HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CƠ BẢN I

BỘ MÔN TIN HỌC CƠ SỞ

Báo cáo Thực tập Cơ sở

Giảng viên hướng dẫn : **TS Kim Ngọc Bách**
Họ và tên sinh viên : **Tống Quang Nam**
Mã sinh viên : **B21DCCN556**
Lớp : **D21CNPM02**
Nhóm : **13**

Hà Nội – 2025

Contents

i. Giới thiệu dự án

Tiêu chí	Nội dung
Tên dự án	Voxel Engine 3D
Lĩnh vực ứng dụng	Nền tảng cho các ứng dụng xử lý đồ hoạ, game đồ họa hiệu năng cao
Mục đích dự án	Xây dựng một core engine xử lý các tác vụ phần cứng để tối ưu cho việc xử lý dữ liệu đồ hoạ. Cho xem mở rộng và xây dựng những phần mềm đồ họa và game 3D hiệu năng cao.
Người thực hiện	Tống Quang Nam
Cố vấn/ hướng dẫn	TS. Hoàng Kim Bách
Thời gian thực hiện	10/03/2025 – 03/06/2025
Công cụ hỗ trợ	Github, Visual Studio Community và các thư viện Open Source như glm, assimp, glad... e.t.c

ii. Mục tiêu của dự án

1. Mục tiêu tổng quát:

Phát triển một core engine đồ hoạt 3D dạng voxel sử dụng OpenGL, có khả năng quản lý hiệu quả các tài nguyên đồ họa(texture, mô hình 3D, shader, dữ liệu vertex..) và hỗ trợ mở rộng để xây dựng các trò chơi hoặc úng dụng đồ họa hiệu năng cao. Engine hướng tới khả năng tái sử dụng, khả năng mở rọng và hiệu suất tối ưu cho lập trình thời gian thực.

2. Mục tiêu cụ thể:

STT	Mục tiêu cụ thể	Mô tả chi tiết
1	Hiển thị thế giới voxel 3D	Xây dựng hệ thống hiển thị khối voxel
2	Quản lý tài nguyên GPU	Thiết klees hệ thống quản lý buffer, texture, shader... để tối ưu truy xuất dữ liệu trên GPU.
3	Hỗ trợ tải dữ liệu mô hình 3D	Xây dựng trình tải các file định dạng phổ biến như .obj, xử lý dữ liệu vertex và nạp và GPU.
4	Tổ chức tài nguyên thành module có thể mở rộng	Áp dụng OOP và các mẫu thiết kế như Singleton, Resource Manager,.. để kiến trúc có thể mở rộng
5	Tối ưu hiệu năng hiển thị	Sử dụng các kỹ thuật như frusum culling, batching, instancing để nâng cao tốc độc khung hình.
6	Xây dựng nền tảng cho game engine	Chuẩn bị nền tảng để engine có thể dùng làm core engine cho các game đồ hoạt thực tế trong tương lai.

3. Các tính năng kỳ vọng.

- Kết xuất voxel 3D theo camera người dùng.
- Tải và quản lý texture dạng atlas hoặc rời.
- Load file .obj, .fbx, .glTF... thành dữ liệu render được.
- Trình quản lý tài nguyên (Resource Manager) tự động tránh load trùng lặp.
- Phân tách module rõ ràng: Graphics, AssetLoader, Scene, Core, IO....
- Khả năng mở rộng để tích hợp hệ thống vật lý, scripting...

4. Mục tiêu về chất lượng kỹ thuật:

Tiêu chí kỹ thuật	Mục tiêu đạt được
Hiệu năng	Khung hình > 60 FPS khi load các model nặng

Kiến trúc hệ thống	Thiết kế module rõ ràng, tuân thủ SOLID và nguyên lý OOP
Tài liệu hóa	Code có chú thích đầy đủ, tài liệu kỹ thuật mô tả class rõ ràng.
Tái sử dụng	Các module như ResourceManager, Shader, InputHandleManager có thể tái sử dụng ở các project khác.

