Benchmark - Project 5: Render Your Scene With Primitives

CST-310 Computer Graphics

Angel Velazquez, Kyler Harden

October 20, 2024

**Part 4 Documentation**

Assigned Image:



List the objects in the foreground, background, and in between.

* 1. Foreground
     1. Speaker
     2. TV
     3. Xbox
     4. PS4
     5. WII
     6. Remote
     7. Candle
     8. TV stand
     9. Batteries
     10. Ping pong balls
     11. Lantern
     12. Floor
     13. Wall
  2. Background

Main Objects of Scene

* 1. TV stand, TV, Xbox, PS4, Wii

Key Characteristics of the scene

* 1. Cozy and organized entertainment area. The focal point is a large TV. The ambient lighting is soft and warm, which suggests a lamp is somewhere in this room. This contributes to the comfortable and relaxed atmosphere.

How the Shapes would be rendered:

* 1. TV
     1. Shape: then rectangular cuboid
     2. Dimensions: flat and wide, with a very shallow depth to mimic the slip profile of modern TV’s.
  2. Gaming consoles
     1. Shape: small, thin rectangular cuboids
     2. Dimensions: slim and compact, slightly smaller than the compartments in the TV stand
     3. Placement: within one of the compartments or on top of the TV stand
  3. Game Controllers
     1. Shape: flattened, rounded ellipsoids
     2. Dimensions: small, to fit within the hands, with slight curves to suggest handles
     3. Placement: on the surface of the TV stand, near the gaming console
  4. Candle
     1. Shape: small cylinders or spheres.
     2. Placement: next to the lamp, in a small cluster or group to suggest a decorative arrangement
  5. TV stand
     1. Main body
        1. Shape: a long rectangular cuboid
        2. Dimensions: longer and shorter in height
        3. Placement: on the floor, beneath the TV
     2. Drawers/Compartments
        1. Shape: smaller rectangular cuboids
        2. Placement: aligned within the main body of the stand, equally spaced to represent storage compartments
  6. Ping pong balls
     1. Shape: small white spheres
     2. Placement: There are 3 of them inside the jar with the two on the left being on the bottom of the Jar while the one on the right is slightly raised.
  7. Lantern
     1. Shape: A small cylindrical shape for the base
     2. Body: A stretched out sphere of rounded out cylinder
     3. Placement: positioned on the far right side of the TV stand
  8. Floor
     1. Shape: A series of repeating rectangles with a small border around each
     2. Texture: Wooden
  9. Wall
     1. Shape: Square
     2. Color: White
     3. Placement: This is the background behind everything.

