HONG KONG INSTITUTE OF VOCATIONAL EDUCATION
Laboratory 7: Introduction to 3D Graphics

Module Intended Learning Outcome (#2):
On completion of the module, students are expected to be able to:

- develop 2D and 3D graphics programs for general gaming purposes;

Lesson Intended Learning Outcome:
On completion of this lab, students are expected to be able to:

- Use vertices and indices to draw basic 3D primitives.

TASK 1 – Drawing with 3D Primitives

This time, we will draw a cube rotating in our MonoGame application. Let's start with a new project.

1. Declare the following class level variables in addition to the given ***graphics*** and ***spriteBatch***.

```
BasicEffect basicEffect; // for storing basic effect (for rendering)
float rotateAngle = 0.0f; // rotate angle of your cube
```

2. Define your vertex array as follows:

```
VertexPositionColor[] vertexList = {
    // face 1
    new VertexPositionColor(new Vector3(-32.0f, 32.0f, -32.0f), Color.Red),
    new VertexPositionColor(new Vector3(32.0f, 32.0f, -32.0f), Color.Red),
    new VertexPositionColor(new Vector3(-32.0f, -32.0f, -32.0f), Color.Red),
    new VertexPositionColor(new Vector3(32.0f, -32.0f, -32.0f), Color.Red),
    // face 2
    new VertexPositionColor(new Vector3(-32.0f, 32.0f, 32.0f), Color.Pink),
    new VertexPositionColor(new Vector3(-32.0f, -32.0f, 32.0f), Color.Pink),
    new VertexPositionColor(new Vector3(32.0f, 32.0f, 32.0f), Color.Pink),
    new VertexPositionColor(new Vector3(32.0f, -32.0f, 32.0f), Color.Pink),
    // face 3
    new VertexPositionColor(new Vector3(-32.0f, 32.0f, 32.0f), Color.Salmon),
    new VertexPositionColor(new Vector3(32.0f, 32.0f, 32.0f), Color.Salmon),
    new VertexPositionColor(new Vector3(-32.0f, 32.0f, -32.0f), Color.Salmon),
    new VertexPositionColor(new Vector3(32.0f, 32.0f, -32.0f), Color.Salmon),
    // face 4
    new VertexPositionColor(new Vector3(-32.0f, -32.0f, 32.0f), Color.Orange),
    new VertexPositionColor(new Vector3(-32.0f, -32.0f, -32.0f),
Color.Orange),
    new VertexPositionColor(new Vector3(32.0f, -32.0f, 32.0f), Color.Orange),
    new VertexPositionColor(new Vector3(32.0f, -32.0f, -32.0f), Color.Orange),
    // face 5
    new VertexPositionColor(new Vector3(32.0f, 32.0f, -32.0f), Color.Coral),
    new VertexPositionColor(new Vector3(32.0f, 32.0f, 32.0f), Color.Coral),
    new VertexPositionColor(new Vector3(32.0f, -32.0f, -32.0f), Color.Coral),
    new VertexPositionColor(new Vector3(32.0f, -32.0f, 32.0f), Color.Coral),
    // face 6
    new VertexPositionColor(new Vector3(-32.0f,  32.0f, -32.0f),
Color.Crimson),
    new VertexPositionColor(new Vector3(-32.0f, -32.0f, -32.0f),
Color.Crimson),
    new VertexPositionColor(new Vector3(-32.0f, 32.0f, 32.0f), Color.Crimson),
    new VertexPositionColor(new Vector3(-32.0f, -32.0f, 32.0f), Color.Crimson)
};
```

3. Create and initialize your **BasicEffect** objects in the ***Intialize()*** function.
   Set the camera position at (0,0,250) and make it look at **`Vector3.Zero`**. Set the Up direction

as **`Vector3.Up`**. Your projection matrix should create a frustum with Field of View of 45°, near clip at z = 1.0f, and far cilp at z = 1000.0f . Get the aspect ratio from your viewport.

4. Remember to enable the vertex color in your **BasicEffect** object.
5. In your ***Update()*** method, increase the rotation angle by 1°. Keep the angle between 0° and 2π. Use the rotation angle to create the world matrix like this:

```
basicEffect.World = Matrix.CreateFromYawPitchRoll(
        MathHelper.ToRadians(rotateAngle),        // rotate along Y-axis
        MathHelper.ToRadians(rotateAngle*2),            // rotate along X-
axis
        MathHelper.ToRadians(rotateAngle*3));           // rotate along Z-
axis
```

6. Apply the basic effect to draw the cube in your ***Draw()*** method as follows:

```
basicEffect.CurrentTechnique.Passes[0].Apply(); // Apply the BasicEffect
DrawCube();                                     // Draw the cube
```

7. Write a private function ***DrawCube()*** to draw the 6 faces by calling ***DrawUserPrimitives()*** function.

```
private void DrawCube() {
      GraphicsDevice.DrawUserPrimitives<VertexPositionColor>(
            PrimitiveType.TriangleStrip, vertexList, 0, 2);
      /*** COMPLETE THE REST OF THE FUNCTION YOURSELF ***/
}
```


8. Compile and execute the program to see what happens.
9. Set the **RasterizerState** to make the whole cube appear:
   i) Add a class-level **RasterizerState** variable ***rs***.
   ii) Initialize your **RasterizerState** variable ***rs*** in ***Initialize()*** function. Set the cull mode to **CullMode.None**. You may want to set the fill mode to **FillMode.WireFrame** so that only the lines of the cube are shown.
   iii) Set the Graphics Device's **RasterizerState** to your variable.


```
GraphicsDevice.RasterizerState = rs;
```


TASK 2 – Drawing with 3D Primitives Using Indices

Try to modify the above project to use ***DrawUserIndexedPrimitives()*** instead.
You may use the following vertex array and index array for this part of the lab.

```
VertexPositionColor[] vertexList2 =  {
      new VertexPositionColor(new Vector3(-32.0f, -32.0f, -32.0f), Color.Red),
      new VertexPositionColor(new Vector3(-32.0f, 32.0f, -32.0f), Color.Orange),
      new VertexPositionColor(new Vector3(32.0f, 32.0f, -32.0f), Color.Yellow),
      new VertexPositionColor(new Vector3(32.0f, -32.0f, -32.0f), Color.Green),
      new VertexPositionColor(new Vector3(-32.0f, -32.0f, 32.0f), Color.Blue),
      new VertexPositionColor(new Vector3(-32.0f, 32.0f, 32.0f), Color.Indigo),
      new VertexPositionColor(new Vector3(32.0f, 32.0f, 32.0f), Color.Violet),
      new VertexPositionColor(new Vector3(32.0f, -32.0f, 32.0f), Color.Black)
};
short[] indexData = {   // 12 triangles forming the cube, each with 3 vertices
      0,1,2,            0,2,3,            4,6,5,            4,7,6,
      4,5,1,            4,1,0,
      3,2,6,            3,6,7,            1,5,6,            1,6,2,
      4,0,3,            4,3,7
};
GraphicsDevice.DrawUserIndexedPrimitives<VertexPositionColor>(
      PrimitiveType.TriangleList, vertexList2,
```

```
      0,  // vertex buffer offset to add to each element of the index buffer
      8,  // number of vertices to draw
      indexData, 0, 12   // number of primitives to draw
);
```