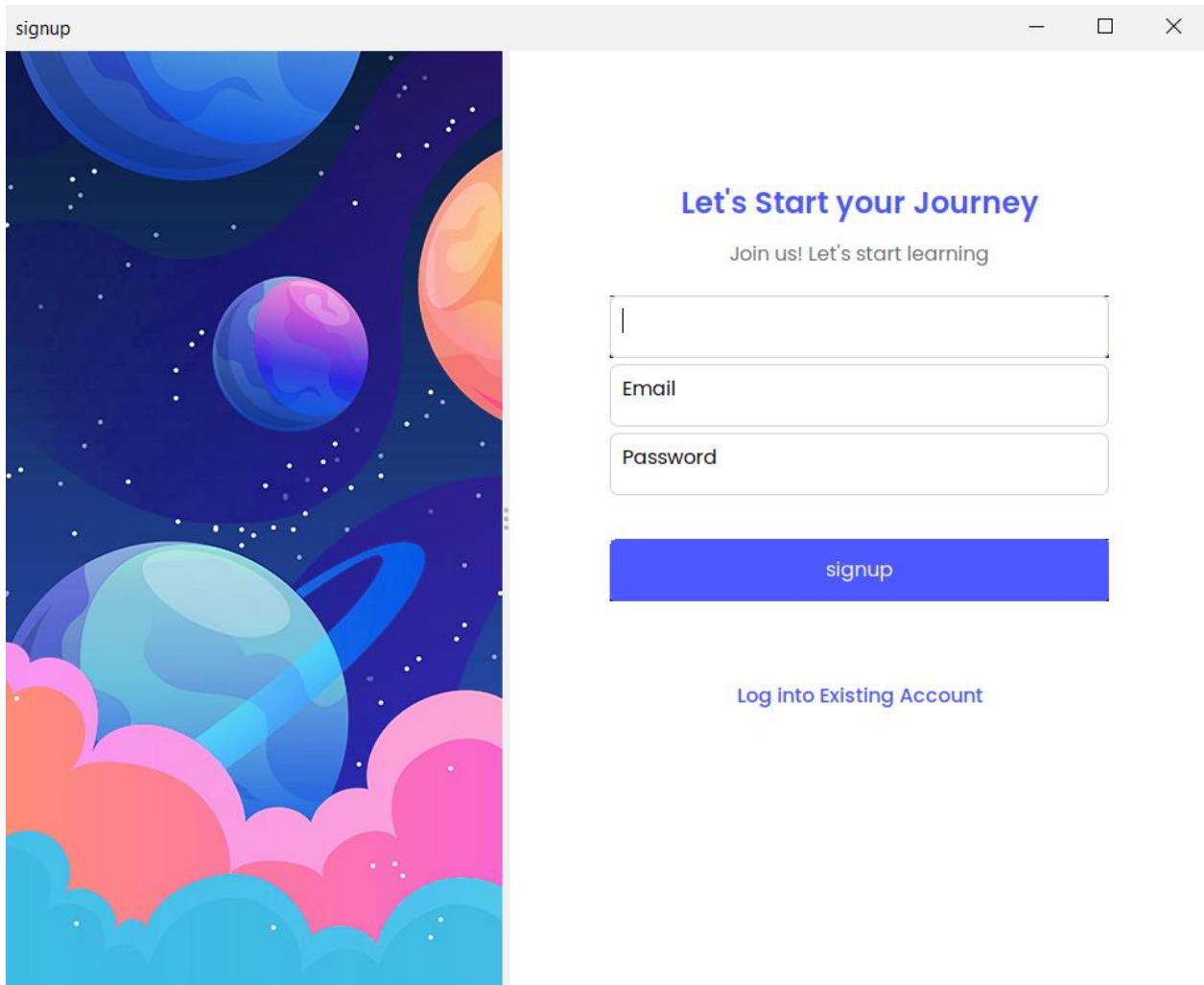


Figure 35: Sign Up

A screenshot of an IDE (IntelliJ IDEA) showing the source code for the "SignUp" class. The code is written in Java and defines a window titled "Sign Up" with a width of 1000 and height of 700 pixels. It includes a main method that sets up the window and calls a static method "createAndShowGUI" to display it. The code is annotated with comments and imports. The IDE interface shows other files like "Earth.java", "Login.java", and "MyRender2.java" in the background.

Figure 36: Source Code of Sign Up

5.3 – Dashboard

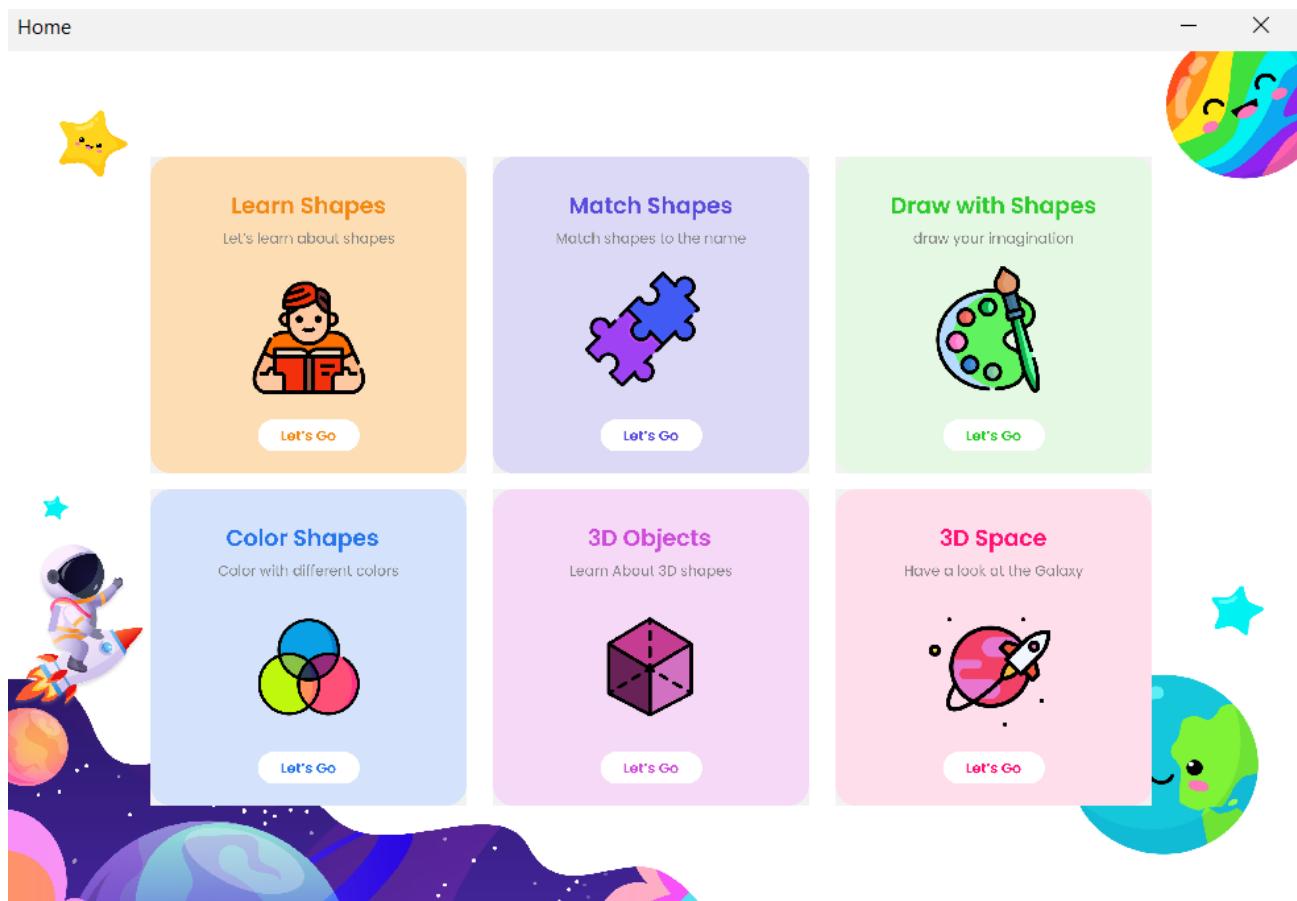


Figure 37: Dashboard

A screenshot of the IntelliJ IDEA IDE showing the Java source code for the "Home.java" file. The code defines a class "Home" with various private fields for JPanel components and a constructor that initializes a JFrame. The code uses annotations like `import ...`, `private JFrame frame;`, `private JPanel panel;`, `private JPanel topPanel;`, `private JPanel bottomPanel;`, `private JPanel topSubPanel1;`, `private JPanel topSubPanel2;`, `private JPanel topSubPanel3;`, `private JPanel bottomSubPanel1;`, `private JPanel bottomSubPanel2;`, `private JPanel bottomSubPanel3;`, and `private BufferedImage backgroundImage;`. The code is annotated with `Bhagya1024`.

Figure 38: Source Code of Dashboard

5.4 – Learn Shapes

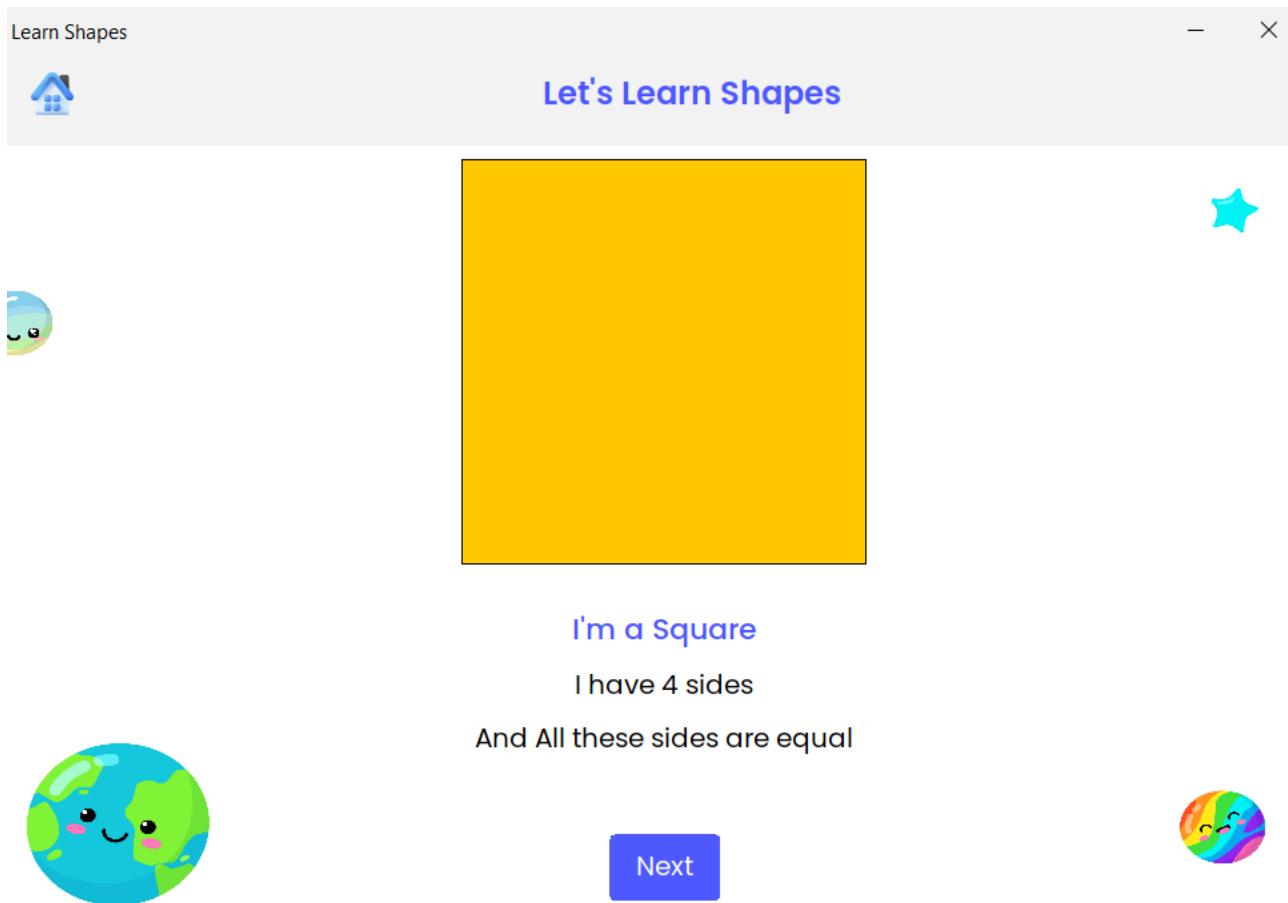


Figure 39: Square

A screenshot of an IDE (IntelliJ IDEA) showing the source code for `SquarePanel.java`. The code is part of a package named `Shapes` under the `Renderables` package. The code defines a class `SquarePanel` that extends `JPanel`. It includes methods for painting the panel, setting the color to orange, and drawing a square outline. The code is annotated with comments explaining the logic.

```
package Renderables.Shapes;
import ...;

public class SquarePanel extends JPanel {
    int size = Math.min(getWidth() - 20, getHeight() - 20);
    int x = (getWidth() - size) / 2;
    int y = (getHeight() - size) / 2;
    private Color selectedColor=Color.ORANGE;
    Graphics g;
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.clearRect(0, 0, getWidth(), getHeight());
        g.setColor(selectedColor);
        g.fillRect(x, y, size, size);
        g.setColor(Color.BLACK);
        g.drawRect(x, y, size, size);
    }
}
```