iii. Phạm vi dự án

1. Phạm vi chức năng:

Dự án tập trung vào việc phát triển một core engine đồ hoạt 3D dựa trên kiến trúc voxel, với các chức năng chính sau:

STT	Chức năng chính	Mô tả chi tiết
1	Kết xuất thế giới Voxel 3D	Hiển thị môi trường voxel theo góc nhìn người dùng, có hỗ trợ di chuyển camera
2	Quản lý tài nguyên đồ họa	Bao gồm texture, shader, buffer, file mô hình .obj và các tài liệu liên quan.
3	Hệ thống tải dữ liệu vào GPU	Load dữ liệu vertex, texture,... vào GPU sử dụng VBO, VAO, Texture Object...
4	Trình quản lý scene cơ bản	Cho phép tổ chức và cập nhật các đối tượng voxel hoặc mô hình trong cảnh
5	Hệ thống module có thể mở rộng	Thiết kế theo hướng module độc lập(Graphics, Assets, IO,...) để phục vụ mở rộng sau này.

2. Pham vi kỹ thuật

Hạng mục	Chi tiết
Ngôn ngữ lập trình	C++(chuẩn C++14)
Graphics API	OpenGL (3.x), GLSL cho shader
Tải mô hình 3D	Hỗ trợ định dạng .obj, .fbx, glTF...vv
Quản lý texture	Texture 2D, có thể tích hợp atlas hoặc texture riêng biệt
Tương thích hệ điều hành	Windows(Linux có thể tương thích nếu sử dụng phần cứng tương tự).
Xử lýd đầu vào	Hỗ trợ xử lý bàn phím và chuột để di chuyển camera và tương tác cảnh

3. Phạm vi không bao gồm.

Dự án này tập trung vào phần core của một engine đồ hoạt và bao gồm các nội dung sau:

- Không phát triển gameplay cụ thể(không có hệ thống nhân vật, nhiệm vụ,...)
- Không tích hợp hệ thống vật lý(physics engine) hoặc collision detection.
- Không tích hợp scripting(Lua, Python...) trong giai đoạn này.
- Không hỗ trợ mạng hoặc multiplayer.
- Không phát triển công cụ editor (GUI) để xây dựng cảnh trong giai đoạn hiện tại.

4. Định hướng mở rộng (ngoài phạm vi hiện tại)

Tuy không nằmt rong phạm vi của bản dụng đầu tiên, nhưng core engine được thiết kế để mở rộng trong tương lai với:

- Tích hợp hệ thống vật lý và va chạm
- Tích hợp scripting cho gameplay
- Phát triển hệ thống UI, ánh sáng động, shadow mapping,...
- Tạo trình biên tập cảnh (scene editor).

- Port sáng nền tảng khác (Linux, macOS)

iv. Công nghệ sử dụng

1. Ngôn ngữ lập trình

Ngôn ngữ	Vai trò chính
C++	Ngôn ngữ chính để xây dựng toàn bộ core engine, xử lý logic, quản lý tài nguyên, luồng dữ liệu.
GLSL	Ngôn ngữ dùng để viết vertex shader và fragment shader trong pipeline của OpenGL, giúp xử lý đồ họa ở GPU.

2. Graphics API

Công nghệ	Vai trò
OpenGL	Giao diện lập trình đồ họa 3D chính của dự án, Sử dùng để kết xuất voxel, tải texture, xử lý buffer.

3. Thư viện và Framework bên thứ ba.

Thư viện	Chức năng
GLFW	Xử lý tạo cửa sổ, khởi tạo context OpenGL, quản lý input từ bàn phím và chuột.
GLAD	Trình tải hàm OpenGL, đảm bảo tính tương thích với phiên bản đang sử dụng
GLM – OpenGL Mathematics	Cung cấp các phép toán vector, matrixv à các phép biến đổi 3D(tránlation, rotation, scaling, projection, ..)
Assimp- OpenGL asset import library	Hỗ trợ load các mô hình 3D từ định dạng .obj và các định dang khác(nếu mở rộng sau này).

