**Project 4: Render Your Scene With Primitives**

Caleb Klinger, Kyungchan Im (Chris)

Grand Canyon University

CST - 435 : Computer Graphics Lecture & Lab

Ricardo Citro

October 8, 2023

**A Description of the Project**

The "Your Surrounding World" project is one part of a four-part series aimed at replication of environments in OpenGL. We are tasked with selecting three distinct scenes from their environment, which could range from the intimacy of a dorm room to the expanse of a garden. We choose to pick classroom table, vending machine, library couches. Alongside these photographs, students should provide a brief description of each scene, detailing their reasons for selection and the standout elements. As a precursor to creating 3D renderings using OpenGL in subsequent projects.

**A Description of the Methodology/Approach used**

Our method was to select rectangular or square objects. As in OpenGL, the complexity of recreating architectural structures and shapes largely depends on their geometric intricacy and the level of detail required. Simple architectural forms, such as basic prisms or cubes, are foundational and can easily be rendered. These forms often represent buildings in minimalist or abstract visualizations. However, as we delve into more intricate architectural details like ornate facades, vaulted ceilings, or intricate tile patterns, the number of vertices, edges, and textures increases, making the rendering process more computationally intensive. For instance, recreating a modern skyscraper with a glass facade might involve handling reflections and transparency, while modeling an ornate Gothic cathedral would require intricate stonework details, flying buttresses, and stained-glass effects. The more detailed and varied the architectural elements, the more challenging they are to recreate accurately in OpenGL.

**Pictures**

An easy initial structure that can have added detail such as number pad and drinks. There is also a do not slip sign that can be added. The picture has noticeable shadows and lighting as well that can be added.

A tv on a stand

Description automatically generated

Figure .Original Photo for the project

TV Set Up in Apartment (Second Picture Preferred)

Easy shapes such as rectangles and squares. The texture is a flat color which makes it easier to finish. There are small details to add on bottom that are not very complicated to add. Flat textures and minor reflections

**A statement**

We took these photos with our phones clearly as all pictures are at GCU.

**Project 3 Part**

Include the entire documentation from Project 2, and make necessary changes and improvements based on feedback received from the instructor.

1. List the objects in the foreground, background, and in-between.
   1. Foreground: Nintendo switch, Controllers and box, Cables
   2. Background: Window, Background of the window
   3. In-between: The white wall, TV screen, TV stand table, Floor, Floor Line
2. Identify the main objects in the scene.

1) The white wall

2) TV screen

3) TV stand table

4) Nintendo switch

5) Controllers and box

6) Floor

7) Floor Line

8) Window

9) Background of the window

1. Describe key characteristics of the scene.

The scene depicts a typical dormitory room setting. The atmosphere is simple and functional, with a focus on entertainment and relaxation. The glossy reflection on the TV, the wooden texture of the table, the carpeted floor, and the window offering a glimpse of the outside world are all elements that add depth and texture to the scene. There are no specific features to it, however, the scene has nice amount of furniture and textures that we can implement and perform on the OpenGL.

1. Explain how you would approach rendering each object in the scene.

* TV: This would be represented by a rectangular prism. The glossy reflection on the TV can be achieved using specular reflection and shaders.
* TV stand table: A rectangular prism with a wooden texture. Depending on the design, legs might be represented as smaller prisms or cylinders.
* White wall with floor mat: A large rectangle for the wall, with a smaller rectangle at the bottom for the floor mat. Textures can be applied to give them the desired appearance.
* Window with background: A large rectangle with a transparent texture for the glass, and another rectangle behind it representing the view outside.
* Nintendo switch, controllers, and box: These would be more complex and would require a combination of prisms and cylinders to represent their shapes. Textures would be applied to give them a realistic appearance.
* Cables: These can be represented using Bezier curves or line strips, with a cylindrical shape.
* Floor: A large rectangle with a carpet texture.
* Floor Line: A thin rectangle or line strip.
* Background of the window: Depending on what's visible, this could be a combination of rectangles and other shapes to represent buildings, trees, or other objects.