Figure 40: Source Code of Square

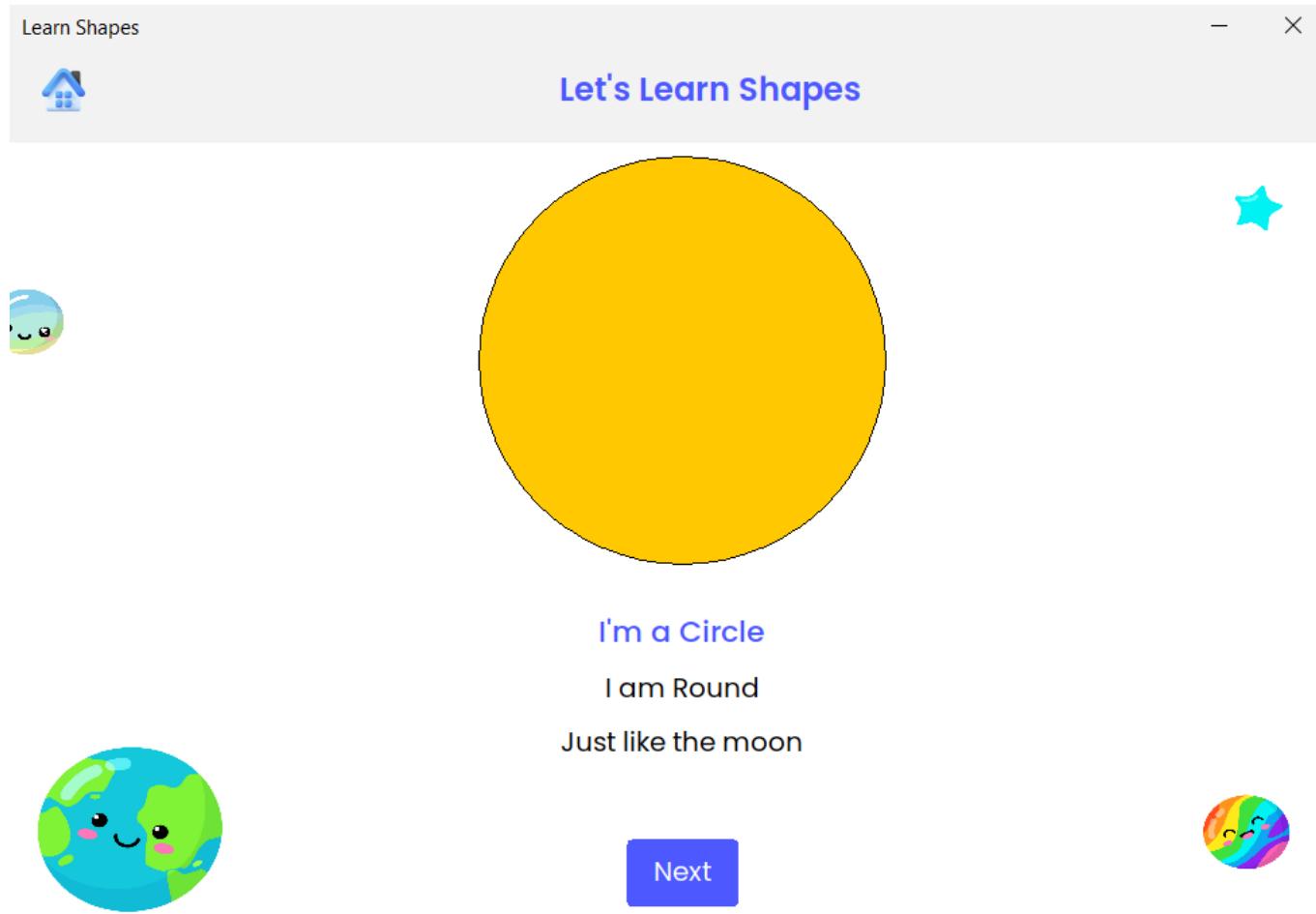
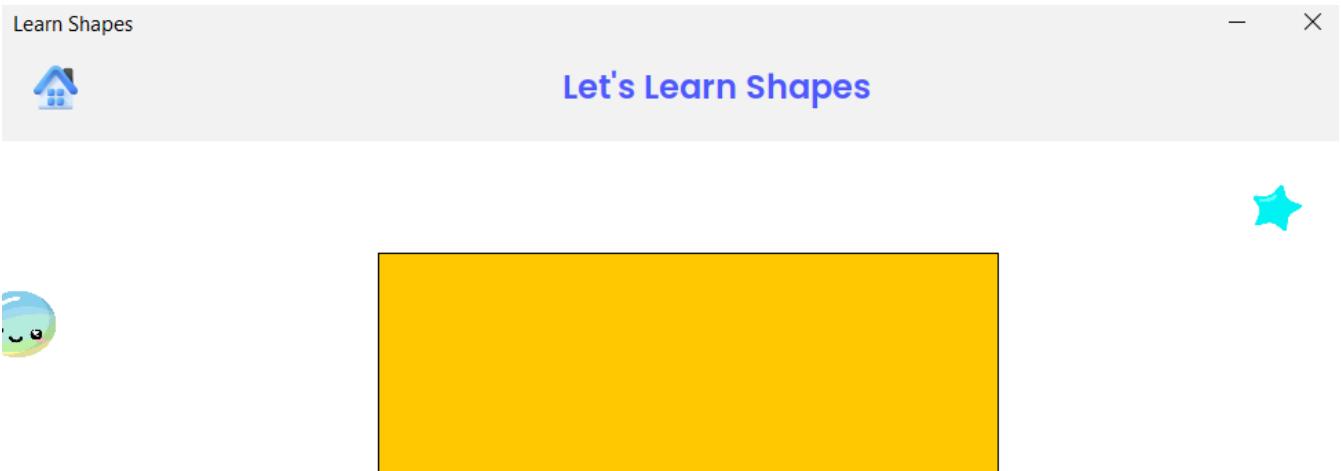


Figure 41: Circle

The screenshot shows the IntelliJ IDEA interface with the following details:

- File Structure:** The left sidebar shows the project structure under "main-coursework-phoenix". The "src" folder contains "Shapes", which further contains "CirclePanel" and other classes like CubePanel, EarthPanel, MarsPanel, etc.
- Code Editor:** The main window displays the content of CirclePanel.java. The code defines a `CirclePanel` class extending `JPanel`. It includes fields for width, height, radius, and center coordinates, and a `paintComponent` method.
- Toolbars and Status Bar:** The top bar has standard menu items like File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, Git, Window, Help, and a status message "main-coursework-phoenix - CirclePanel.java". The bottom bar shows tabs for Git, TODO, Problems, Terminal, Services, and Profiler. The status bar at the bottom right shows "22:52 CRLF UTF-8 4 spaces Sandaru".

Figure 42: Source Code of Circle



I'm a Rectangle

I have 4 sides

2 long ones and 2 short ones



Next



Figure 43: Rectangle

The screenshot shows an IDE interface with the file "main-coursework-phoenix - RectanglePanel.java" open. The code defines a class "RectanglePanel" that extends JPanel. It contains variables for width and height, calculates rectangleWidth and rectangleHeight as half of the respective dimensions, and determines the center coordinates x and y. It also includes a private variable for selected color and an overridden paintComponent method that sets the color to orange and calls super.paintComponent(g). The code is annotated with comments and imports.

```

package Renderables.Shapes;
import ...;

public class RectanglePanel extends JPanel {
    int width = getWidth();
    int height = getHeight();
    int rectangleWidth = width / 2;
    int rectangleHeight = height / 2;
    int x = (width - rectangleWidth) / 2;
    int y = (height - rectangleHeight) / 2;

    private Color selectedColor=Color.ORANGE;

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        g=g;
        width = getWidth();
        height = getHeight();
        rectangleWidth = width / 2;
    }
}

```

Figure 44: Source Code of Rectangle

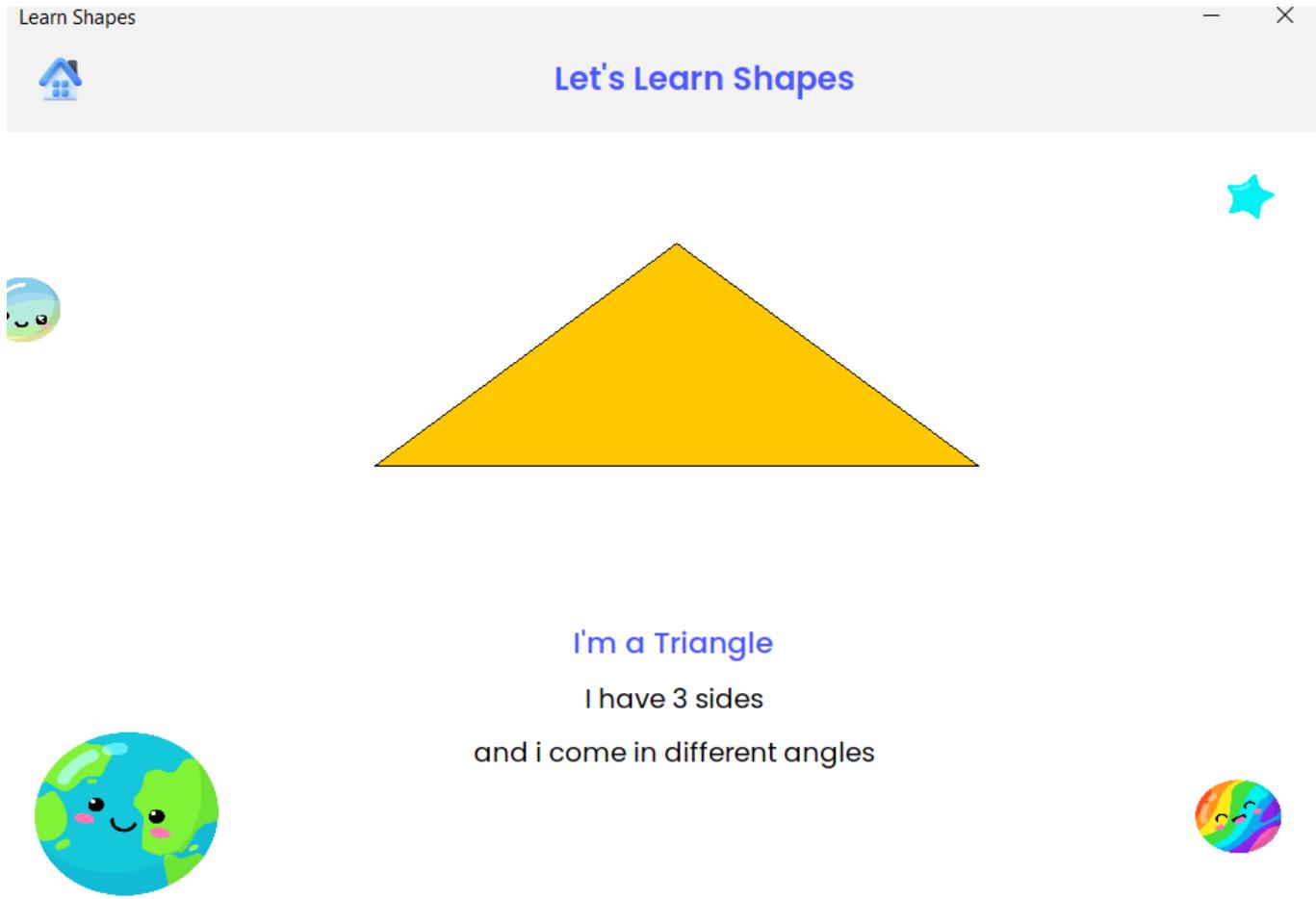


Figure 45: Triangle

```

package Renderbles.Shapes;
import ...;

public class TrianglePanel extends JPanel {
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        int width = getWidth();
        int height = getHeight();

        int newWidth = width / 2;
        int newHeight = height / 2;

        int[] xPoints = {(newWidth / 2)+newWidth/2, newWidth+newWidth/2, 0+newWidth/2};
        int[] yPoints = {0+newHeight/2, newHeight+newHeight/2, newHeight+newHeight/2};

        // Set the color and fill the triangle
        g.setColor(Color.ORANGE);
        g.fillPolygon(xPoints, yPoints, nPoints: 3);

        // Set the color and draw the triangle outline
        g.setColor(Color.BLACK);
        g.drawPolygon(xPoints, yPoints, nPoints: 3);
    }
}