4. Công cụ phát triển và biên dịch.

Công cụ	Mục đích sử dụng
Visual Studio	IDE được sử dụng để phát triển, tổ chức project, debug, và quản lý mã nguồn.
Cmake	Dùng để quản lý build system đa nền tảng, hỗ trợ compile với các thư viện ngoài.

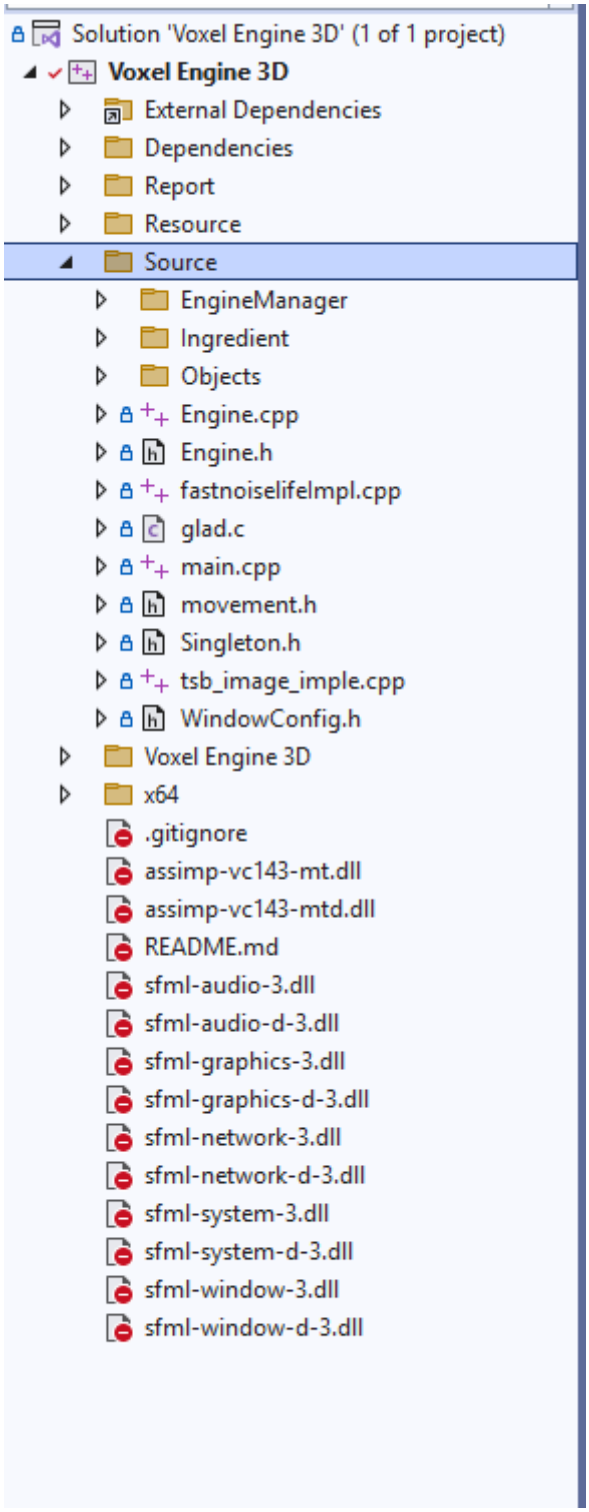
Git/Github	Quản lý mã nguồn, lưu trữ phiên bản và làm việc nhóm nếu mở rộng sau này.
OpenGL Debug Tools	Dùng để kiểm tra pipeline đồ hoạt, profile hiệu năng

5. Hệ điều hành và nền tảng mục tiêu.

Yếu tố	Mô tả
Hệ điều hành phát triển	Window 11
Nền tảng mục tiêu	Desktop PC (Windows), cấu hình có hỗ trợ OpenGL 3x, 4x)