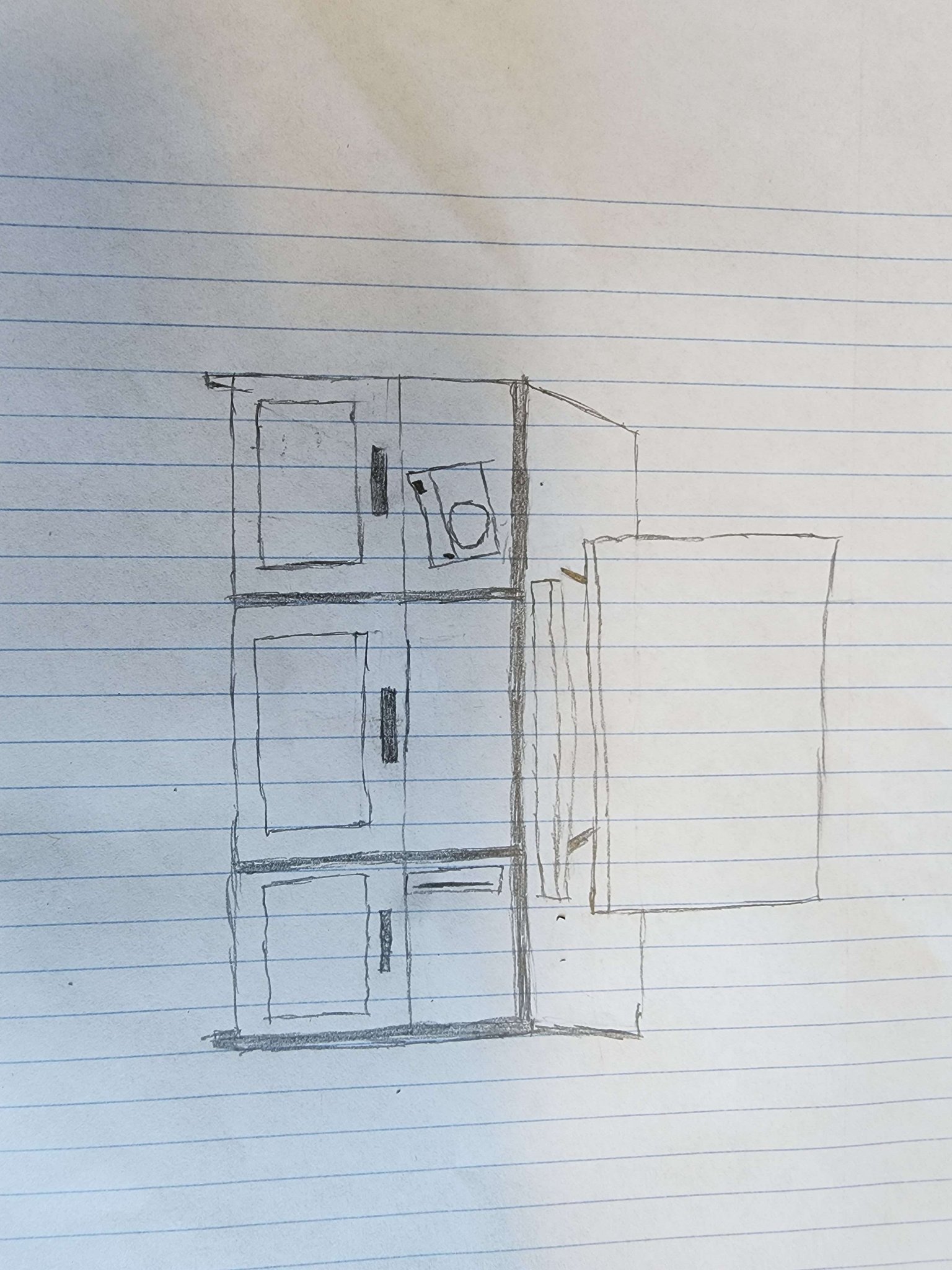
Rank of Difficulty

* Easiest

1. TV
2. Wii
3. Xbox
4. PS4
5. TV stand
6. Gaming controller
7. Candle
8. Remote
9. Lantern
10. Floor
11. Wires or cables

* Hardest

Hand Drawn Next to Original:



Original Hand Drawn

Here’s a breakdown of the **mathematical characteristics**, **primitives**, **transformations**, and **shaders** used in this project:

**1. Mathematical Characteristics**

The program primarily uses the mathematical concepts of 3D geometry, matrices, and lighting equations to render a TV stand in a 3D environment.

• **Coordinates**: Objects are placed in 3D space using transformations like translations, scaling, and rotations. The vertices in this space are described using homogeneous coordinates (x, y, z, w), and the camera is positioned in 3D space to view the scene.

• **Lighting Model**: The Phong lighting model is used to simulate light reflection. We defined ambient and diffuse light properties, which interact with the surface of objects to produce realistic lighting effects.

• **Depth Buffering**: This is used to manage which object appears in front of another based on distance from the camera, ensuring proper occlusion.

**2. Primitives Used to Render**

The main primitive used in this code is the **cube**, specifically through the function glutSolidCube(). Each component of the TV stand, including the body, cubbies, and drawers, is constructed by transforming the cube to the desired shape using scaling, translation, and color changes.

The cube is used repeatedly to create the following components:

• **TV Stand body**: A scaled cube representing the shelf.

• **Cubbies**: Three scaled cubes representing the compartments.