```

Figure 46: Source Code of Triangle

5.5 – Match Shapes

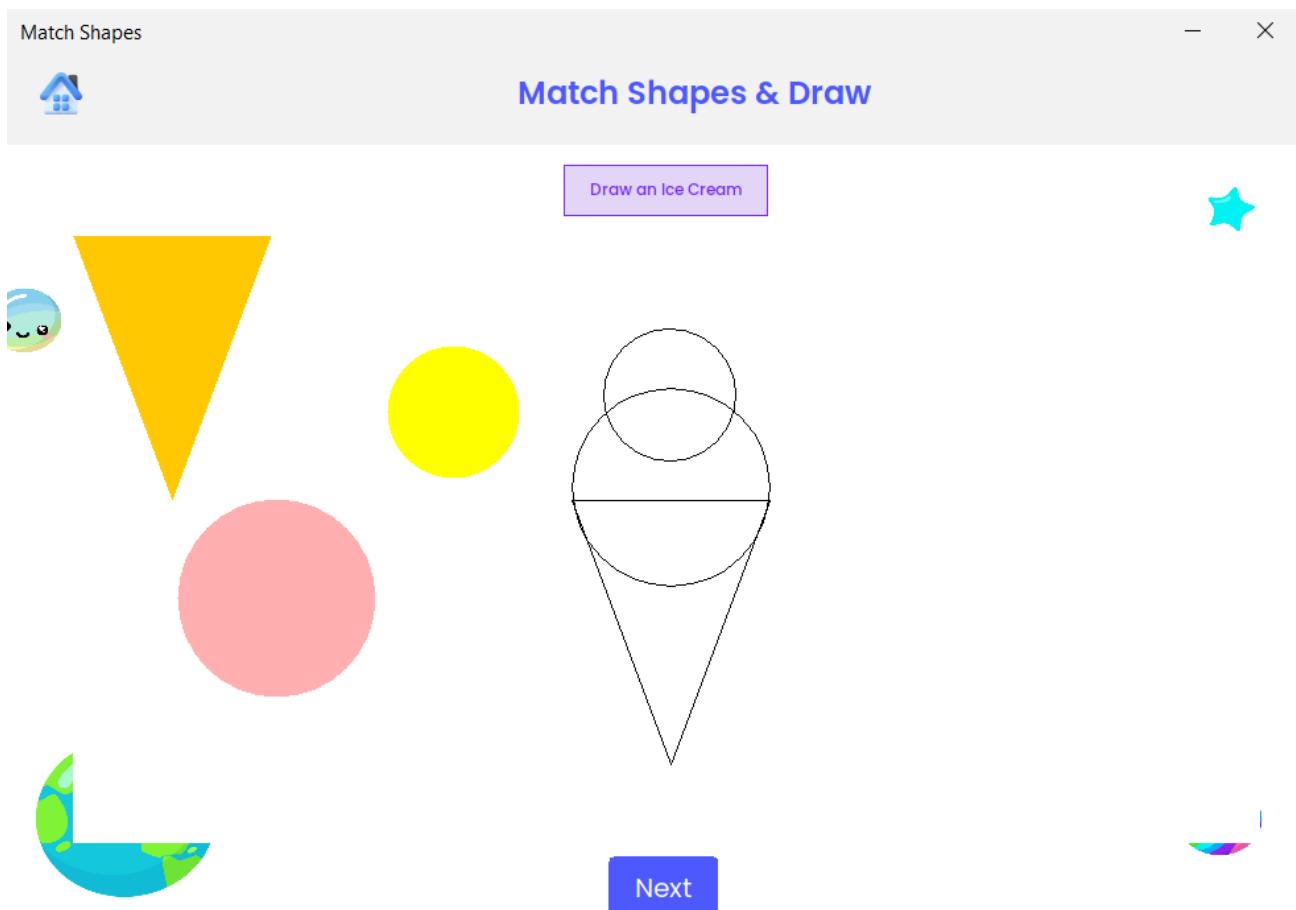


Figure 47: Create Ice Cream

The screenshot shows the IntelliJ IDEA interface with the following details:

- File Menu:** File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, Git, Window, Help.
- Project Bar:** main-coursework-phoenix [HCI_Coursework].
- Code Editor:** The file `MSIceCream.java` is open. The code defines a class `MSIceCream` that extends `JPanel`. It contains fields for a square (`private Rectangle square`), two ovals (`private Ellipse2D oval` and `oval2`), and a triangle (`private Polygon triangle`). The code includes several usage annotations (e.g., 1 usage, 11 usages) and a note about the `JPanel` class being a container.
- Toolbars and Status Bar:** Includes icons for Git, TODO, Problems, Terminal, Services, and Profiler. The status bar at the bottom shows 192.2 CRLF, UTF-8, 4 spaces, and the name Sandaru.

Figure 48: Source Code of Create Ice Cream

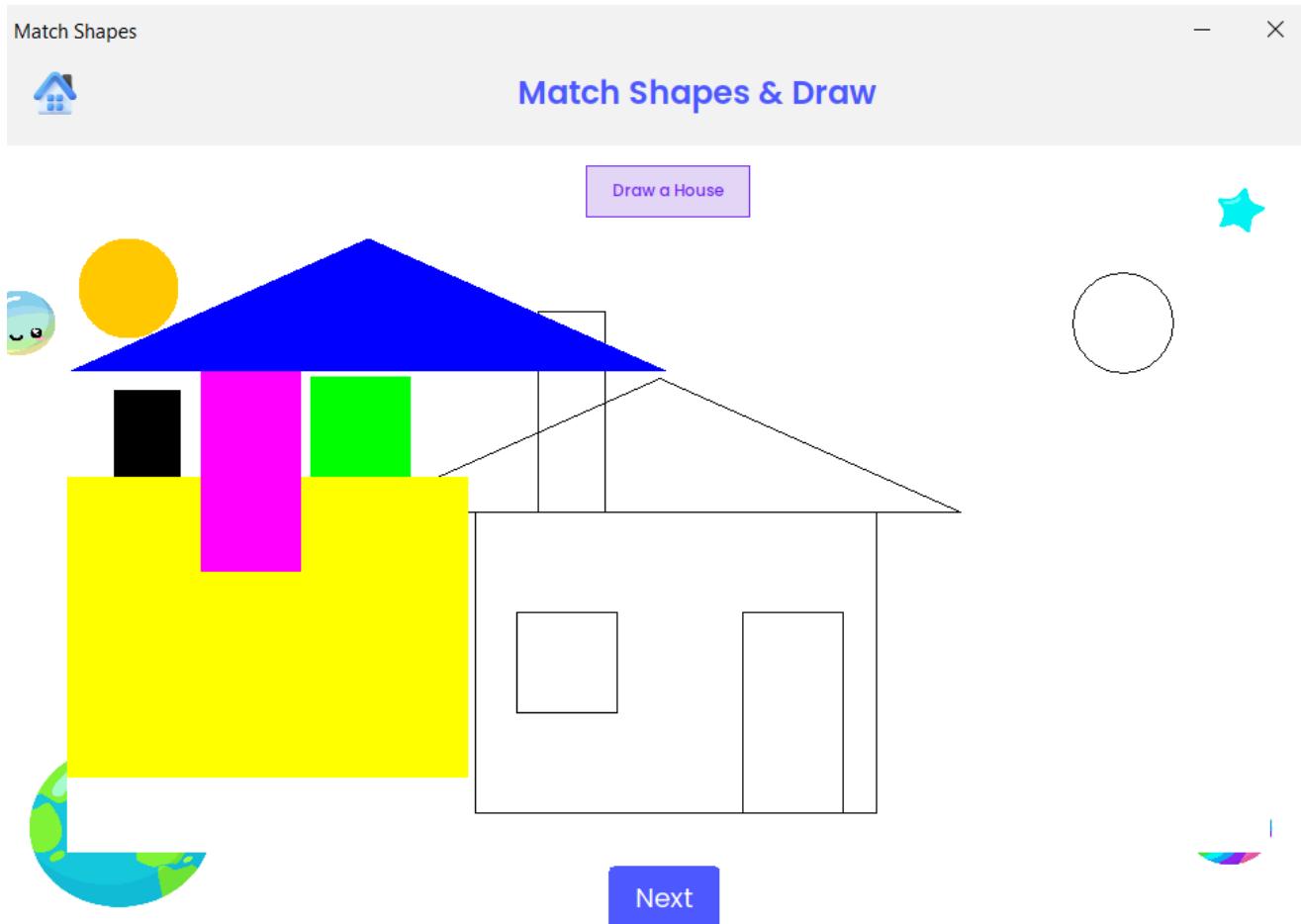


Figure 49: Create House

```
package Renderbles;
import ...;

public class MSHouse extends JPanel { //---->defines a class named Square that extends the JPanel class. The JPanel class is a container
{
    private static final int WINDOW_WIDTH = 800;
    private static final int WINDOW_HEIGHT = 600;

    private Rectangle square2;
    private Rectangle square3;
    private Rectangle square4;
    private Rectangle square5;
    private Ellipse2D oval; //head

    private Polygon triangle;

    //for draw
    private Rectangle triangle2;
}
```

Figure 50: Source Code of Create House

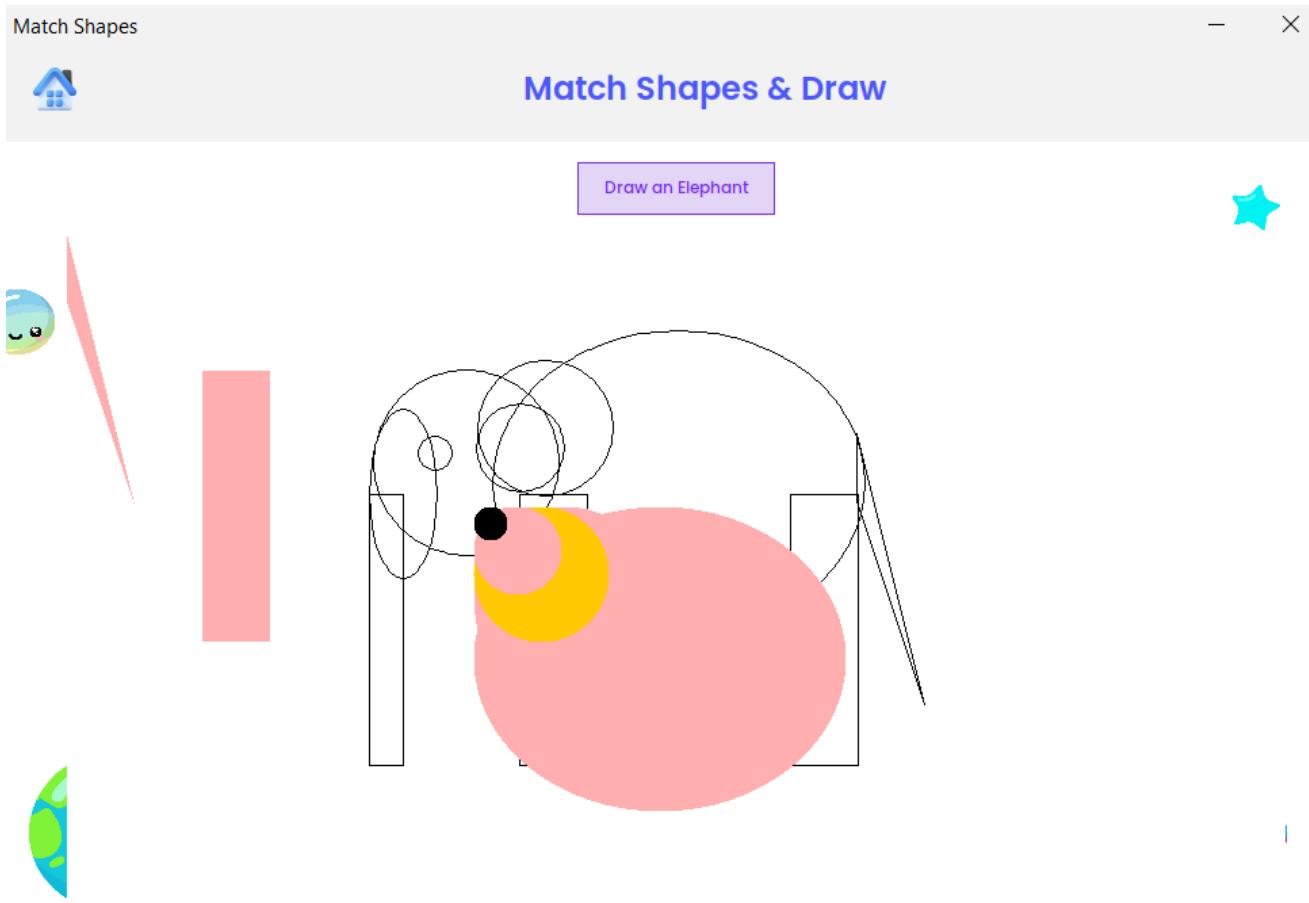


Figure 51: Create Elephant

```

File Edit View Navigate Code Behavior Build Run Tools Git Window Help main-coursework-phoenix - MSElephant.java
main-coursework-phoenix [HCI_Coursework] Project MarsPanel.java SolarSystem.java MatchShapesElephant.java MarsBttn.java Login.java SignUp.java MSElephant.java MyRender2.java
src out
Font Images Renderables Shapes MSElephant MSHouse MSCream SolarSystem BackgroundPanel BlurredBg ClearBttn ClickedEarthBttn ClickedGalaxyBttn ClickedMarsBttn ClickedMoonBttn Correct DrawCircle DrawLine DrawRectangle EarthBttn GalaxyBttn GradientButton HomeBttn MarsBttn MoonBttn MyRender2 MyRenderer NameLabel RoundedTextField StyledToggleButton ColorCircle
1 package Renderables;
2
3 import javax.swing.*;
4
5 public class MSElephant extends JPanel { //---->defines a class named Square that extends the JPanel class. The JPanel class is a container for other components.
6
7     private static final int WINDOW_WIDTH = 800;
8
9     private static final int WINDOW_HEIGHT = 600;
10
11     private Rectangle square1;
12
13     private Rectangle square2;
14
15     private Rectangle square3;
16
17     private Rectangle square4;
18
19     private Ellipse2D oval1; //body
20
21     private Ellipse2D oval2; //head
22
23     private Ellipse2D oval3; //ear
24
25     private Ellipse2D oval4; //back
26
27     private Ellipse2D oval5; //eye
28
29
30 }

