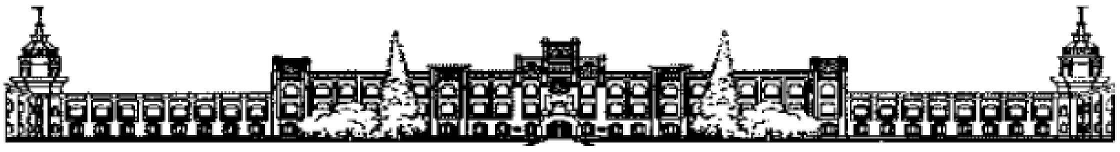


MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

NATIONAL TECHNICAL UNIVERSITY OF UKRAINE

" IHORY SIKORSKY KYIV POLYTECHNIC INSTITUTE"

Victor Porev

Computer Graphics Programming

Laboratory Work 5

3D Object Texturing using OpenGL ES

Kulubecioglu Mehmet

IM-14 FIOT

Class Number: 12

Kyiv

IHORY SIKORSKY KYIV POLYTECHNIC INSTITUTE

2025

1. Introduction

The objective of this project is to design 3D scenes using OpenGL ES 3.0 in Android-based mobile applications and enhance visual realism by applying different textures (images) to the 3D objects. This Lab5 project covers various topics in graphics programming such as user interaction, camera control, lighting, shader programming, skybox rendering, texture atlas usage, and texture mapping techniques.

2. Project Scope

This project involves the design of multiple scenes, each demonstrating different visual concepts and technical approaches. Key modules developed in this project include:

- ❖ Basic Texture Mapping (**`mySimplestMode`**)
- ❖ Chessboard Texture Mapping (**`myChessMode`**)
- ❖ Cube with Separate Textures and Texture Atlas (**`myCubeMode`**, **`myCubeAtlasMode`**)
- ❖ Skybox Rendering (**`mySkyBoxMode`**)
- ❖ Planet Sphere Scene (**`myPlanetMode`**)
- ❖ Torus Shape with Texture (**`myTorusMode`**)
- ❖ Sea View (Mixed Scene) (**`mySeaMode`**)

3. Technologies and Tools Used

Technology	Description
Android Studio	Development environment
Java	Programming language
OpenGL ES 3.0	Graphics rendering library
GLSurfaceView & Renderer	3D scene rendering base
Shader Programming	Vertex and Fragment Shaders
PNG Texture Files	Image files for surface mapping

4. Project Architecture

The project follows a modular architecture. Each scene is implemented in its own mode class, all derived from a base class called **`myWorkMode`.** The class **`myGraphicPrimitives`** handles the geometric object creation like quads, spheres, torus shapes, etc.

Key Components:

- ❖ **MainActivity:** Manages menu interactions, mode switching, and touch events.

- **myWorkMode:** Manages shaders, VAO/VBO, texture loading, and drawing routines.
- **myGraphicPrimitives:** Handles all geometric object generation.
- **myShadersLibrary:** Stores all vertex and fragment shader code.

5. Scene Analysis

5.1 mySimplestMode

- Demonstrates a basic 2D texture mapping.
- Texture Used: `img_17.png`
- Camera control via touch interaction.

5.2 myChessMode

- Applies a chessboard texture mapping.
- Texture Used: `img_15.png`
- Manual light color and position defined.

5.3 myCubeMode / myCubeAtlasMode

- **`myCubeMode`:** Each face has its own texture (`img_2.png` to `img_7.png`).
- **`myCubeAtlasMode`:** Uses a single texture atlas (`img_8.png`) with UV mapping adjustments.

5.4 mySkyBoxMode

- Skybox textures are applied to simulate a 360° environment.
- **Textures:** `img_9.png` to `img_14.png`

5.5 myPlanetMode

- Applies a planet texture to a sphere.
- **Texture Used:** `img_1.png`

5.6 myTorusMode

- Torus (donut-shaped object) with texture mapping.
- **Textures Used:** `img_16.png`, base: `img_15.png`

5.7 mySeaMode

- Mixed scene combining cube, skybox, torus, and chessboard.
- **Textures Used:** `img_9-14`, `img_15`, `img_16`

6. Shader Usage

Vertex Shaders

- **vertexShaderCode8:** Used for simple texture-only rendering.
- **vertexShaderCode9:** Used for advanced lighting including normals and texture coordinates.

Fragment Shaders

- **fragmentShaderCode11:** Outputs texture color.
- **fragmentShaderCode13:** Applies ambient + diffuse lighting with texture mapping.

7. Texture Handling

Textures are stored under the Android **`res/drawable`** directory in **`.png`** format. Using the method **`myLoadTexture():`**

- The Bitmap is loaded from resources.
- Bound to a **`GL_TEXTURE_2D`** object.
- Texture wrapping **(`GL_REPEAT`, `GL_CLAMP_TO_EDGE`)** and filtering parameters are set.