• **Drawer panels**: Scaled cubes representing the front panels of the drawers.

• **Drawer dividers and handles**: Scaled cubes representing the separators and handles.

• **TV Screen**: A scaled and translated cube representing the TV screen on top of the stand.

**3. Transformations Used and Explanations**

Several transformations are applied using OpenGL’s matrix stack to modify the position, size, and orientation of each component. These include:

• **Scaling (**glScalef**)**: Used to resize the cube to resemble different parts of the TV stand. For example, the stand body is scaled with glScalef(4.0f, 1.0f, 1.5f), making it wider and shorter to resemble a rectangular shelf.

• **Translation (**glTranslatef**)**: Used to move objects to their correct locations in the scene. For instance, the cubbies are moved using translations like glTranslatef(-1.3f, 0.23f, 0.51f).

• **Push/Pop Matrix (**glPushMatrix **/** glPopMatrix**)**: These functions are used to save and restore the current transformation state. This allows you to apply transformations to individual objects without affecting the transformations of others. Each cube is drawn between a push and pop to ensure its transformations don’t interfere with other objects.

• **Perspective Projection (**gluPerspective**)**: Sets up a perspective projection matrix, which simulates the way objects appear smaller as they get farther from the camera. The parameters specify a field of view of 45 degrees, an aspect ratio of 800/600, and a clipping range from 1.0 to 100.0 units.

• **Camera Setup (**gluLookAt**)**: Defines the position of the camera and where it is looking. In this case, the camera is positioned at (0.0f, 2.0f, 5.0f) and is looking toward the origin (0.0f, 0.0f, 0.0f), with an up vector of (0.0f, 1.0f, 0.0f) to ensure correct orientation.

**4. Shaders and Lighting**

While the code does not explicitly define shaders (as it uses the fixed-function pipeline of OpenGL), it does include basic **lighting**:

• **Ambient Light**: This is the light that scatters in all directions, affecting the color of objects regardless of their position relative to the light source. It’s defined by the lightAmbient[] array and applied to GL\_LIGHT0 using glLightfv(GL\_LIGHT0, GL\_AMBIENT, lightAmbient).

• **Diffuse Light**: This is the directional light that simulates how light strikes a surface directly. It’s defined by the lightDiffuse[] array and applied to GL\_LIGHT0 using glLightfv(GL\_LIGHT0, GL\_DIFFUSE, lightDiffuse).

• **Position of Light**: The light source is positioned at (0.0f, 5.0f, 5.0f) in the scene, meaning it’s above and in front of the TV stand. This is set with glLightfv(GL\_LIGHT0, GL\_POSITION, lightPosition).

• **Color Material**: The glEnable(GL\_COLOR\_MATERIAL) function ensures that the object’s material properties (how they reflect light) are automatically set by their color, so you don’t need to explicitly define materials.

**What the Lighting Achieves:**

• **Realism**: By enabling ambient and diffuse lighting, the scene is given a more realistic appearance, with objects reflecting light based on their color and position relative to the light source.

• **Depth Perception**: Proper lighting allows the viewer to perceive depth and shading, making the objects appear three-dimensional.

**Summary of What is Accomplished**

• **Rendering a TV Stand**: The code successfully renders a TV stand with compartments and drawers using scaled cubes.

• **3D Scene**: A simple 3D scene is created with realistic lighting and camera positioning to give a sense of depth.

• **Lighting and Shading**: The lighting model provides ambient and diffuse light, making the objects interact naturally with light, simulating a more real-world appearance without the need for custom shaders.

**Screenshot of Project:**

The following image is the recreation of the Hand drawing from Part3 using glut and soil for the xbox texture.



**Part 5 Documentation**

**1. Mathematical Characteristics**

• **Coordinate System**: The scene uses a standard Cartesian coordinate system in 3D space, with the positive x-axis extending to the right, the positive y-axis extending upwards, and the positive z-axis extending towards the viewer.