```

Figure 52: Source Code of Create Elephant

5.6 – Draw Shapes

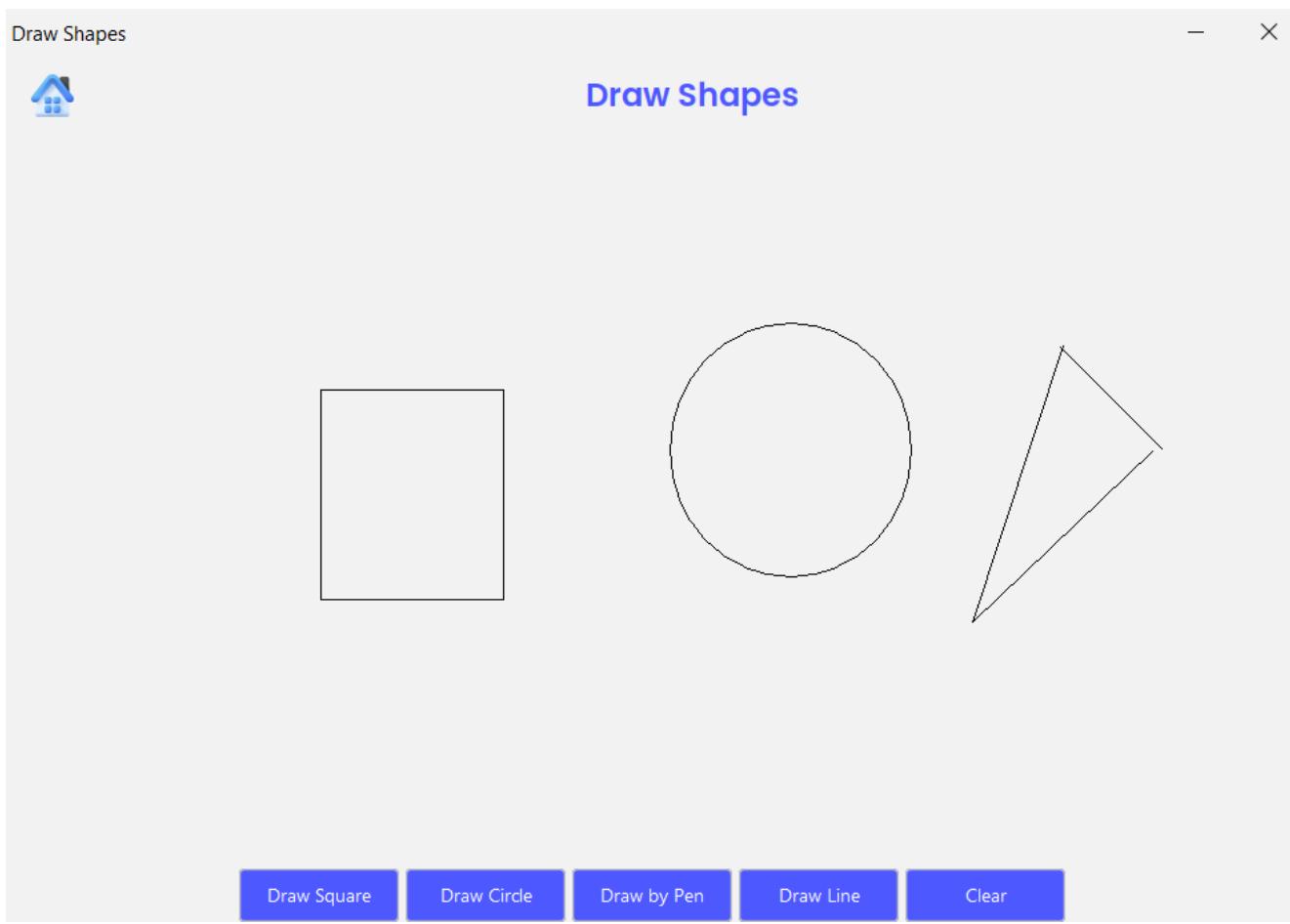


Figure 53: Draw Shapes

A screenshot of an IDE (IntelliJ IDEA) showing the source code for "DrawShapes.java". The code defines a class "DrawShapes" that extends "JFrame". It includes fields for buttons ("squareButton", "circleButton", "penButton", "lineButton", "clearButton") and a panel ("shapePanel"). The constructor sets the title to "Draw Shapes", sets the size to 1000x700, and makes the frame non-resizable. A code editor on the right shows the implementation of the constructor.

Figure 54: Source Code of Draw Shapes

5.7 – Color Shapes

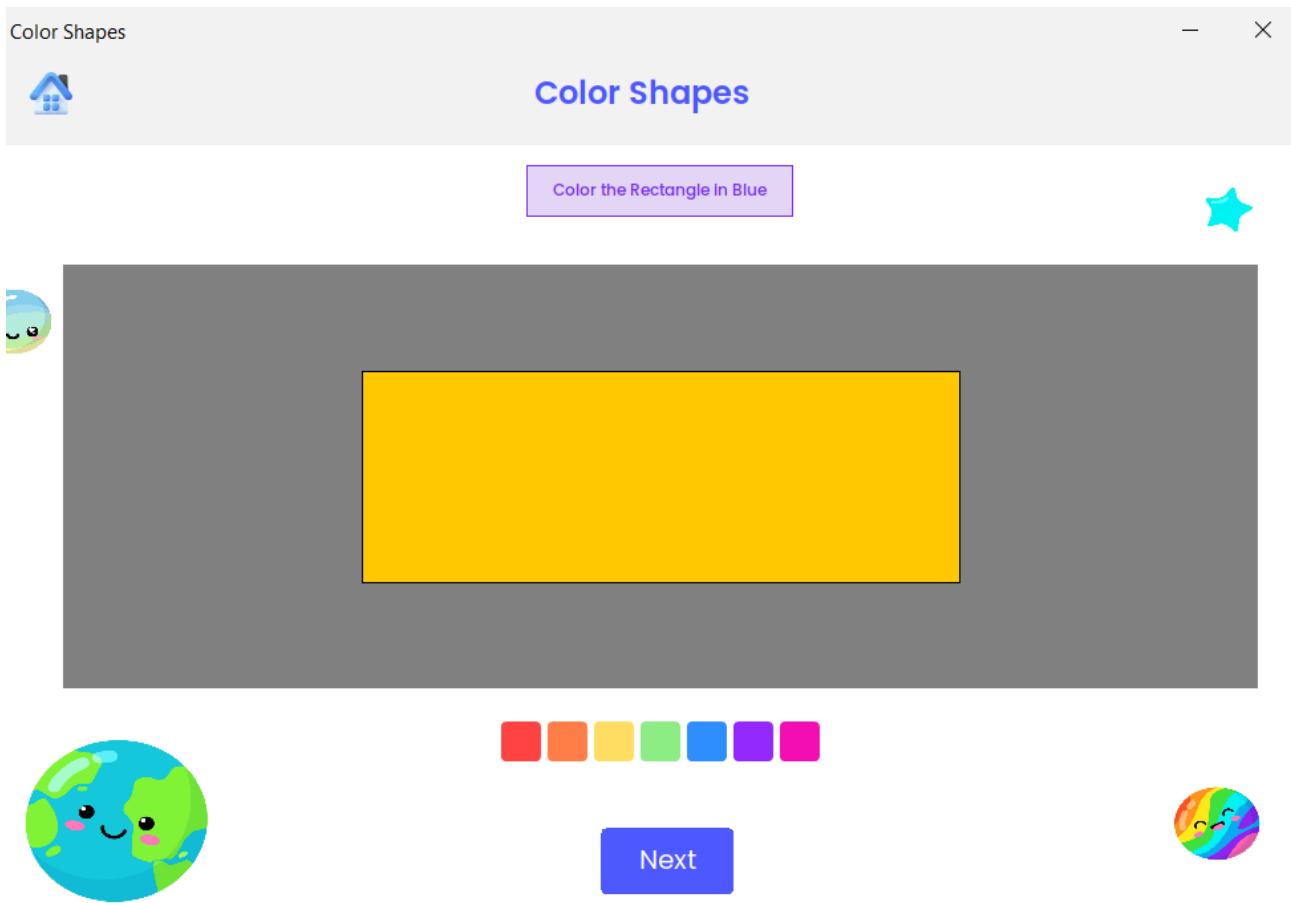


Figure 55: Color Rectangle

The screenshot shows an IDE interface with the file "ColorRectangle.java" open. The code is as follows:

```
File Edit View Navigate Code Refactor Build Run Tools Git Window Help main-coursework-phoenix - ColorRectangle.java
main-coursework-phoenix > src > ColorRectangle > frame
Project Pull Requests Bookmarks Structure
import ...
5 usages ▲ Bhagya1024 +1
public class ColorRectangle extends JFrame {
    3 usages
    private BufferedImage backgroundImage;
    private Font headingFont;
    3 usages
    private static JFrame frame;
    1 usage ▲ Bhagya1024
    public ColorRectangle() {
        setTitle("Color Shapes");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize( width: 1000, height: 700 );
        setResizable(false);
        setLocationRelativeTo(null);
        frame=this;
        setVisible(true);
    }
    ▲ Bhagya1024 +1
    public static void createAndShowGUI() {
        SwingUtilities.invokeLater(() -> {
            ColorRectangle colorRectangle = new ColorRectangle();
            colorRectangle.setVisible(true);
            colorRectangle.loadBackgroundImage( imagePath: "Images/learnbg.png" );
            JLabel headingLabel = new JLabel( text: "Color Shapes" );
            headingLabel.setFont(loadFont( path: "Font/poppinssemibold.ttf").deriveFont(Font.PLAIN, size: 24));
            headingLabel.setForeground(new Color( int: 0x2A5AFF));
        });
    }
}
All files are up-to-date (9 minutes ago)
```