Important Notes:

- Texture dimensions are recommended to be power-of-two (e.g., 256x256, 512x512).
- NullPointerException errors may occur if texture files are missing or incorrectly named.

8. Camera and Interaction

Camera view is controlled by variables:

- **`alphaViewAngle`**, **`betaViewAngle`**: rotation
- **`zDistance`**: zoom level

User interaction is handled via:

- **`onActionDown()`** – for zooming in/out
- **`onActionMove()`** – for rotating the camera

9. Issues and Solutions

Issue	Solution
GLUtils.texImage2D NullPointerException	Ensure texture exists in the drawable folder and is correctly referenced
Texture not visible	Check wrap/filtering settings and UV mapping coordinates
Shader compilation errors	Use **`myCompileShader`** output for debugging

10. Texture Atlas Implementation

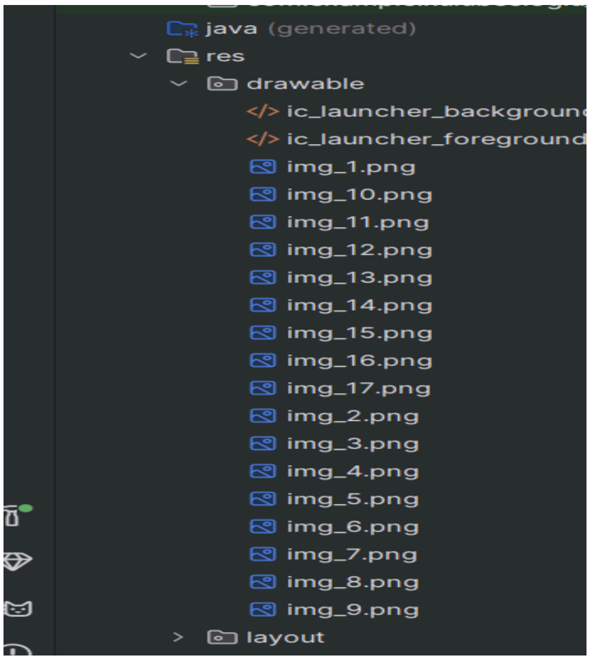
- Texture Atlas allows using a single image (**`img_8.png`**) to map multiple surfaces.
- In **`myCubeAtlasMode`**, texture coordinates are manually adjusted per face.
- This improves performance and simplifies resource management.

11. Suggestions for Further Development

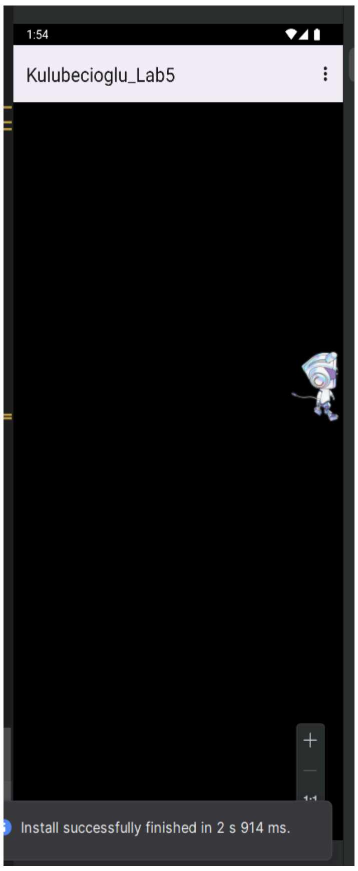
- Add specular lighting and highlights.
- Enable dynamic texture switching via UI controls.
- Implement Phong or Blinn-Phong shading models for realism.
- Animate objects or apply camera path movements.

12. Conclusion

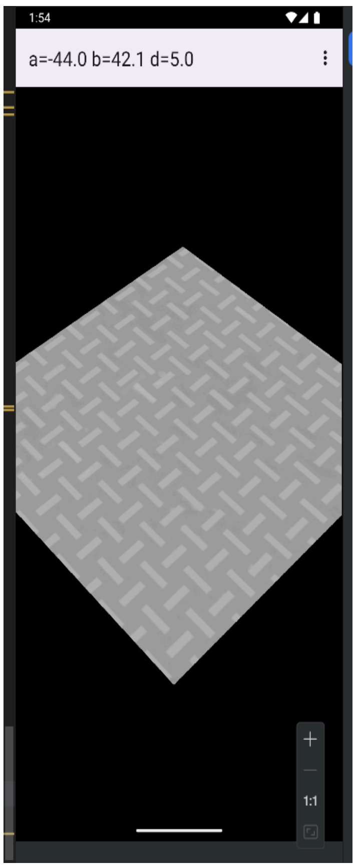
This Lab 5 project successfully demonstrates the creation of interactive, textured 3D scenes using OpenGL ES on Android. Through various scene modes, lighting models, and shader techniques, the project provides solid practical experience in real-time rendering and 3D programming.

