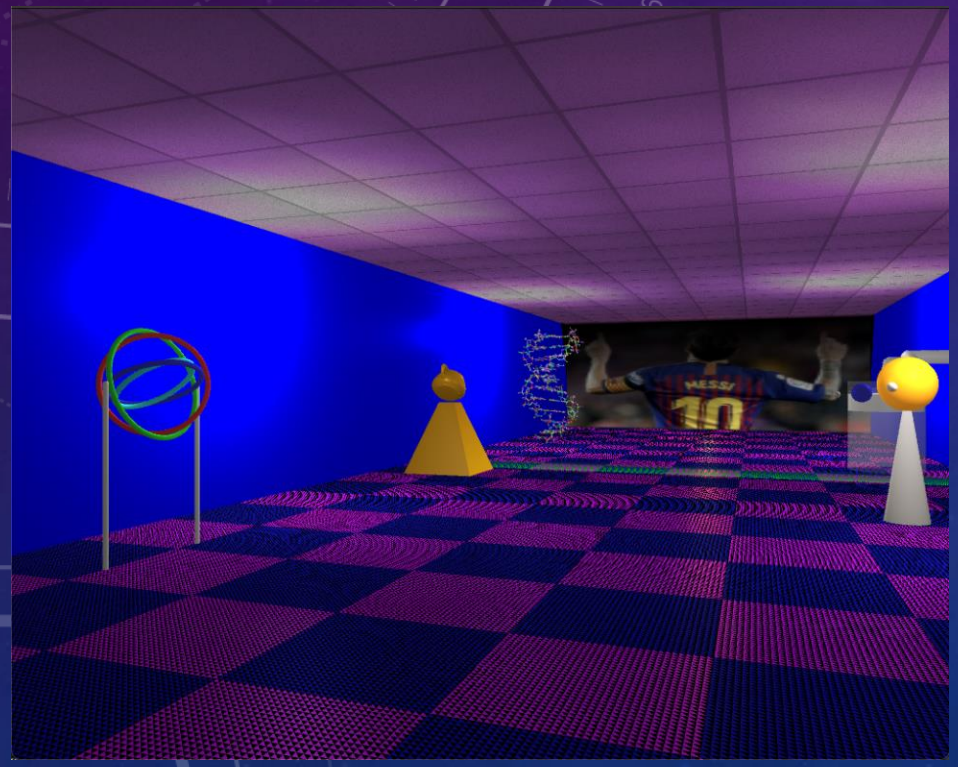
INTERACTIVE 3D VIRTUAL WALKTHROUGH OF A MUSEUM



COMP 410 PROJECT PRESENTATION

11.06.2025

BORA SOYDAN 76741

ATA CAN TAŞDAN 76216

INTRODUCTION

We designed a interactive 3D virtual museum via OpenGL. In this museum, you can freely walkthough the museum, view the sculptures and change the museum animations such as speed, camera position, lighting and so on. You can also change the design of the museum like walls, floor and ceiling.

Why Museum?

We chose to build a interactive virtual walkthough of a museum because we thought that it would be a great project to show and implement what we have learned in this course. Also, we thought that building a museum had a lot pros:

It gave us a lot of opportunity to demonstrate our creativity, to add new objects, functionalities etc.

HOW WE IMPLEMENTED THE MUSEUM

We started by setting up the basic OpenGL environment with GLUT and initializing the camera, lighting and display settings. Once the scene was ready, we began modeling and placing the sculptures and museum structure using custom draw functions. After that, we focused on implementing navigation, allowing the user to move around the space with different functionalities using keyboard and mouse controls.

TOOLS USED

OpenGL, GLUT, GLU, C

Why?

OpenGL: Core graphics library and extensive support for rendering the 3D environment and museum objects.

GLUT (OpenGL Utility Toolkit): Manage window creation, input handling and basic event control

Focus on the core navigation and interaction logic. (glutPostRedisplay, glutTimerFunc)

GLU (OpenGL Utility Library): Collection of geometry helpers and math utilities that sit on top of core OpenGL.

Tessellate spheres, cylinders, disks, build mip-maps, and help with matrix calculations. (gluSphere, gluCylinder, gluDisk)

C programming language: Direct control over performance and memory management, essential for real-time rendering tasks, efficiency and compatibility with low-level graphics APIs

COMP 410 KNOWLEDGE

In our project, we focused on using the things we learned in COMP 410 class.

- Transformation: We used rotation and traslation in some of our sculptures.

- Lighting : We used lighting and shading inside our museum. You can change the lighting and shading however you like

- Viewing and Projection: View: Depth and size variation (navWindowResize(), glFrustum()), Projection: Camera’s position and orientation (navUpdateCamera(), gluLookAt()).