Figure 56: Source Code of Color Rectangle

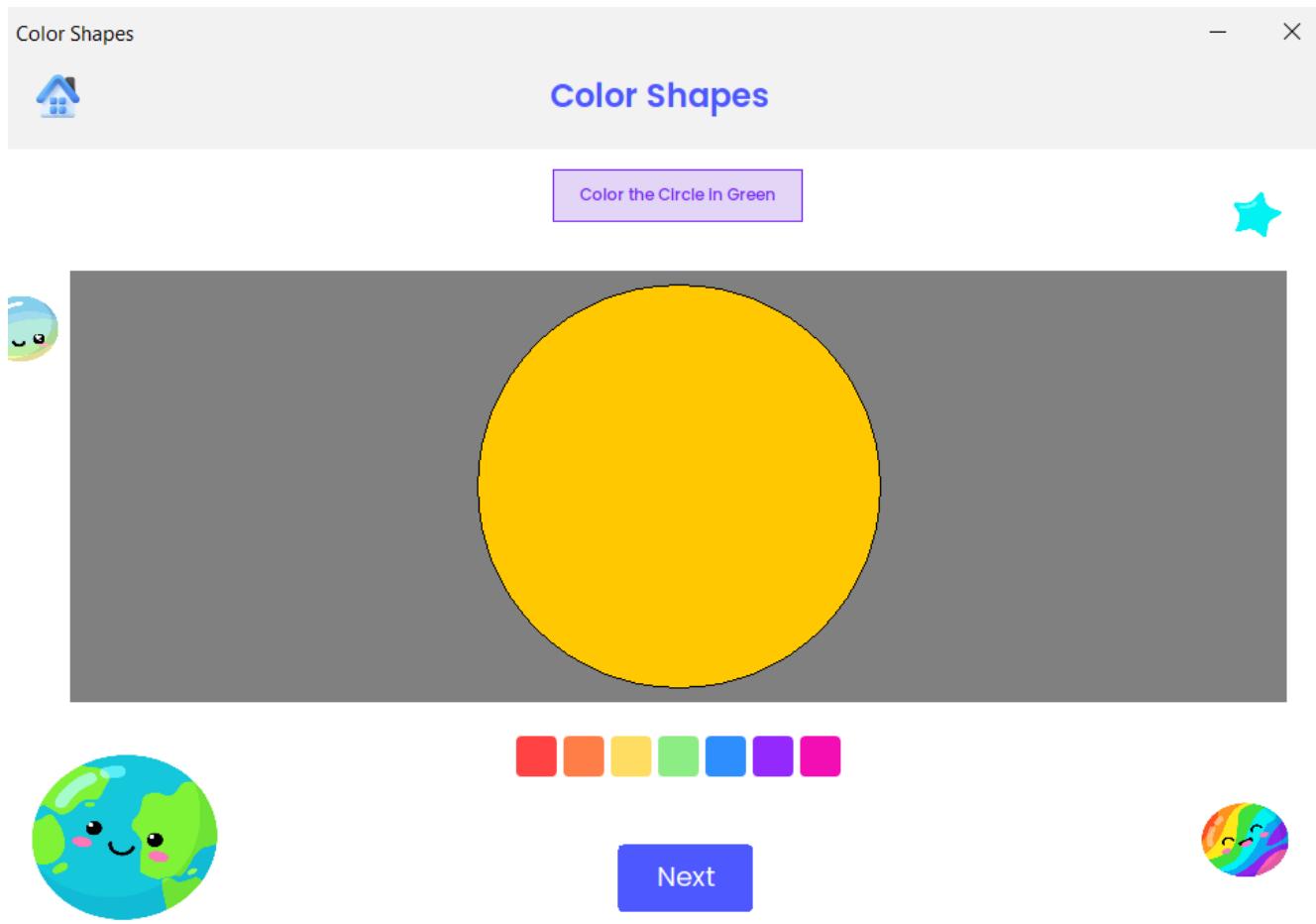


Figure 57: Color Circle

```

File Edit View Navigate Code Refactor Build Run Tools Git Window Help main-coursework-phoenix - ColorCircle.java
main-coursework-phoenix src ColorCircle
Project Commits Pull Requests Bookmarks Structure
ClickedMoonBtn 1 import ...
Correct 12
DrawCircle 13 public class ColorCircle extends JFrame {
DrawLine 14
DrawRectangle 15
EarthBtn 16
GalaxyBtn 17
GradientButton 18
HomeBtn 19
MarsBtn 20
MoonBtn 21
MyRender2 22
MyRenderer 23
NameLabel 24
RoundedTextField 25
StyledToggleButton 26
ColorCircle 27
ColorCircle.form 28
ColorRectangle 29
ColorTriangle 30
CubeObject 31
DrawShapes 32
Earth 33
Home 34
Home.form 35
LearnShapesCircle 36
LearnShapesRectangle 37
LearnShapesSquare 38
LearnShapesTriangle 39
Login 40
Mars 41
MatchShapesElephant 42
MatchShapesHouse 43
Label headingLabel = new JLabel("Color Shapes");
private BufferedImage backgroundImage;
private Font headingFont;
private static JFrame frame;
public Colorcircle() {
    setTitle("Color Shapes");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setSize( width 1000, height 700 );
    setResizable(false);
    setLocationRelativeTo(null);
    frame=this;
    setVisible(true);
}
public static void createAndShowGUI() {
    SwingUtilities.invokeLater(() -> {
        ColorCircle colorCircle = new ColorCircle();
        colorCircle.setVisible(true);
        colorCircle.loadBackgroundImage( imagePath: "Images/learnbg.png" );
    });
}

```

Figure 58: Source Code of Color Circle

5.8 – 3D Objects

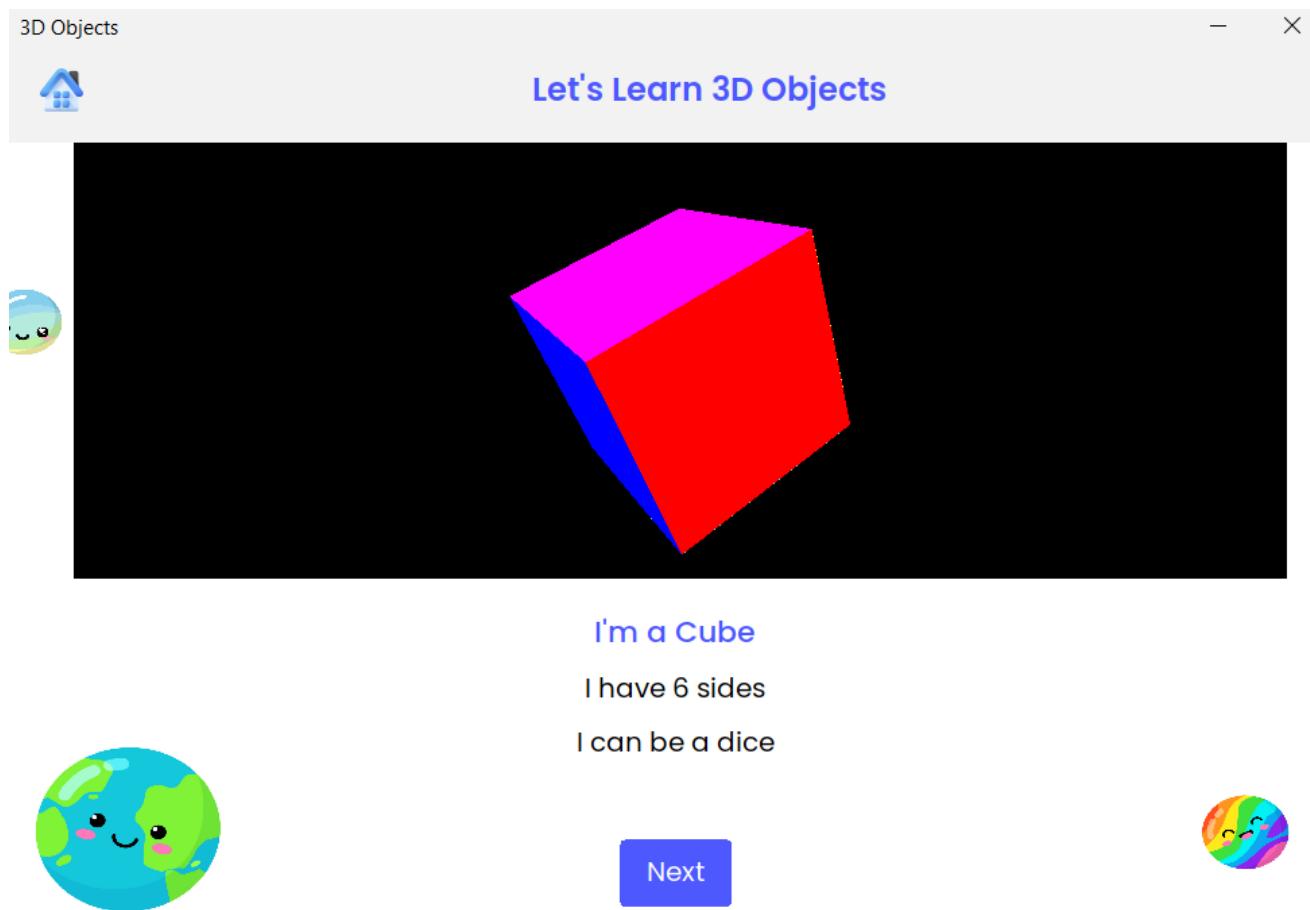


Figure 59: Cube Object

A screenshot of an IDE (IntelliJ IDEA) showing the source code for `CubePanel.java`. The code implements the `GLEventListener` interface to handle OpenGL events. It initializes the OpenGL context, sets clear color, depth, and hint parameters, and overrides the `init` and `dispose` methods.

```
package Renderables.Shapes;  
import ...  
  
public class cubePanel implements GLEventListener {  
    private GLU glu;  
    private float rotationAngle = 0.0f;  
    private long startTime;  
  
    public void init(GLAutoDrawable drawable) {  
        GL2 gl = drawable.getGL().getGL2();  
        glu = new GLU();  
        startTime = System.currentTimeMillis();  
        gl.glClearColor(0.0f, 0.0f, 0.0f, 0.0f);  
        gl.glClearDepth(1.0f);  
        gl.glenable(GL2.GL_DEPTH_TEST);  
        gl.glDepthFunc(GL2.GL_LESS);  
        gl.glHint(GL2.GL_PERSPECTIVE_CORRECTION_HINT, GL2.GL_NICEST);  
    }  
  
    public void dispose(GLAutoDrawable drawable) {}  
}
```

Figure 60: Source Code of Cube Object



I'm a Sphere

I'm round

I look like the Moon



Figure 61: Sphere Object

```

File Edit View Navigate Code Refactor Build Run Tools Git Window Help main-coursework-phoenix - SpherePanel.java
main-coursework-phoenix > src > Renders > Shapes > SpherePanel
Project CirclePanel.java CubePanel.java EarthPanel.java MarsPanel.java RectanglePanel.java SpherePanel.java MoonPanel.java MSHouse.java MSIceCream.java SolarSystem.java BackgroundPanel.java BlurredBg.java ClearBtn.java ClickedEarthBtn.java ClickedGalaxyBtn.java ClickedMarsBtn.java ClickedMoonBtn.java Correct.java DrawCircle.java DrawLine.java DrawRectangle.java EarthBtn.java GalaxyBtn.java GradientButton.java
main-coursework-phoenix [HCI Coursework]
> idea
> out
src
> Font
> Images
Renders
Shapes
CirclePanel
CubePanel
EarthPanel
MarsPanel
MoonPanel
RectanglePanel
SpherePanel
SquarePanel
TrianglePanel
MSElephant
MSHouse
MSIceCream
SolarSystem
BackgroundPanel
BlurredBg
ClearBtn
ClickedEarthBtn
ClickedGalaxyBtn
ClickedMarsBtn
ClickedMoonBtn
Correct
DrawCircle
DrawLine
DrawRectangle
EarthBtn
GalaxyBtn
GradientButton
SpherePanel.java
public package Renderables.Shapes;
import ...
public class SpherePanel implements GLEventListener {
    private GLU glu;
    private GLUT glut;
    float rotateAngle=5f;
    @Override
    public void init(GLAutoDrawable drawable) {
        GL2 gl = drawable.getGL().getGL2();
        glu = new GLU();
        glut = new GLUT();
        gl.glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
        gl.glClearDepth(1.0f);
        gl.glEnable(GL2.GL_DEPTH_TEST);
        gl.glDepthFunc(GL2.GL_LESS);
        gl.glHint(GL2.GL_PERSPECTIVE_CORRECTION_HINT, GL2.GL_NICEST);
        // Enable lighting
        gl.glEnable(GL2.GL_LIGHTING);
        gl.glEnable(GL2.GL_LIGHT0);
        // Define light source properties
        float[] lightPosition = {0.0f, 5.0f, 1.0f}; // Position of the light source
        float[] lightAmbient = {0.2f, 0.2f, 0.2f, 1.0f}; // Ambient light color
    }
}

```

Figure 62: Source Code of Sphere Object

5.9 – 3D Space

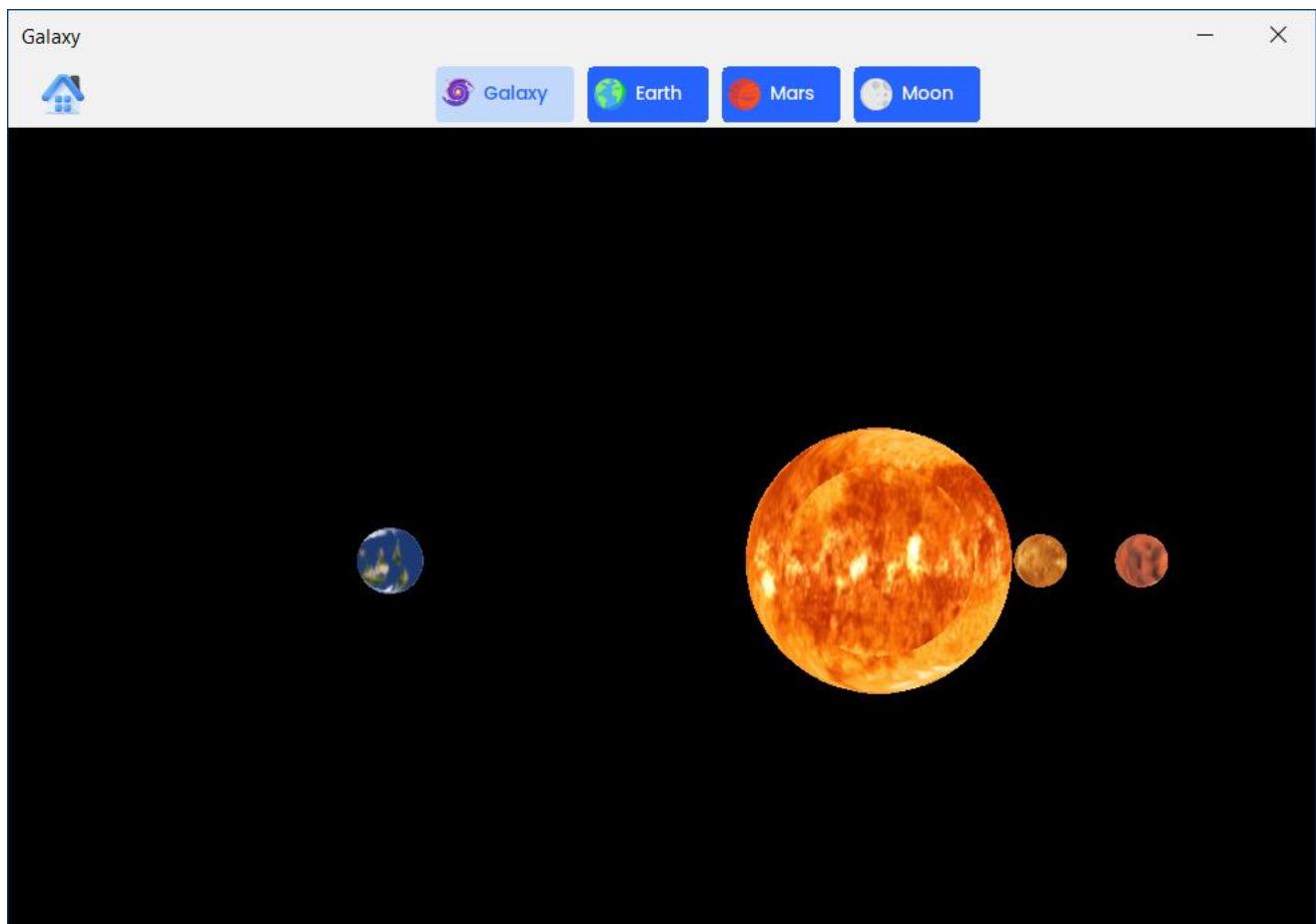


Figure 63: 3D Galaxy

A screenshot of an IDE (IntelliJ IDEA) showing the source code for `SolarSystem.java`. The code implements a `GLEventListener` interface and defines several static final variables for planet radii and distances. The code is as follows:

```
package Renderables;
import ...;

public class SolarSystem implements GLEventListener {
    private static final int WINDOW_WIDTH = 800;
    private static final int WINDOW_HEIGHT = 600;
    private static final float SUN_RADIUS = 100.0f;
    private static final float PLANET1_RADIUS = 15.0f;
    private static final float PLANET2_RADIUS = 20.0f;
    private static final float PLANET3_RADIUS = 25.0f;
    private static final float PLANET4_RADIUS = 20.0f;
    private static final float PLANET5_RADIUS = 75.0f;
    private static final float PLANET6_RADIUS = 65.0f;
    private static final float PLANET1_DISTANCE = 200.0f;
    private static final float PLANET2_DISTANCE = 300.0f;
```