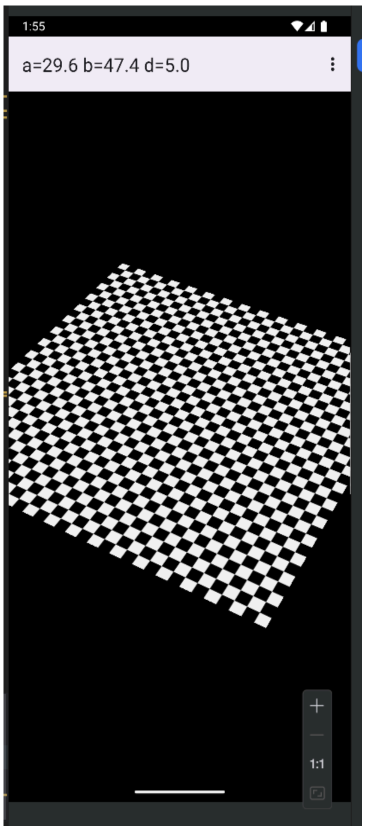


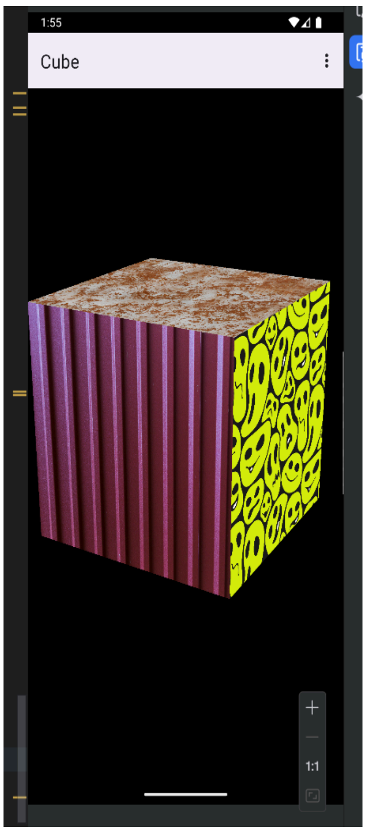
```
java (generated)
res
  drawable
    ic_launcher_backgroun
    ic_launcher_foreground
    img_1.png
    img_10.png
    img_11.png
    img_12.png
    img_13.png
    img_14.png
    img_15.png
    img_16.png
    img_17.png
    img_2.png
    img_3.png
    img_4.png
    img_5.png
    img_6.png
    img_7.png
    img_8.png
    img_9.png
  layout
```

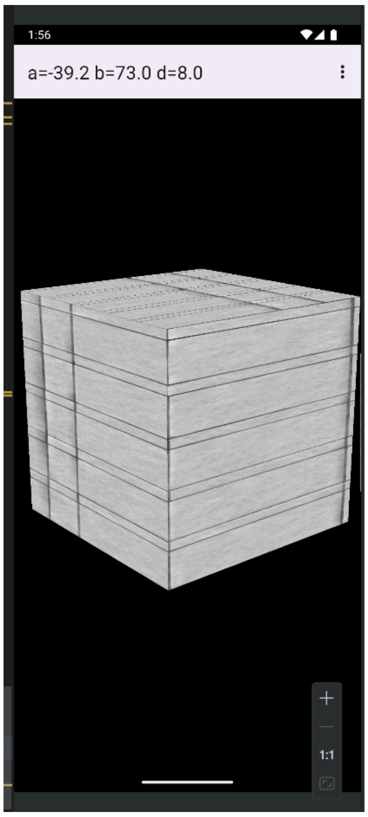


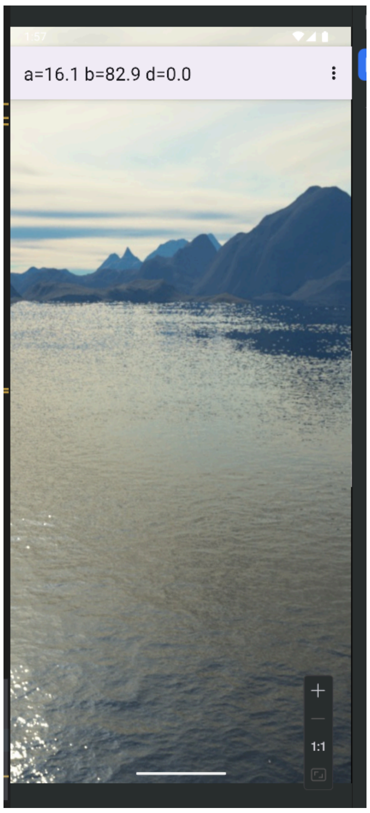
Install successfully finished in 2 s 914 ms.