1. Rank all the objects in the scene in order of rendering difficulty.
2. Nintendo switch, controllers, and box (due to their intricate design and multiple components)
3. TV (because of the glossy reflection)
4. Cables (due to their flexible nature and interaction with other objects)
5. Window with background (transparency and external view can be challenging)
6. TV stand table (due to the wooden texture and potential design intricacies)
7. White wall with floor mat
8. Floor
9. Floor Line Background of the window (depending on the complexity of the view)
10. Draw a geometrical representation of the picture on paper, by hand, using pencil only.

A screenshot of a blackboard with writing

Description automatically generated

The drawing is from Kyungchan Im’s tablet. It is all hand drawing pictures. For our chosen photo, we can select 3 big field for this project. Mostly TV, Table, and the Background scene takes the most part of the picture, therefore I choose these objects as a main.

Mostly objects have rectangular shape, therefore it is easy to approach, however, textures, reflections, and size of the object will be the problem for this project.

1. A screen shot of a drawing

   Description automatically generatedA tv on a stand

   Description automatically generatedShow the original and the hand-drawn images side by side.

Hand-drawn scene created in Project 3 and reproduce each object using primitives. Our hand drawn scene does not contain the outside of the windows as that will be an applied texture in the future rather than a group of polygons and lines.

**Primitives:**

The primitives we used consisted of polygons and lines.

GL\_POLYGON: This primitive is used to draw polygons, which are flat, closed shapes bounded by three or more line segments. In your code, GL\_POLYGON is used to draw various parts of the window, the wall, the floor, the table, and the TV. Each glVertex2i(x, y) call specifies a vertex of the polygon, and the vertices are connected in the order they are called.

GL\_LINES: This primitive is used to draw a series of unconnected line segments. In your code, GL\_LINES is used to draw the window blinds. Each pair of glVertex2i(x, y) calls specifies a line segment.

Some challenges we faced were creating the complex geometric primitives specifically for the switch controllers. This is due to the more natural shape and angle of the controllers.

**3D Primitives**

Table, walls, and windows are going to be our 3D primitives. Since they require to put a shadow on, it is one of the tasks to achieve. We are considering to use texture for next project, however, those objects are implemented with shadows and figures so far.

**Mathematical characteristics:**

Some mathematical characteristics of the openGL program include coordinate system, color encoding, geometric primitives, color encoding, transformation, projection, and   
Coordinate System:

OpenGL uses a coordinate system to position objects in the space (2D or 3D). In your 2D scene:

X-Axis: Horizontal axis.

Y-Axis: Vertical axis.

You use glVertex2i(x, y) to specify the vertices of your polygons and lines in this 2D space.

Color Encoding:

Colors are encoded using RGBA (Red, Green, Blue, Alpha) and are specified using glColor4b(r, g, b, a). The mathematical characteristic here is the normalization of color values. Typically, colors are represented in the range [0, 255], but OpenGL expects them in a normalized form (i.e., in the range [0, 1]). Your code uses a peculiar scaling factor (127/255), which might be intended to achieve a specific visual effect.

Geometric Primitives

Polygons are defined by a set of vertices. The vertices are connected in the order they are defined, forming a closed shape. Mathematically, a polygon with vertices (x1,y1),(x2,y2),…,(xn,yn)(x1​,y1​),(x2​,y2​),…,(xn​,yn​) in 2D space defines a piecewise linear curve. While lines are defined by two points. A line from (x1,y1)(x1​,y1​) to (x2,y2)(x2​,y2​) can be represented by the line equation in a 2D space.

**Transformations Used:**

Transformations are translations, rotations and scaling which is a necessary concept for operations in graphics rendering. Projection transformations are also a mathematical characteristic used in our code specifically in gluOrtho2D(left, right, bottom, top) to define an orthographic projection. It creates a viewing volume that is a box, and anything inside this box is rendered onto the screen. An example from the code is this orthographic projection gluOrtho2D(0.0, 377, 0.0, 463); which means it defines a box-like viewing volume in the 2D space. Anything inside this box will be rendered onto the screen. The parameters (0.0, 377, 0.0, 463) define the left, right, bottom, and top of this box, respectively. Essentially, it maps the coordinate system defined by these parameters to the window's viewport. glViewport(0, 0, 377, 463); glViewport() sets up how the normalized coordinates are mapped to window coordinates.