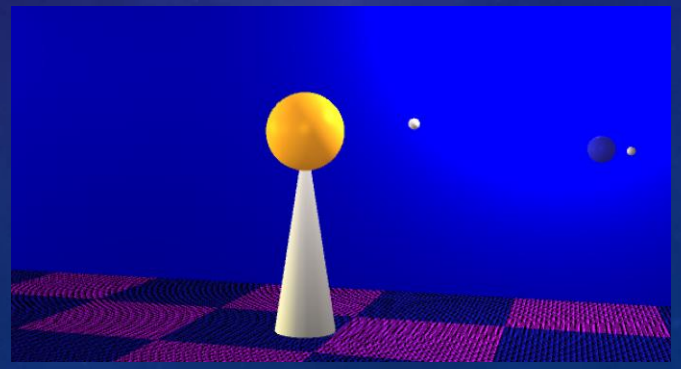
- Texture Mapping: 2D Texture Mapping, image on the wall.

- Input and Interaction: Mouse & keyboard callbacks, position

EX. : SCULPTURE 1 (SOLAR SYSTEM MODEL)

Little solar system model that contains the planets Sun, Earth, Moon and Mercury. These objects are created using OpenGL's gluSphere function and are colored with carefully chosen ambient, diffuse, and specular material properties to reflect realistic lighting. There is a stand base which is built with a circular triangle fan. The Earth and Mercury are placed using trigonometric transformations to simulate their orbits, while the Moon is attached to Earth's position, creating a nested orbital effect.

Each orbit is calculated based on angular velocity and elliptical motion formulas, meaning the distance from the center changes as the celestial bodies rotate. The Earth and Mercury move around the Sun using sine and cosine functions with dynamically updated angles and distances to mimic elliptical orbits. The Moon, in turn, orbits the Earth in a smaller loop.



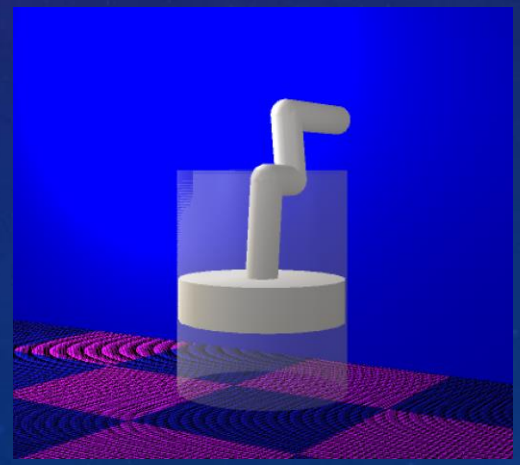
```
void drawSculpture1()
    // Stand base
    setMaterial(coneA, coneD, coneS, 27.8f);
    glBegin(GL_TRIANGLE_FAN);
        glNormal3f(0, 1.0, 0.0);
        glVertex3d(0.0, 0.0, 0.0);
        for (i = 0; i <= TILE_RES; ++i) {
            double angle = i * 2.0 * M_PI / TILE_RES;
            glNormal3f(sin(angle), 0.0, cos(angle));
            glVertex3d(100.0 * sin(angle), FLOOR_LEVEL, 100.0 * cos(angle));
        }
    glEnd();

    // Sun
    setMaterial(sunA, sunD, sunS, 100.0f);
    gluSphere(quadric, 128.0, 60, 40);
    glRotated(5.0, 0.0, 0.0, 1.0);  // Tilt for aesthetics

    // Earth + Moon
    glPushMatrix();
        glTranslated(earthDist * sin(earthTheta), 0.0, earthDist * -cos(earthTheta));
        setMaterial(earthA, earthD, earthS, 100.0f);
        gluSphere(quadric, 32.0, 35, 25);

        glTranslated(moonDist * sin(moonTheta), 0.0, moonDist * -cos(moonTheta));
        setMaterial(moonA, moonD, moonS, 1.0f);
        gluSphere(quadric, 10.0, 20, 15);
    glPopMatrix();

    // Mercury
    glPushMatrix();
        glTranslated(mercuryDist * sin(mercuryTheta), 0.0, mercuryDist * -cos(mercuryTheta));
        setMaterial(mercuryA, mercuryD, mercuryS, 1.0f);
        gluSphere(quadric, 20.0, 20, 15);
    glPopMatrix();

    glPopMatrix();
```

```
// update sculpture1 animation
void updateSculpture1()
{
    // Orbital constants
    const GLdouble EARTH_P = 350.0;
    const GLdouble EARTH_E = 0.75;

    const GLdouble MERCURY_P = 250.0;
    const GLdouble MERCURY_E = 0.58;

    GLdouble delta = (ANI_RATE / 200.0) * speedMultiplier;

    // Earth update
    GLdouble earthVelocity = (75000.0 / (earthDist * earthDist)) - (M_PI / 220.0);
    earthTheta += earthVelocity * delta;
    earthTheta  = fmod(earthTheta, 2.0 * M_PI);
    earthDist   = EARTH_P / (1 + EARTH_E * cos(earthTheta));

    // Moon update
    moonTheta += (M_PI / 6.0) * delta;
    moonTheta  = fmod(moonTheta, 2.0 * M_PI);

    // Mercury update
    GLdouble mercuryVelocity = (60000.0 / (mercuryDist * mercuryDist)) - (M_PI / 220.0);
    mercuryTheta += mercuryVelocity * delta;
    mercuryTheta  = fmod(mercuryTheta, 2.0 * M_PI);
    mercuryDist   = MERCURY_P / (1 + MERCURY_E * cos(mercuryTheta));
}
```