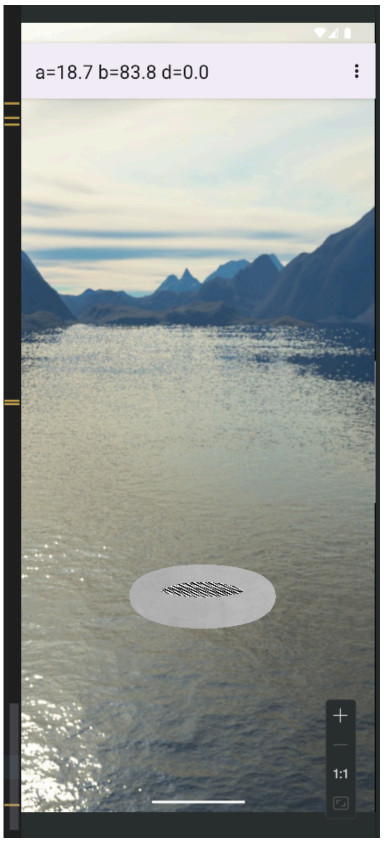


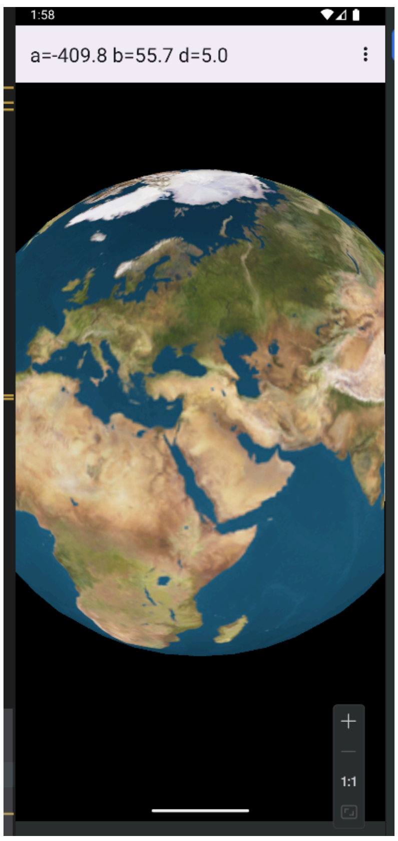


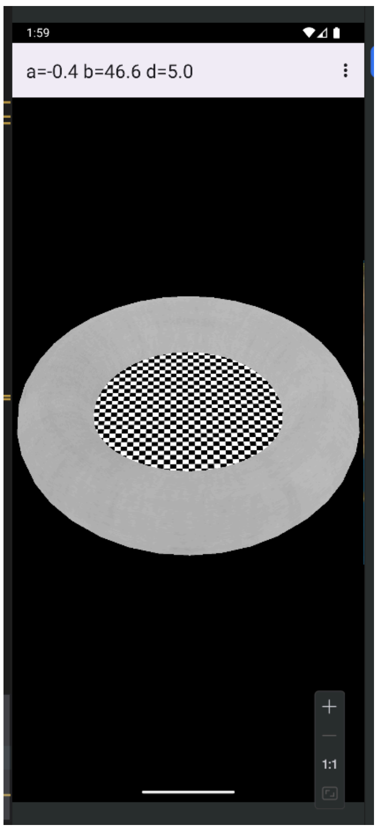












Control Questions & Answers

Q1: What is a texture?
A texture is a 2D image (bitmap) applied to the surface of a 3D model to enhance its realism. It works like wrapping an image over the object's surface.

Q2: How do we load a texture?
Textures are loaded using the `myLoadTexture()` method, which binds the image resource, sets filtering and wrapping parameters, and uploads it to GPU memory using `GLUtils.texImage2D()`.

Q3: What is a texture atlas and what benefits can it provide?
A texture atlas is a single large image containing multiple smaller textures grouped together. It reduces the number of texture bind calls, improving performance and memory usage.

Q4: How do we loop a texture image?
By setting texture coordinates beyond 1 (e.g., from 0 to 4) and using `GLES32.GL_REPEAT` in `glTexParameteri`, the image is repeated across the surface (tiling effect).

Q5: What are texture coordinates?
Texture coordinates (xT, yT) map each vertex of a 3D object to a specific location on a texture image. These coordinates range from (0,0) (bottom-left) to (1,1) (top-right).

Q6: How can we simulate an environment background image?
A Skybox cube with six textures on each face simulates a 360° environment background. The scene is rendered inside this cube to provide an immersive visual context.

Q7: What do torus and sphere mappings have in common?
Both use parametric equations and angular coordinates (latitude and longitude) to define vertices. Their texture coordinates are also calculated based on angular displacement, making them similar in mapping technique.