Figure 64: Source Code of 3D Galaxy

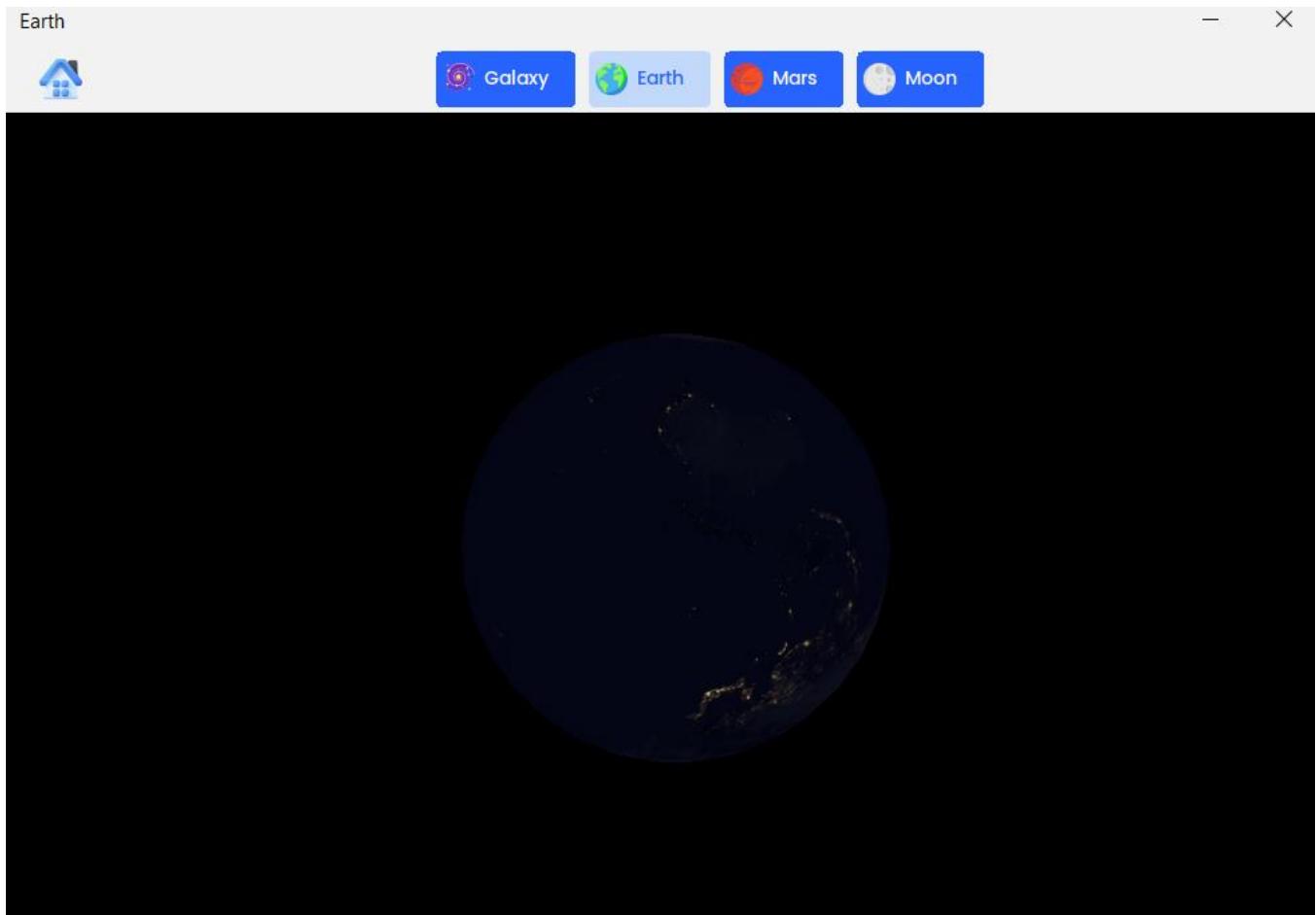


Figure 65: 3D Earth

```
File Edit View Navigate Code Behavior Build Run Tools Help main-coursework-phoenix - EarthPanel.java
main-coursework-phoenix > src > Renderables > Shapes > EarthPanel > init
Project > main-coursework-phoenix [HCI_Coursework] > src > Renderables > Shapes > EarthPanel > init
1 package Renderables.Shapes;
2
3 import ...
4
5 3 usages ▾ Nadeesha Nuwanthika *
6 public class EarthPanel implements GLEventListener {
7     3 usages
8         private GLU glu;
9         1 usage
10        private GLUT glut;
11        2 usages
12        float rotateAngle=5f;
13
14        4 usages
15        private Texture texture;
16        ▾ Nadeesha Nuwanthika *
17        @Override
18        public void init(GLAutoDrawable drawable) {
19            GL2 gl = drawable.getGL().getGL2();
20            glu = new GLU();
21            glut = new GLUT();
22
23            try {
24                texture = TextureIO.newTexture(new File("C:/Users/user/Downloads/img/2k_earth_nightmap.jpg"), b: true);
25            } catch (IOException e) {
26                throw new RuntimeException(e);
27            }
28            gl.glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
29            gl.glClearDepth(1.0f);
30            gl glEnable(GL2.GL_DEPTH_TEST);
31            gl glDepthFunc(GL2.GL_LESS);
32            glHint(GL2.GL_PERSPECTIVE_CORRECTION_HINT, GL2.GL_NICEST);
33
34        }
35    }
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
59
```

Figure 66: Source Code of 3D Earth

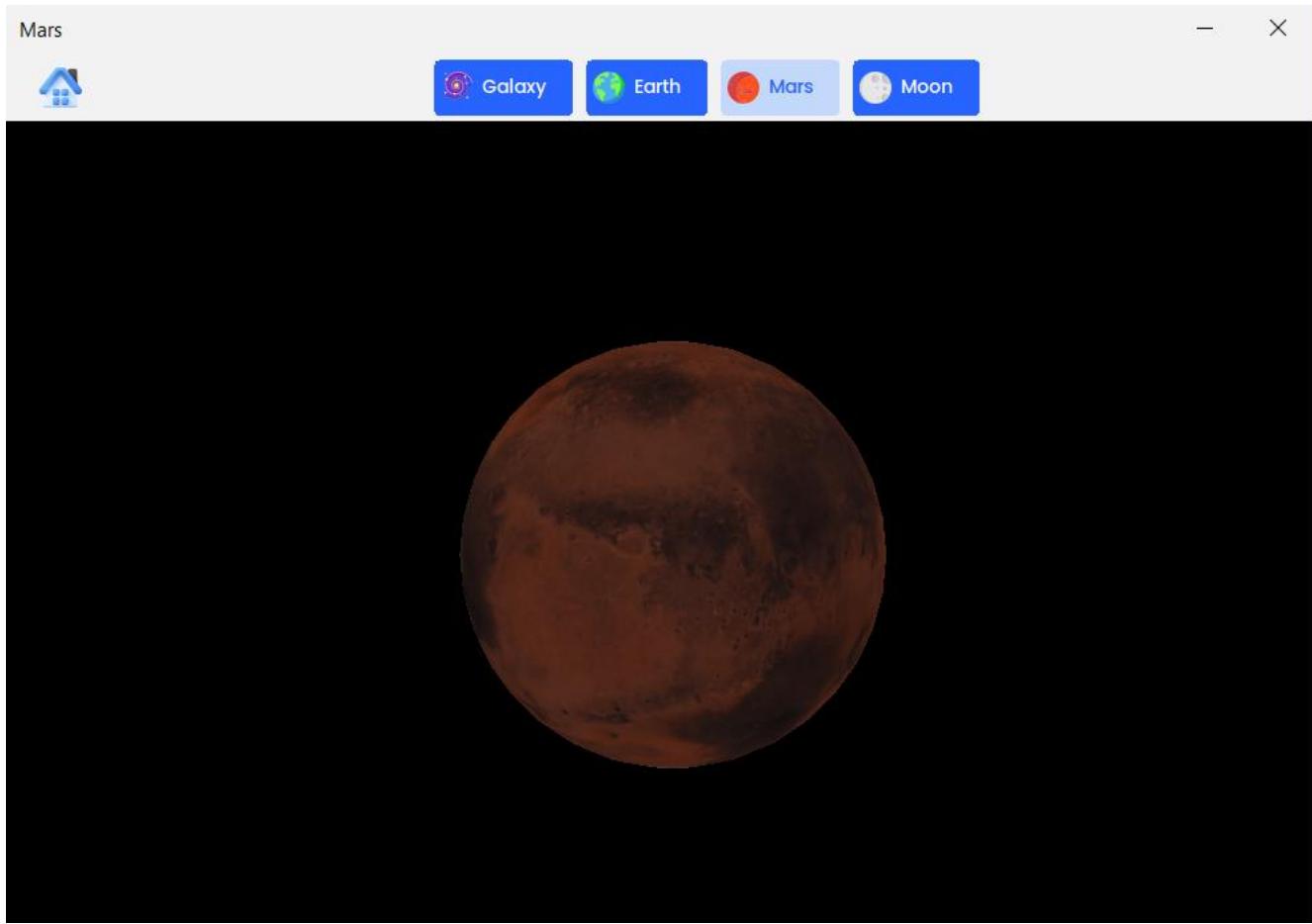


Figure 67: 3D Mars

The screenshot shows the IntelliJ IDEA interface with the MarsPanel.java file open in the editor. The code implements the GLEventListener interface and initializes OpenGL components like GLU and GLUT. It also loads a texture for the Mars panel.

```
package Renderables.Shapes;

import ...

public class MarsPanel implements GLEventListener {
    private GLU glu;
    private GLUT glut;
    float rotateAngle=5f;

    private Texture texture;
    @Override
    public void init(GLAutoDrawable drawable) {
        GL2 gl = drawable.getGL().getGL2();
        glu = new GLU();
        glut = new GLUT();

        try {
            texture = TextureIO.newTexture(new File("C:/Users/user/Downloads/img/2k_mars.jpg"), true);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
        gl.glClearColor( 0.0f, 0.0f, 0.0f, 0.0f );
        gl.glClearDepth( 1.0f );
        gl glEnable(GL2.GL_DEPTH_TEST);
        gl.depthFunc(GL2.GL_EQUAL);
        gl.hint(GL2.GL_PERSPECTIVE_CORRECTION_HINT, GL2.GL_NICEST);
    }

    public void display(GLAutoDrawable drawable) {
        glut.reshape(drawable.getWidth(), drawable.getHeight());
        glut.timerFunc(10, this);
        glut.timer(10);
        glut.postRedisplay();
    }

    public void reshape(GLAutoDrawable drawable, int width, int height) {
        glu.gluOrtho2D(0, width, 0, height);
    }

    public void displayChanged(GLAutoDrawable drawable, int mode, int flags) {
        if ((flags & GL2.GL_OPENGL_3D) != 0) {
            glut.reshape(drawable.getWidth(), drawable.getHeight());
        }
    }
}
```