EX. : SCULPTURE 4 (PISTON AND CRANK)

Mechanical piston assembly. It consists of multiple cylinders and spheres constructed using gluCylinder, gluSphere, and gluDisk functions. The crankshaft, connecting rods and piston are carefully rotated and translated to visually demonstrate their mechanical roles. A transparent cylindrical shell surrounds the mechanism, giving the viewer a clear view while suggesting an enclosed engine block.

The crank angle (crankTheta) is updated based on time and a speed multiplier, ensuring consistent motion across frames. Using trigonometric relationships, the height of the piston (pistHeight) is dynamically calculated to reflect the vertical movement produced by the rotating crankshaft. When the crankshaft is near a vertical orientation, a boolean flag showBurn is triggered



```
void drawSculpture4()
    // --- Draw Main Piston Assembly ---
    glPushMatrix();

    // Crank Shaft Wall Mount
    glPushMatrix();
        glTranslated(150.0, 0.0, 0.0);
        glRotated(90.0, 0.0, 1.0, 0.0);
        gluSphere(quadric, 50.0, 20, 30);
        gluCylinder(quadric, 50.0, 50.0, 362.0, 20, 30);
    glPopMatrix();

    // Rotating Crank
    glPushMatrix();
        glTranslated(150.0, 0.0, 0.0);
        glRotated(-crankTheta * 180.0 / M_PI + 90.0, 1.0, 0.0, 0.0);
        gluCylinder(quadric, 50.0, 50.0, crankRadius, 20, 30);
        glTranslated(0.0, 0.0, crankRadius);
        gluSphere(quadric, 50.0, 20, 30);
    glPopMatrix();

    // Apply piston height and orientation
    glTranslated(0.0, -pistHeight, 0.0);
    glRotated(90.0, 1.0, 0.0, 0.0);

    // Main piston
    gluCylinder(quadric, 256.0, 256.0, 128.0, 20, 30);

    // Top of piston
    glRotated(180.0, 1.0, 0.0, 0.0);
    gluDisk(quadric, 0.0, 256.0, 20, 30);

    // Push Rod Mechanism
    glPushMatrix();
        glRotated(asin(crankRadius * sin(crankTheta) / rodLength) * 180.0 / M_PI, 1.0, 0.0, 0.0);
        gluSphere(quadric, 50.0, 20, 30);
        gluCylinder(quadric, 50.0, 50.0, rodLength, 20, 30);
```

```
void updateSculpture4()
{
    // Increment crank angle based on animation rate and speed multiplier
    crankTheta += (35.0 * (M_PI / 180.0)) * (ANI_RATE / 200.0) * speedMultiplier;
    crankTheta = fmod(crankTheta, 2.0 * M_PI);

    // Enable burn effect if crank is near vertical
    showBurn = fabs(crankTheta) < (90.0 * (M_PI / 180.0));

    // Update piston height using trigonometric crank-rod relationship
    double sinTheta = sin(crankTheta);
    double cosTheta = cos(crankTheta);
    pistHeight = crankRadius * cosTheta + sqrt(rodLength * rodLength - crankRadius * crankRadius * sinTheta * sinTheta);
}
```

INTERACTIVE FEATURES

- In our museum, we added some functionalities that can be controlled by the users. The user can control these functionalities with their keyboards. Here are these functionalities:

```
=== Museum Summary ===
• Sculpture 1: Mini Solar System model with Sun, Earth, Moon, Mercury.
• Sculpture 2: Rotating multi-colored torus disks.
• Sculpture 3: Golden teapot tilting animation.
• Sculpture 4: Mechanical piston and crank assembly.
• Sculpture 5: DNA double helix structure.
• Press 't': Image on wall and ceiling
• Press '1-8': Lighting of the museum.
• Press 'Arrow Keys': Move in the museum.
• Press 'd': Little up and down movement.
• Press 'i': Information about the museum.
• Press 'p': Play teapot pouring sound.
• Press 's': Double animation speed.
• Press 'a': Freeze/unfreeze animations.
• Press 'm': Music for museum.
• Press 'q': Quitting the museum.
• Press 'h': Show/Disappear Sculptıre 5.
• Press '+': Zoom in.
• Press '-': Zoom out.
• Press 'j': Jump in the museum.
=======================
```

FUTURE IMPLEMENTATIONS

- We have implemented a museum that has a well-designed structure and various animations. Later on, we are planning to keep improving the museum and add new features. These are the features that we have on mind.

  - Full screen mode (Started working on it but not finished)

  - Teleporting right next to the sculptures.

  - Coming from outside the museum, a door opens and we enter. (Main Focus)

  - Camera Shaking (Started working on it bu not finished)