• **Scaling and Translation**: Various objects, such as the TV stand and gaming consoles, are scaled and translated into their positions using glScalef() and glTranslatef(). These transformations modify the objects’ sizes and locations in the 3D space.

• **Camera View**: The virtual camera is set up using the gluLookAt() function, defining the camera’s position, the point it looks at, and the up vector. This simulates the observer’s perspective in the scene.

**2. Primitives Used**

• The scene primarily uses cubes (glutSolidCube) to construct the objects, such as the TV stand, compartments, and gaming consoles. The simplicity of these cubes helps maintain performance but limits the scene’s complexity.

• The floor, cubby textures, and wall trims use textured quads (glBegin(GL\_QUADS)), where textures from external image files are applied using the SOIL library. This improves realism by providing detailed visuals for surfaces like the floor and cubbies.

**3. Transformations and Their Explanations**

• **Translation (**glTranslatef**)**: This function repositions objects in the 3D space. For example, the TV stand’s legs and drawers are translated to their correct positions relative to the main TV stand.

• **Scaling (**glScalef**)**: Scaling is applied to adjust the size of objects. The TV stand is scaled to make it larger, while the gaming consoles are scaled to smaller proportions. Scaling allows for creating variations in object sizes using the same basic primitive (a cube).

• **Push and Pop Matrix (**glPushMatrix**,** glPopMatrix**)**: These functions ensure transformations only affect specific objects. After applying translation and scaling, glPopMatrix() restores the previous transformation state, preventing changes from being applied to unrelated objects.

**4. Shaders and Textures**

• **Color Materials (**glColorMaterial**)**: This enables the materials of the objects to track the color specified by glColor3f(), adding realism to objects like the TV stand and its compartments.

• **Textures**:

• Textures are applied to the cubbies, floor, and wall trim using the SOIL library. The use of textures allows the scene to depict more complex surfaces, such as wood grains on the floor and cubby compartments, without needing additional geometry.

• **GL\_MODULATE** is used for texture environment mode, allowing the texture to blend with the object’s lighting, enhancing the realism by making the texture darker or brighter based on the lighting conditions.

**5. Camera Setup**

• **Camera Position and Orientation**: The camera is positioned at (0.0f, 2.0f, 5.0f) using gluLookAt(), giving a slightly overhead view of the TV stand and the gaming consoles. The camera looks towards the origin (0.0f, 0.0f, 0.0f) with the up vector set to (0.0f, 1.0f, 0.0f).

• **Perspective Projection**: The camera uses a perspective projection defined by gluPerspective(45.0, 800.0 / 600.0, 1, 100.0), which simulates real-world perspective where objects farther from the camera appear smaller. The 45-degree field of view is standard and prevents distortion.

**6. Camera Geometry Diagram**

Below is a description of the camera geometry, which can be visualized:

• **Eye Position (Camera)**: (0.0f, 2.0f, 5.0f)

• **LookAt Point**: (0.0f, 0.0f, 0.0f)

• **Up Vector**: (0.0f, 1.0f, 0.0f)

• **Near and Far Clipping Planes**: The near clipping plane is set to 1.0, and the far clipping plane is set to 100.0, meaning objects outside this range won’t be rendered.

Here’s a diagram of the camera setup:

+Y

|

|

Camera(0.0, 2.0, 5.0)

|

|

LookAt(0.0, 0.0, 0.0) -------------- +X

**Improvements Towards Realism (Compared to Project 4)**

• **Texturing**: Adding textures to surfaces such as the cubbies, wall trims, and floor significantly improves realism compared to using flat-colored polygons in Project 4.

• **Lighting**: The lighting setup, which includes ambient and diffuse light sources, gives the scene a more realistic feel by simulating how light interacts with objects in the real world.

• **Shadows**: The code draws shadows under the TV stand and the TV, giving a more realistic sense of depth and positioning in the 3D space.

In summary, this scene improves realism by incorporating textured objects, more refined lighting, proper object scaling, and shadows. The shaders and textures enhance the material properties, while the virtual camera setup ensures a realistic perspective view of the scene.