Figure 68: Source Code of 3D Mars

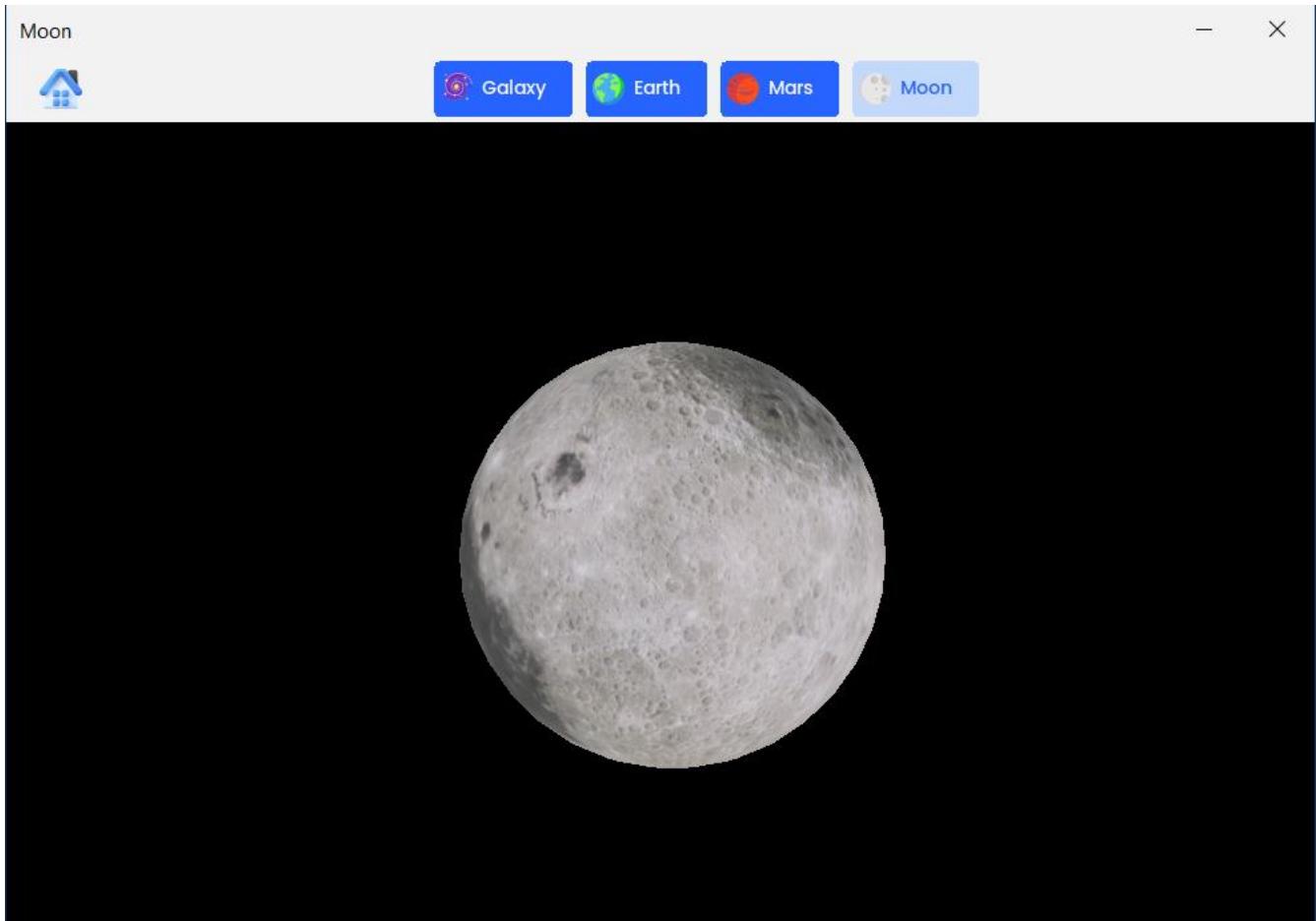
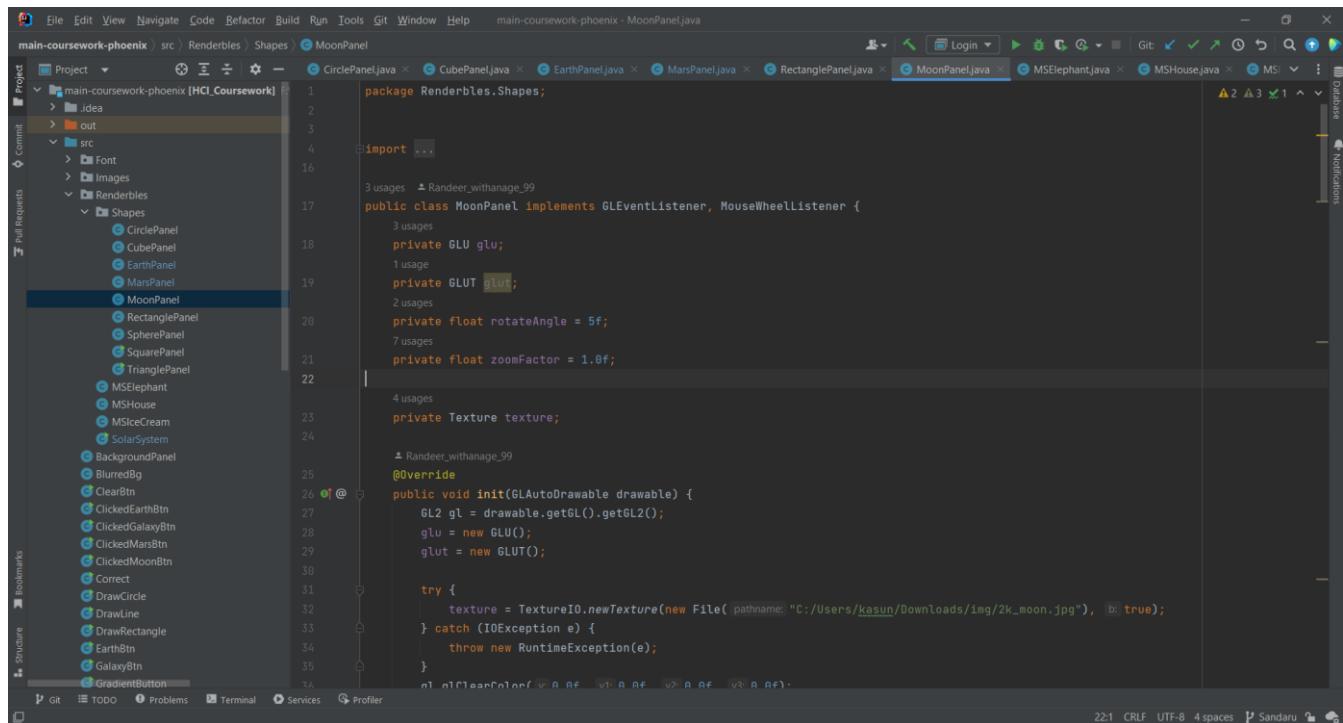


Figure 69: 3D Moon



The screenshot shows an IDE interface with the following details:

- Project:** main-coursework-phoenix
- File:** MoonPanel.java
- Code Preview:** The code is for a class named MoonPanel, which implements GLEventListener and MouseWheelListener. It includes imports for GLU and GLUT, and defines variables for rotateAngle (5f) and zoomFactor (1.0f). It also initializes a Texture named texture.
- Toolbars and Menus:** Standard IDE menus like File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, Git, Window, Help are visible.
- Sidebar:** Shows a file tree with packages like Renderables and sub-packages like Shapes, containing various panel classes (CirclePanel, CubePanel, EarthPanel, MarsPanel, MoonPanel, RectanglePanel, SpherePanel, SquarePanel, TrianglePanel) and other utility classes.
- Status Bar:** Shows file statistics (221 CRLF, 4 spaces, etc.) and user information (Sandaru).

Figure 70: Source Code of 3D Moon

Chapter 06 – Evaluation

- Participant 1 – Rajapaksha Rajapaksha

The implementation of the JOGL (Java OpenGL) library serves as a fundamental tool for rendering both two-dimensional and three-dimensional graphics. In order to achieve an authentic representation of the solar system, the technique of texture mapping has been employed. This method entails the application of images, referred to as textures, onto the surfaces of three-dimensional entities. By utilizing high-resolution textures capturing the characteristics of celestial bodies, ranging from the sun to the planets, the visual quality has been significantly enhanced, effectively breathing life into the solar system.

Furthermore, lighting effects have been successfully applied to the objects present within the scene, resulting in a heightened sense of realism. These lighting effects contribute to a more convincing visual appearance of the objects, making them closely resemble their real-life counterparts. By amalgamating these techniques of lighting and shading, a visually captivating and interactive environment has been created, facilitating the exploration of shapes and enabling children to acquire knowledge about their properties.

- Participant 2 – Merenna Amarasinghe

The primary objective in designing the user interface was to create an engaging and child-friendly experience catering specifically to children between the ages of 4 and 6. To facilitate the learning process, vibrant colours and playful graphics were incorporated, aiming to captivate the attention and interest of the children. Simplicity emerged as a key focal point during the development of the user interface. The intentional emphasis on straightforwardness and ease of navigation ensures that even the youngest of learners can effortlessly explore the system. Recognizing the varying reading capabilities of the target audience, particular attention was given to the inclusion of easily identifiable buttons and icons, allowing all users, regardless of their reading proficiency, to access the system seamlessly.

Moreover, a Learn Shapes feature was implemented, and that feature serves as a pedagogical tool to aid students in comprehending two-dimensional shapes. Foundational shapes such as rectangles, squares, triangles, and circles were meticulously defined, accompanied by a clear exposition of their respective properties. In order to enhance the visual appeal and facilitate shape recognition, the Fill method was employed to apply appropriate colours to these shapes.

- Participant 3 – Sakaladhipathige Fernando

The implementation includes a "Match Shapes" feature. The primary objective of incorporating this feature was to foster the identification and correlation of shapes among children, while simultaneously nurturing the development of spatial reasoning skills and critical thinking abilities. This feature offers a diverse range of difficulty levels, ensuring that children can progress at their own individual pace, thus establishing a solid foundation in shape recognition.

Within this feature, two distinct types of shapes are presented: coloured shapes and colourless shapes. Coloured shapes were created utilizing a fixed size and were further enhanced using the fill method. Also, these coloured shapes possess interactive capabilities, permitting them to be manipulated via a mouse onClick listener, thereby facilitating an engaging and interactive user experience. The generation of colourless shapes is achieved through the implementation of the draw method.

- Participant 4 – Randeera Withanage

The incorporation of the 3D objects feature within the application aims to facilitate the instruction of students regarding the characteristics of three-dimensional objects. To achieve this, specific techniques were employed for the cube and sphere objects.

For the cube, the `gluPerspective()` function was utilized to apply perspective projection, generating a visual perception of depth and distance. The structure of the cube was defined by specifying the 3D coordinates of the vertices using the `glVertex3f()` function. To ensure accurate rendering, depth testing was enabled using the `glEnable()` function and the `glDepthFunc()` function, allowing for proper depiction of objects based on their distance from the viewer.

In the sphere, realistic lighting effects were achieved by enabling lighting using the `glEnable()` function for `GL_LIGHTING` and `GL_LIGHT0`. The properties of the light source, including position, ambient, diffuse, and specular colours, were defined using the `glLightfv()` function. This interaction between lighting and the sphere's material properties produced highlights and shadows, enhancing its realism.

- Participant 5 – Udugama Nuwanthika

A drawing tool has been developed to facilitate the creation of shapes, utilizing a list of predetermined shapes and effectively managing user interactions. This was accomplished through the implementation of mouse listeners to capture relevant mouse events, such as pressing, releasing, and dragging.

The Java 2D Graphics library was employed to realize the shape-drawing functionality. This library offers a range of classes, including `Rectangle2D`, `Ellipse2D`, and `Line2D`, which facilitate the creation and manipulation of geometric shapes. The rendering of these shapes on the panel is achieved by utilizing the `Graphics2D` object. Furthermore, the tool incorporates buttons that enable the clearing of the drawing area, thereby providing users with the ability to reset the canvas and start afresh.

- Participant 6 – Bopage Muthumala

The colour shape functionality allows users to fill shapes with their desired colours by utilizing the `fill` method. The chosen colour is selected from a colour palette and applied to the shape accordingly. To ensure the accuracy of colour selection, users are provided with instructions indicating the correct colour to be chosen from the colour palette. If the user selects the correct colour, a message will display saying “Great Job”. However, if user select an incorrect colour, then user is unable to proceed to the next activity, reinforcing the importance of accurate colour identification and selection.

Plymouth ID	Name	Contribution
10749121	Rajapaksha Rajapaksha	<ul style="list-style-type: none"> • 3D Space
10749150	Merenna Amarasinghe	<ul style="list-style-type: none"> • UI Designing & Prototyping • Learn Shapes
10749110	Sakaladhipathige Fernando	<ul style="list-style-type: none"> • Match Shapes • Report Writing
10749185	Randeera Withanage	<ul style="list-style-type: none"> • 3D Objects
10749139	Udugama Nuwanthika	<ul style="list-style-type: none"> • Draw with Shapes
10749145	Bopage Muthumala	<ul style="list-style-type: none"> • Colour Shapes

Figure 71: Workload Matrix

Presentation Video Link - https://liveplymouthac-my.sharepoint.com/:f/g/personal/10749121_students_plymouth_ac_uk/Ei5uOqcX_EtLpaVrlY5Y95IBoaRRZNJKTaE3hhROu0IBw?e=TEgOn4

Chapter 07 – Summary

This study provides an overview of a shape learning application developed for primary school children. It includes detailed information on the background, data gathering methods, design, implementation, and evaluation of the application. The primary goal of the application is to facilitate the learning of shapes in 2D and 3D formats, providing users with the ability to create, visualize, and manipulate shapes while learning fundamental drawing principles. Research, survey, and user feedbacks were utilized to gather data. These techniques were carefully chosen to ensure a comprehensive and reliable collection of relevant information. The design section consists of storyboard, and prototypes. In the prototypes both low-fidelity and high-fidelity prototypes were utilized in the designing phase. The design considerations from a usability perspective to enhance the learning experience. The implementation section showcases the screenshots of source code, which include implementation process and the incorporation of Java Swing for creating the user interfaces and Java 3D for rendering 3D objects. In the evaluation section discusses the participants contribution, methods and techniques used to develop the application.