v. Triển khai và thực hiện

1) Cấu trúc phân lớp của Source code và module của engine

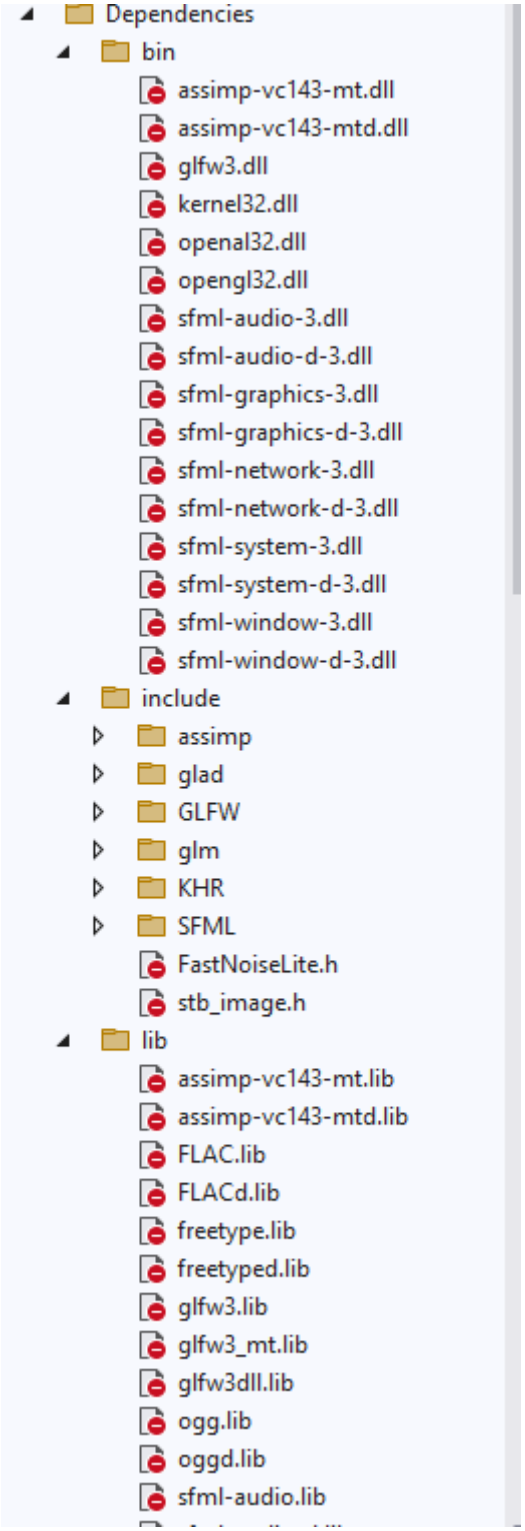


2) Cài đặt thư viện và quản lý Dependencies.

6. Danh sách Dependencies

Thư viện	Loại	Mô tả ngắn
GLFW	External	Tạo cửa sổ, quản lý context OpenGL, xử lý input bàn phím, chuột.
GLAD	External	Tải động các hàm OpenGL theo phiên bản cần sử dụng
GLM	Header-only	Thư viện toán học vector/matrix dành cho OpenGL (API giống GLSL)
Assimp	External	Load mô hình 3D từ định dạng .obj, .fbx, .dae

7. Cấu hình thư viện và dependencies



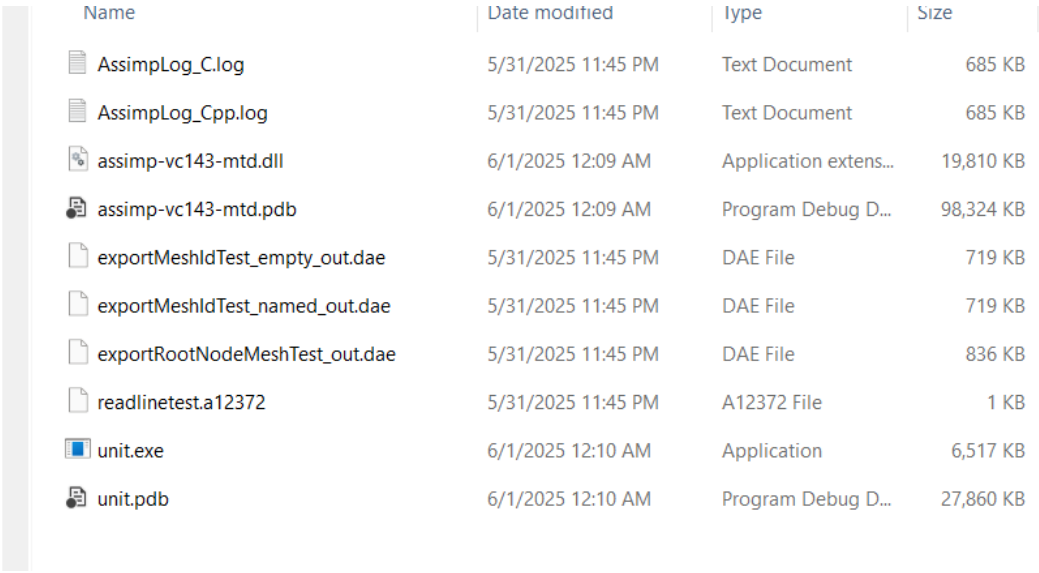
Mô tả:

- thư mục bin, chứa các file thực thi định dạng .dll, dùng cho việc chương trình .exe gọi đến khi có yêu cầu sử dụng thư viện.

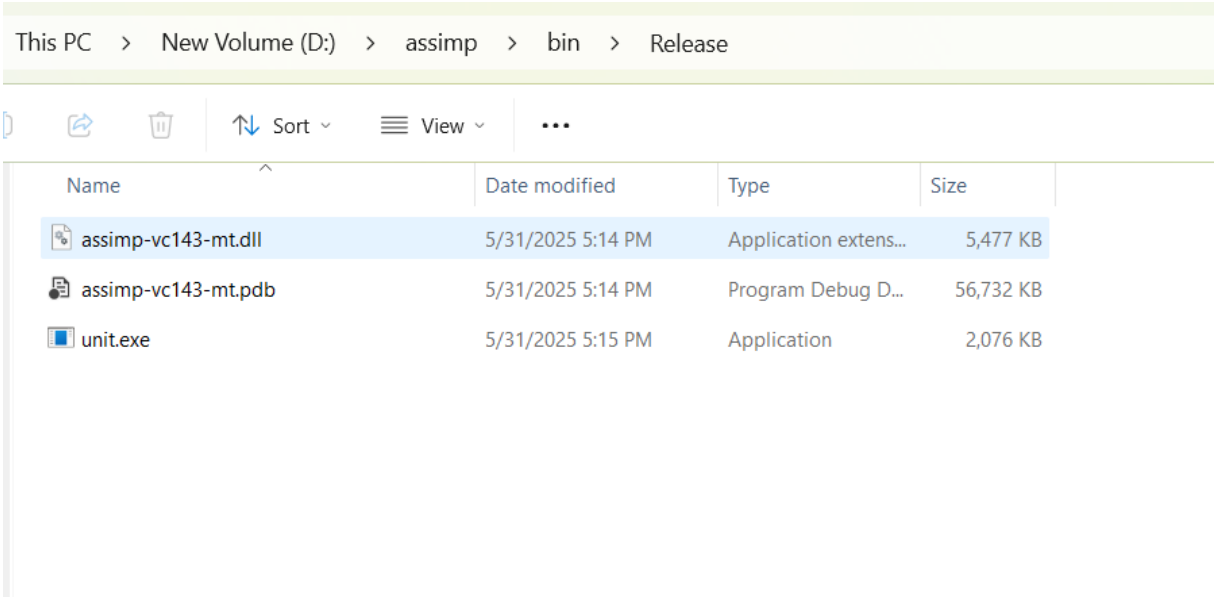
- Thư mục include, chứa các file header định nghĩa thư viện và class, struct, sử dụng để include thư viện.
- Thư mục lib: chứa các file .lib là các file thư viện đã được biên dịch.

