BBM 414 - Computer Graphics Laboratory

HACETTEPE UNIVERSITY

Department of Computer Engineering

Project
Fall 2025-2026
Assoc. Prof. Dr. Ufuk Çelikcan

Due Date: 23:59 pm on Wednesday, January 7th, 2026

WebGL2 Mini Game Engine

Project Overview

The objective of this project is to implement a basic 3D rendering engine from scratch using the WebGL 2.0 API. You are not permitted to use high-level abstraction libraries such as Three.js, Babylon.js, or A-Frame. The goal is to understand the low-level graphics pipeline, shader programming (GLSL), and scene management.

You may use helper libraries for linear algebra (e.g., glMatrix) and DOM manipulation, but the rendering core and state management must be your own work.

Core Requirements (100 Points)

All students must implement the following features to achieve a full grade on the main project. The engine must run in a modern web browser without plugins.

Geometry Generation & Rendering

Your engine must be able to generate and render the following primitives procedurally (calculating vertices, indices, and normals via code):

- Cube
- Sphere (UV Sphere or Ico Sphere)
- Cylinder
- Prism (Triangular or Hexagonal)

External Model Loading

The engine must support importing 3D models exported from software like Blender, Maya, or 3ds Max.

- Format: Support for .OBJ (Wavefront) or .GLTF/.GLB files.
- The loader must correctly parse vertex positions, normals, and texture coordinates.

Texture Mapping

You must implement a texture system that allows images to be mapped onto your 3D objects.

- Support for albedo (diffuse) maps.
- Correct implementation of UV coordinates for both procedural shapes and imported models.

Lighting System

Implement a lighting model (Phong or Blinn-Phong) in your fragment shader.

- A basic GUI (you may use dat.GUI or Lil-GUI) to add objects to the scene, change their positions/rotations/scales, and adjust light properties.
- Directional Light: Simulating a sun-like source.
- Point Light: A light source that emits in all directions from a specific position (must calculate attenuation).

Perspective Camera

You must implement a perspective camera in engine view.

- WASD movement with mouse look.

Scene Graph & UI

- A basic GUI (you may use dat.GUI or Lil-GUI) to add objects to the scene, change their positions/rotations/scales, and adjust light properties.

Bonus Requirements (+50 Points)

These features are optional but highly recommended for those wishing to exceed the standard curriculum or make up for lost points elsewhere.

Dual Viewports (Viewport Scissoring) (+25 Points)

Render the scene twice within the same canvas or on two separate canvases to show different perspectives simultaneously:

- Engine View: A free-roaming view to inspect the scene.
- Camera View: The perspective of the actual "game" camera.

Advanced Camera Controller (+25 Points)

Implement a robust camera control system in game view. Choose one:

- First Person Controller: WASD movement with mouse look.
- Third Person Controller: Orbit camera that rotates around a target object.

Suggested Weekly Milestones

To ensure you finish on time, the following schedule is highly recommended:

Table 1: Milestones

Week	Milestone	Description
Week 1	Setup & Basic Pipeline	Set up the WebGL2 context, compile basic shaders, and render a simple hard-coded triangle or quad.
Week 2	Primitives & Math	Implement the Shape classes (Cube, Sphere, etc.) and integrate the glMatrix library for Model-View-Projection (MVP) matrices.
Week 3	Textures & Models	Implement the texture loader and write the parser for .OBJ files.
Week 4	Lighting	Update shaders to handle normals and implement the Phong rendering equation (Ambient + Diffuse + Specular).
Week 5	Bonus & Polish	Implement the Dual Viewport, Camera Controllers, and finalize the UI. Clean up code.

Implementation Details

1. Implement your experiment using **WebGL2**. All programming assignments must use the shader-based functionality of **WebGL2**: at least one vertex shader and one fragment shader.
2. The experiment must be original work. Turning in someone else's work, in whole or in part, as your own will be considered as a violation of academic integrity. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else. **Detection of such plagiarism in a submission will automatically void the submission and establish grounds to trigger an official disciplinary investigation.** General discussion of the assignment among peers is allowed, but do not share answers, algorithms or source codes. **Also using other resources (book, webpage etc.) as a code and javascript libraries (except starter code) are not allowed.**

What to Hand In

You should submit entire project directory including javascript files and html file in a zip file. Submission file structure is as given in below:

- b<studentNumber>.zip
  |–Project_2025 **(The whole project)**

Archieve this folder as **b<studentNumber>.zip** and send to the submit system.

Grading

Core Project (100 Points Total)

- 20 Points: Geometry Generation & Model Loading (Correct vertices/indices/normals).
- 20 Points: Lighting & Shaders (Correct GLSL implementation of Phong model).
- 20 Points: Texture Mapping (Correct UV handling).
- 20 Points: Perspective Camera (Shows objects and controllable).
- 20 Points: Code Quality & UI (Modular code, working GUI).

Bonus (50 Points Total)

- +25 Points: Dual Viewport Implementation.
- +25 Points: Advanced Camera Controller.

Academic Integrity

All work on assignments must be done individually unless otherwise stated. You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in pseudocode) will not be tolerated. In short, turning in someone else's work, in whole or in part, as your own will be considered as a violation of academic integrity. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else.