References

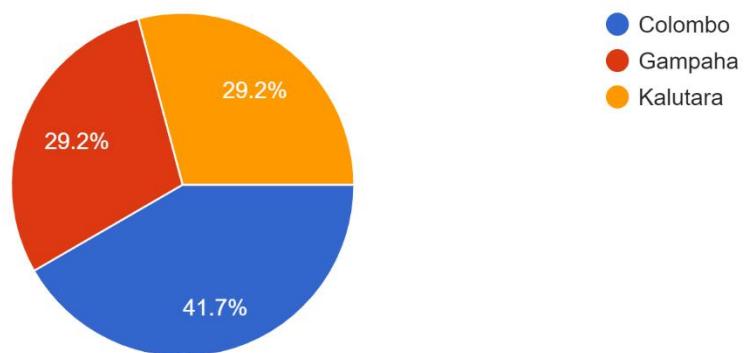
- Tan, T.H., Ahmad Tarmizi, R., Md. Yunus, A.S. and Mohd. Ayub, A.F. (2015). Understanding the Primary School Students' van Hiele Levels of Geometry Thinking in Learning Shapes and Spaces: A Q-Methodology. EURASIA Journal of Mathematics, Science and Technology Education, 11(4). doi:<https://doi.org/10.12973/eurasia.2015.1439a>.
- Thangamani, U. and Eu, L.K. (2018). STUDENTS' ACHIEVEMENT IN SYMMETRY OF TWO DIMENSIONAL SHAPES USING GEOMETER'S SKETCHPAD. MOJES: Malaysian Online Journal of Educational Sciences, [online] 7(1), pp.14–22. Available at: <https://mjlis.um.edu.my/index.php/MOJES/article/view/15448>.
- Hock, T.T., Yunus, A.S.Md., Tarmizi, R.A. and Ayub, A.F.M. (2015). Understanding Primary School teachers' perspectives of teaching and learning in geometry: Shapes and Spaces. [online] IEEE Xplore. doi:<https://doi.org/10.1109/ICREM.2015.7357044>.
- Hascoët, M. (2015). Curricula of HCI and Computer Graphics: From theory to practice. [online] ResearchGate. Available at: <https://www.researchgate.net/publication/255571346>.
- Lewis, M. and Sturdee, M. (2022) 'Curricula Design & Pedagogy for Sketching Within HCI & UX Education' Available at: <https://www.frontiersin.org/articles/10.3389/fcomp.2022.826445/full>

Appendices

Appendix 1 – Responses of Survey

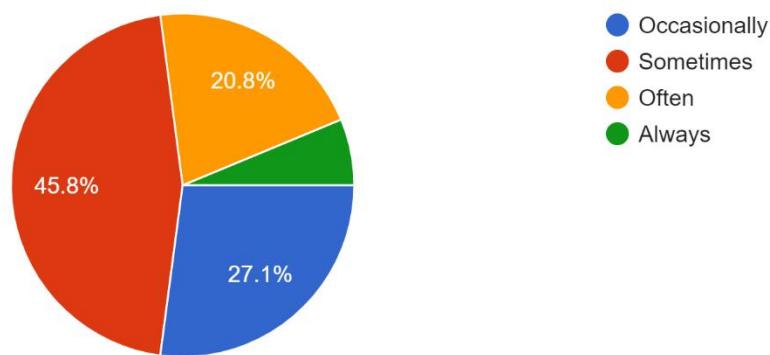
Please select the primary school district in which you are currently teaching.

48 responses



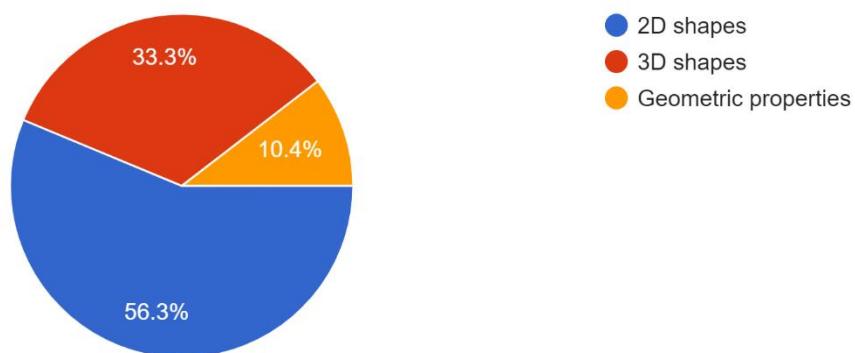
How frequently do you incorporate shape learning activities in your classroom?

48 responses



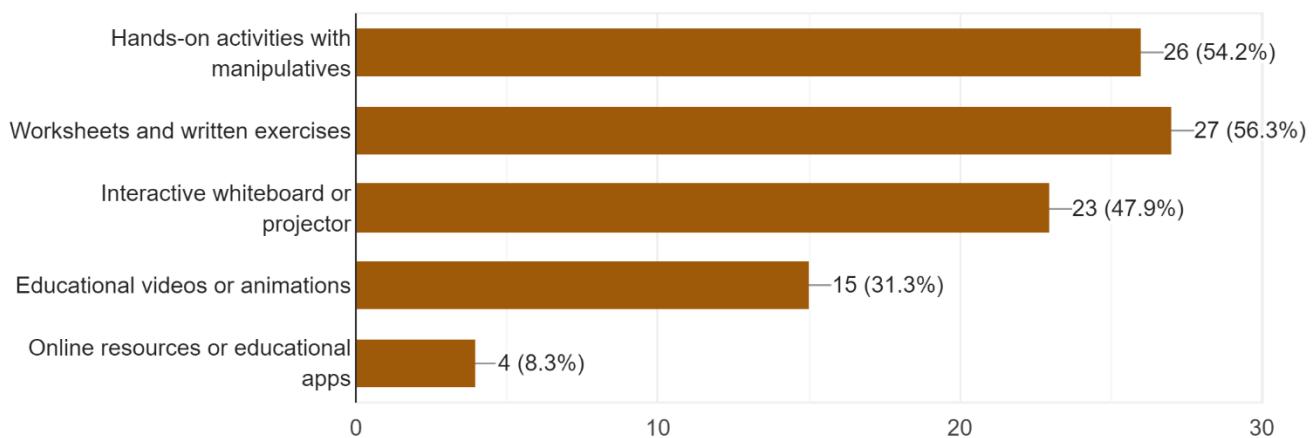
Which shape concepts do you focus on teaching in your classroom?

48 responses



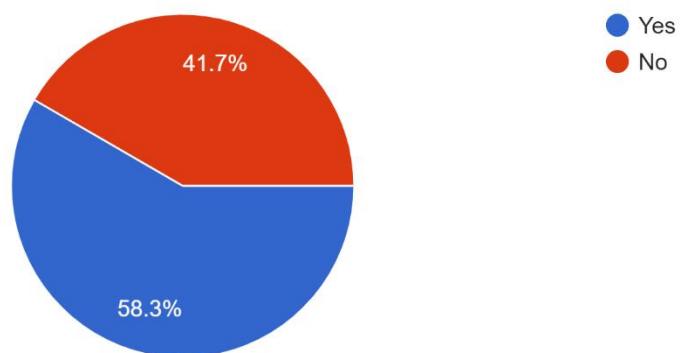
Which teaching methods do you use to introduce and reinforce shape concepts?

48 responses



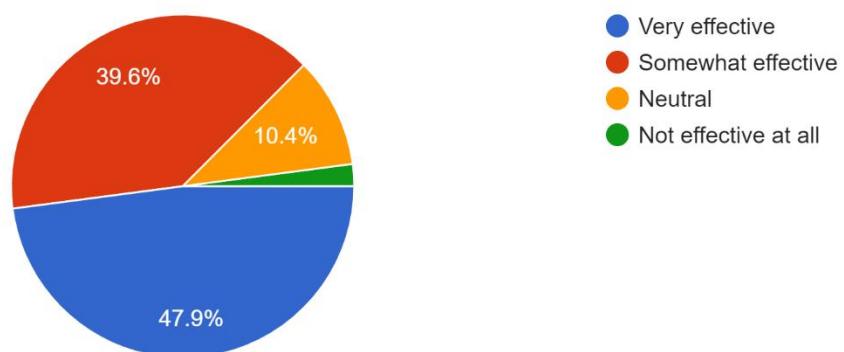
Have you used educational technology specifically designed for shape learning in your classroom?

48 responses



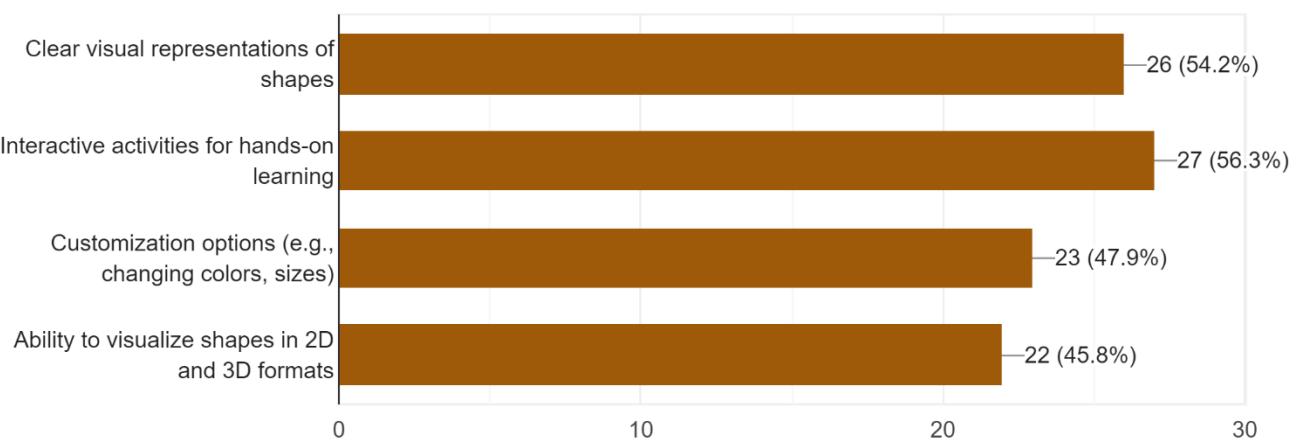
How effective do you think utilizing technology in supporting shape learning in primary education ?

48 responses



What features or functionalities do you consider important in an application designed to teach shapes to primary school children?

48 responses



Appendix 2 – Project Backlog

The screenshot shows the Jira Software interface for the HCISWING project. The left sidebar includes links for Roadmap, Backlog (which is selected and highlighted in blue), Board, Project pages, Add shortcut, and Project settings. A message at the bottom states "You're in a team-managed project" with a "Learn more" link. The main area displays the "Backlog" for "HCIS Sprint 1" from May 12 to May 26, containing 28 issues. The backlog items are:

- HCIS-3 Research
- HCIS-5 Design
- HCIS-6 Low Fidelity Prototype
- HCIS-7 High Fidelity Prototype
- HCIS-9 Login
- HCIS-10 Sign-Up
- HCIS-11 Dashboard
- HCIS-12 3D Object
- HCIS-27 Create Cube
- HCIS-28 Create Sphere
- HCIS-13 Shape Drawer

Each item has a status indicator (e.g., DONE) and a small profile picture next to it. At the top right of the backlog view, there are buttons for "Complete sprint" and three dots for more options.

Appendix 3 – Kanban Board

The screenshot shows a Jira Software Kanban board for the project 'HCISWING'. The board is titled 'HCIS Sprint 1'. The left sidebar includes links for 'Projects / HCISWING', 'Roadmap', 'Backlog', 'Board' (which is selected), 'Code', 'Project pages', 'Add shortcut', and 'Project settings'. A message at the bottom of the sidebar says 'You're in a team-managed project' with a 'Learn more' link. The main board area has three columns: 'TO DO', 'IN PROGRESS 2 ISSUES', and 'DONE 26 ISSUES'. The 'TO DO' column is empty. The 'IN PROGRESS' column contains two issues: 'Create 3D Galaxy' (issue HCIS-32) and 'Elephant' (issue HCIS-18). The 'DONE' column contains six issues: 'Research' (issue HCIS-3), 'Design' (issue HCIS-5), 'Low Fidelity Prototype' (issue HCIS-6), 'High Fidelity Prototype' (issue HCIS-7), and 'Login' (issue HCIS-9). The top right of the board shows a search bar, a notification badge with '6', and various user and settings icons. Below the board are buttons for 'Complete sprint' and 'Insights'.

Column	Issue	Key	Status
IN PROGRESS 2 ISSUES	Create 3D Galaxy	HCIS-32	In Progress
	Elephant	HCIS-18	In Progress
DONE 26 ISSUES	Research	HCIS-3	Done
	Design	HCIS-5	Done
	Low Fidelity Prototype	HCIS-6	Done
	High Fidelity Prototype	HCIS-7	Done
	Login	HCIS-9	Done