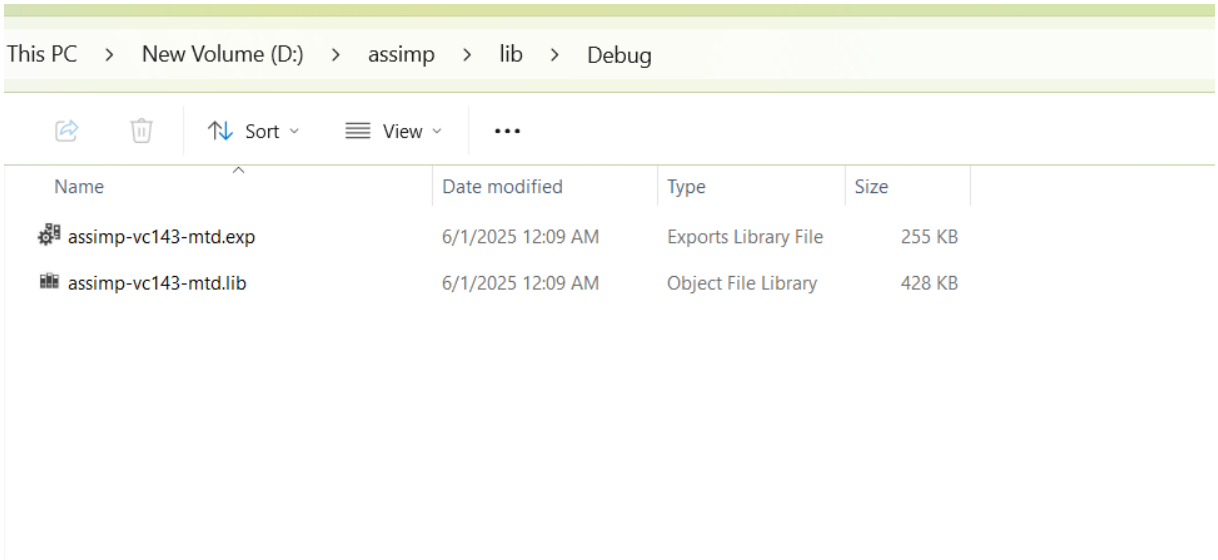
Các thư viện như glm, glad, glfw được biên dịch sẵn và đầy đủ file phụ thuộc phục vụ cho việc cấu hình. Thư viện assimp yêu cầu người dùng phải biên dịch mã nguồn để có thể sử dụng. Repository : [assimp/assimp: The official Open-Asset-Importer-Library Repository. Loads 40+ 3D-file-formats into one unified and clean data structure.]

Các triển khai: sử dụng cmake với câu lệnh " cmake –build" để tạo các file cấu hình cho Visual Studio. Mở dự án với file .sln sau đấy build mã nguồn. Kết quả được là các file phụ thuộc như hình bên dưới.



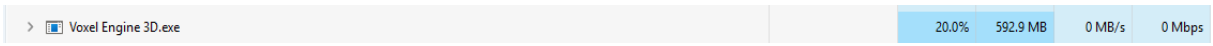
Name	Date modified	Type	Size
AssimpLog_C.log	5/31/2025 11:45 PM	Text Document	685 KB
AssimpLog_Cpp.log	5/31/2025 11:45 PM	Text Document	685 KB
assimp-vc143-mtd.dll	6/1/2025 12:09 AM	Application extens...	19,810 KB
assimp-vc143-mtd.pdb	6/1/2025 12:09 AM	Program Debug D...	98,324 KB
exportMeshIdTest_empty_out.dae	5/31/2025 11:45 PM	DAE File	719 KB
exportMeshIdTest_named_out.dae	5/31/2025 11:45 PM	DAE File	719 KB
exportRootNodeMeshTest_out.dae	5/31/2025 11:45 PM	DAE File	836 KB
readlinetest.a12372	5/31/2025 11:45 PM	A12372 File	1 KB
unit.exe	6/1/2025 12:10 AM	Application	6,517 KB
unit.pdb	6/1/2025 12:10 AM	Program Debug D...	27,860 KB