The parameters (0, 0, 377, 463) define the lower-left corner of the viewport and the width and height of the viewport in pixels, respectively. This means that the output will be mapped to the entire window. An implicit transformation while explicit transformations are not utilized, the vertex coordinates specified in glVertex2i(x, y) implicitly determine the position and size of the objects. For example:

glVertex2i(112, 463 - 303);

glVertex2i(312, 463 - 287);

These vertex coordinates implicitly define the position and shape of the polygon.

Some other less notable transformations used are

Translation: Moving objects from one place to another.

Rotation: Rotating objects around an axis.

Scaling: Changing the size of objects.

These can be achieved using transformation matrices in OpenGL.

glTranslatef(x, y, z): Translates objects along the x, y, and z axes.

glRotatef(angle, x, y, z): Rotates objects around the x, y, or z axis by a specified angle.

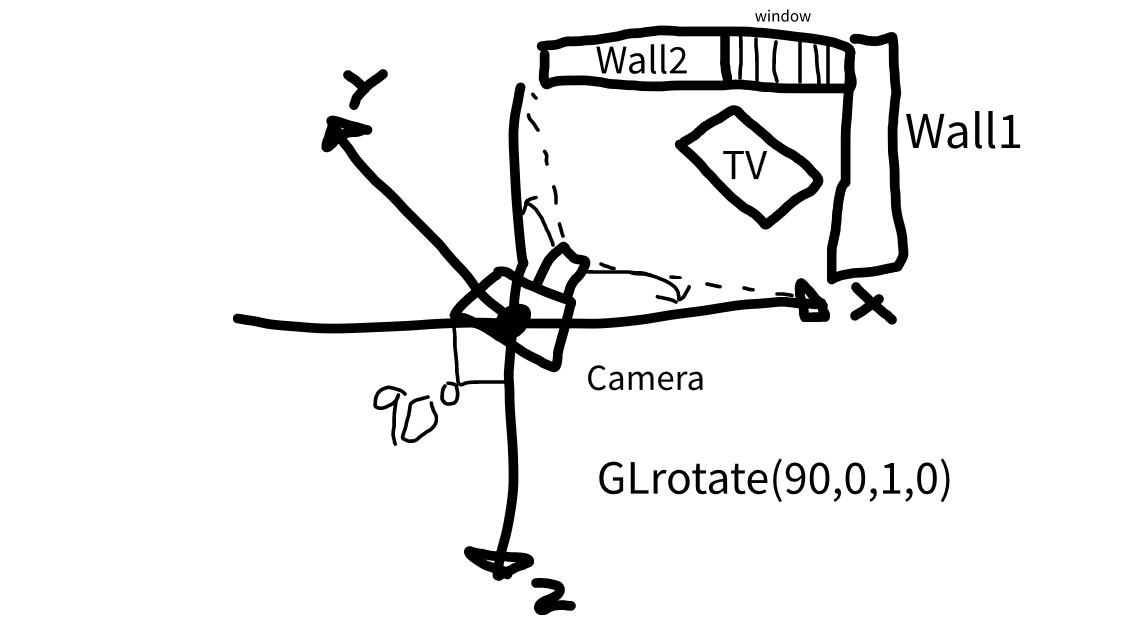
glScalef(x, y, z): Scales objects in the x, y, and z directions.

**Shaders:**

We used mapping to manually insert the shaders into the picture specifically in the table halves and the wall specifically the corner of the wall. To do this we used fragment shader which is the process where each pixel handles color calculations, lighting, and texture mapping. A potential future approach to improve the shading would use a shader-based approach. This shader provides flexibility and control over the graphics pipeline, enabling advanced effects like programmable lighting, shadows, and texture mapping. They are essential for 3D graphics and are a standard in modern OpenGL development.

**Camera:**

This is our plan for the camera position and movement of the scene. It will be a rotation of 90 deg from the corner of each wall.



**Execution and Output**

**A screenshot of a computer

Description automatically generated**

A tv on a stand

Description automatically generatedA screen shot of a drawing

Description automatically generatedA computer screen on a table

Description automatically generated**Comparison Origin vs Hand Drawing vs OpenGL result**