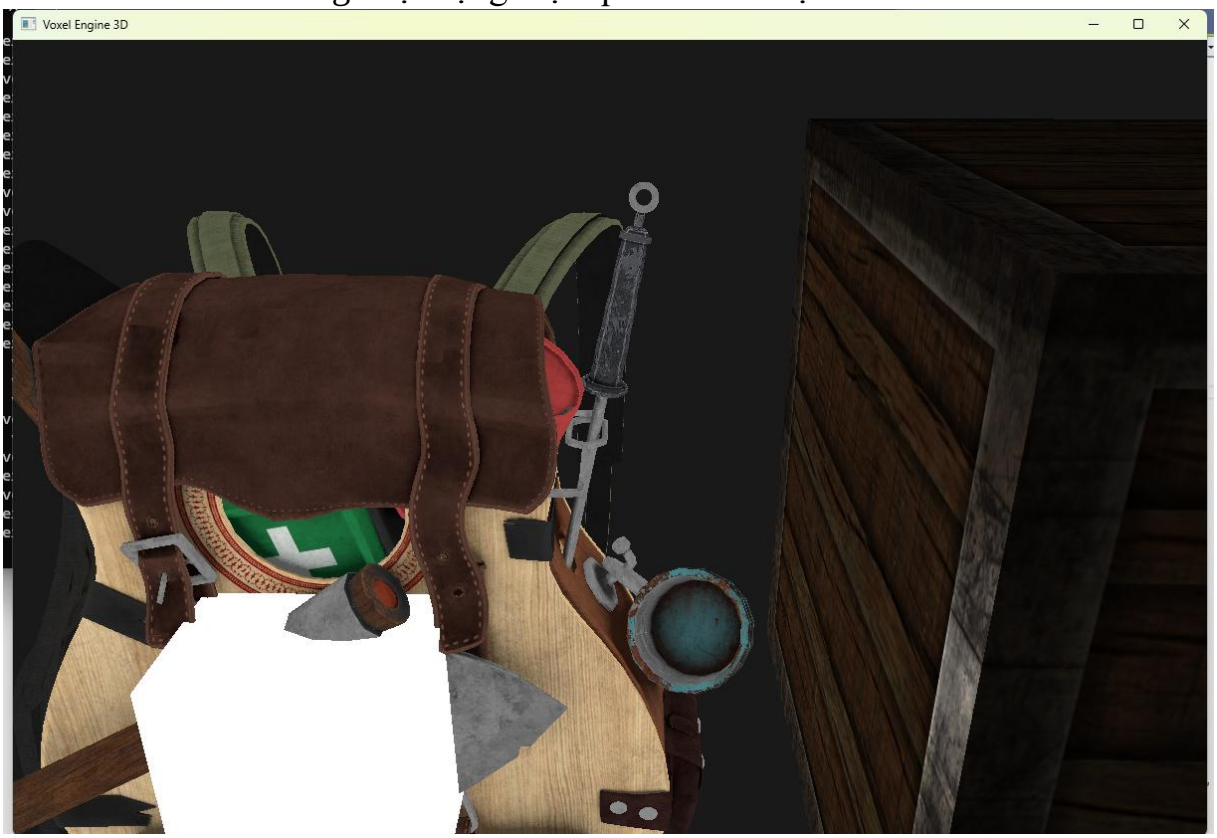
vi. Kiểm thử

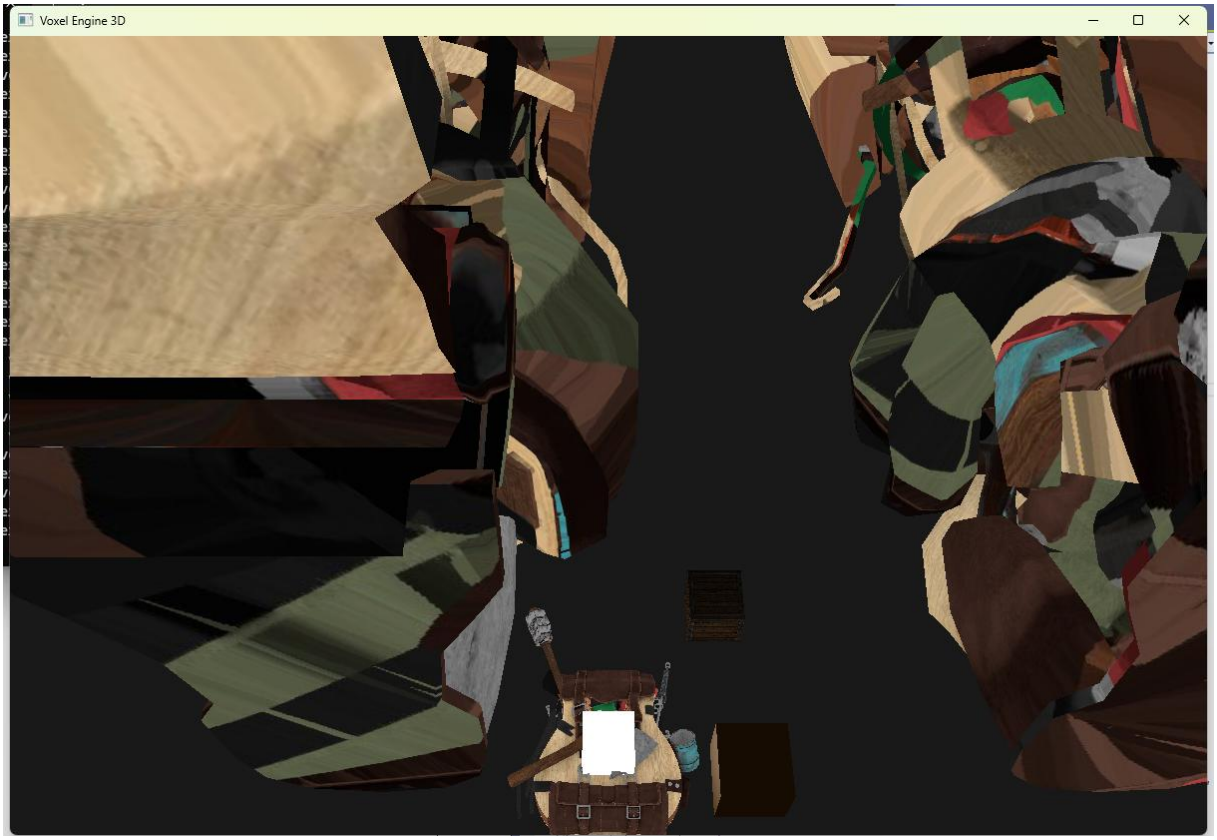
Ví dụ về kết quả kiểm thử, core engine thực hiện việc load model và khởi tạo màn hình cho người dùng với hiệu suất khung hình ổn định và bộ nhớ sử dụng lý tưởng:



vii. Kết quả đạt được

Core engine biểu diễn khả năng load và hiển thị các model mượt mà, các kiến trúc và model ánh sáng hoạt động hiệu quả cho các vật thể





viii. Tài liệu tham khảo

1) Thư viện và mã nguồn sử dủ dụng

Tên thư viện	Mục đích sử dụng	Link tham khảo
GLFW	Tạo cửa sổ, xử lý input, context OpenGl	https://www.glfw.org/
GLAD	Load function pointer cho OpenGL	https://glad.dav1d.de/
GLM	Thư viện toán học (vector, matrix...) chuẩn hóa cho OpenGL	https://github.com/g-truc/glm
Assimp	Load mô hình 3D từ các định dạng như .obj, .fbx,...	https://github.com/assimp/assimp
OpenGL	API đồ họa chính dùng để render	https://www.khronos.org/opengl/
stb_image	Load ảnh (PNG, JPG) làm texture	https://github.com/nothings/stb

2) Tài liệu hướng dẫn và học tập

Tên nguồn	Nội dung	Link
LeanOpenGL	Hướng dẫn học OpenGL từ cơ bản đến nâng cao, có phần về lighting, model loading, instancing,...	https://learnopengl.com/
OpenGL Wiki	Tài liệu chuẩn và API tham khảo chính thức	https://www.khronos.org/opengl/wiki/

The Cherno (Youtube)	Series hướng dẫn viết game engine bằng C++ và OpenGL	https://www.youtube.com/user/TheCherno
Cpppreference	Tài liệu C++ chuẩn hiện đại (STL, pointer, template...)	https://en.cppreference.com/
Assimp Documentation	Hướng dẫn sử dụng API Assimp	https://documentation.help/assimp/