TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI
KHOA CÔNG NGHỆ THÔNG TIN
======*======**

BÁO CÁO BTL THUỘC HỌC PHẦN:
ĐỒ HỌA MÁY TÍNH

XÂY DỰNG CHƯƠNG TRÌNH MÔ PHỎNG PHÒNG LÀM VIỆC SỬ DỤNG OPENGL KHẢ LẬP TRÌNH VÀ VISUAL C++

GVHD : Ths. Vũ Đức Huy

Nhóm - Lớp : 13 – 20241IT6010003

Thành viên : Bùi Văn Nghiêm - 2022603281 - K17

Trần Hồng Quân - 2022605308 - K17

Vũ Xuân Thương - 2022605678 - K17

Nguyễn Văn Tiến - 2022603015 - K17

Hà Nội - Năm 2024

LỜI CẢM ƠN

Trước tiên, nhóm chúng em xin gửi lời cảm ơn chân thành và sâu sắc nhất Ths. Vũ Đức Huy, người đã tận tình hướng dẫn, cung cấp những kiến thức quý báu và luôn sẵn sàng giải đáp mọi thắc mắc trong suốt quá trình thực hiện đề tài bài tập lớn môn Đồ họa máy tính.

Bên cạnh đó, nhóm xin gửi lời cảm ơn đến mọi người đã ủng hộ, chia sẻ ý tưởng và động viên trong suốt quá trình hoàn thành bài tập lớn.

Mặc dù nhóm đã cố gắng hết sức trong việc nghiên cứu và thực hiện, nhưng do hạn chế về thời gian và kinh nghiệm, bài báo cáo không tránh khỏi những thiếu sót. Nhóm rất mong nhận được sự góp ý từ thầy cô và các bạn để có thể hoàn thiện hơn trong tương lai. Mọi góp ý, nhận xét từ thầy và mọi người sẽ là động lực để chúng em tiếp tục hoàn thiện và nâng cao sản phẩm của mình

Chúng em xin chân thành cảm ơn!

MỤC LỤC

DANH MỤC HÌNH ẢNH

MỞ ĐẦU

1. Tên đề tài

Xây dựng chương trình mô phỏng phòng làm việc sử dụng OpenGL khả lập trình và Visual C++

2. Nội dung nghiên cứu

1.2.1. Kiến thức

- Nghiên cứu và ứng dụng các kiến thức cơ bản về đồ họa máy tính, bao gồm: hệ tọa độ, phép chiếu, ánh sáng, vật liệu, và các thuật toán dựng hình.
- Tìm hiểu và sử dụng thư viện OpenGL để lập trình các đối tượng đồ họa và xây dựng không gian 3D.
- Hiểu cách thức kết hợp giữa Visual C++ và OpenGL trong việc phát triển ứng dụng đồ họa.

1.2.2. Kỹ năng

- Kỹ năng lập trình sử dụng Visual C++ để xây dựng giao diện và tích hợp các chức năng đồ họa.
- Kỹ năng thiết kế và dựng mô hình 3D của một không gian phòng làm việc, bao gồm bố trí các đối tượng như bàn, ghế, máy tính và các thiết bị văn phòng khác.
- Kỹ năng kiểm tra và tối ưu hóa hiệu năng chương trình, đảm bảo tính tương tác và độ mượt mà khi sử dụng.
- Kỹ năng xử lý các yếu tố trực quan như ánh sáng, bóng đổ, và vật liệu để tạo hiệu ứng chân thực trong không gian mô phỏng.

CHƯƠNG I. XÁC ĐỊNH VÀ PHÂN TÍCH BÀI TOÁN

1.1. Bài toán

Nhóm em đã đưa ra bài toán về xây dựng mô hình 3D của một phòng làm việc cơ bản với các vật dụng như bàn ghế làm việc, bàn ghế sofa, quạt trần, tủ sách , máy tính, đèn, đồng hồ,... nhằm tạo nên một môi trường làm việc giống thực tế. Ngoài ra cũng có các hoạt động đóng mở cửa, bật tắt quạt trần, điều hòa, di chuyển ghế...

Mục tiêu:

+ Tạo 1 không gian phòng ảo có thể quan sát ở mọi góc độ, vị trí khác nhau, sự tương tác giống như thực.
+ Giúp mang đến trải nghiệm chân thực về môi trường, không gian làm việc.

Mục đích nghiên cứu:

+ Trau dồi kiến thức đồ họa, mô phỏng không gian 3d cũng như nâng cao kỹ năng lập trình của bản thân.
+ Đúc kết ra những kinh nghiệm và kiến thức mới sau dự án, từ đó cải thiện và hoàn thiện những dự án sau này.

Yêu cầu của bài toán xây dựng mô hình phòng làm việc:

+ Xây dựng và thiết kế các vật dụng cơ bản trong phòng làm việc nhằm tạo ra một mô hình phòng học có bố cục hợp lý, ưa nhìn.
+ Hình thành các đối tượng 3D với tỉ lệ chính xác cao dựa trên hình học cơ bản (khối lập phương) , màu sắc, vị trí sao cho giống với thực tế.
+ Thiết kế các đối tượng động như quạt trần, đèn,... mô phỏng các tính năng của các đồ vật thường thấy ở trong thực tế.
+ Thiết kế, tạo tương tác chuyển động đối với các đối tượng động sát với thực tế.

1.2. Mô tả các đối tượng cần thiết kế

Mô hình phòng làm việc gồm có các đối tượng chính như bộ bàn ghế làm việc, máy tính, điều hòa, bộ sofa, quạt trần, đèn...

Mô hình của từng đối tượng cụ thể như sau:

- Phòng làm việc: gồm có 4 bức tường, 1 trần nhà và 1 sàn nhà. Bên cạnh đó có 1 của chính và cửa sổ có thể hoạt động đóng mở tùy vào nhu cầu của người dùng.
- Bóng đèn được gắn trên tường có thể bặt hoặc tắt tạo ánh sáng cho phòng làm việc.
- Camera được gắn trên tường có thể quay qua lại một góc nào đó để quan sát, theo dõi được cả căn phòng.
- Bàn ghế sofa: Bộ bàn ghế sofa được đặt để có thể trò chuyện, tiếp khách. Ghế sofa có thể di chuyển trên mặt sàn, tạo tiện lợi trong việc muốn đặt vào vị trí nào đó.
- Bàn ghế làm việc: Với bất kỳ phòng làm việc nào cũng đa số có bàn và ghế làm việc. Ghế làm việc có thể di chuyển lên xuống trái phải trên sàn phòng làm việc theo nhu cầu của người dùng, tạo cảm giác thuận tiện, thoải mái và thư giãn.
- Quạt trần: Là một thiết bị cơ khí được lắp trên trần nhà, có chức năng làm mát và lưu thông không khí trong phòng, bao gồm một trục treo và cánh quạt có thể quay 360 độ liên tục với tốc độ nhanh.
- Tủ tài liệu: Gồm các ô tủ nhỏ chứa sách và các vật dụng trang trí… Bên cạnh đó có thêm cánh tủ có thể đóng mở theo nhu cầu của người dùng, bảo vệ các vật dụng, đồ dùng trên tủ luôn được sạch sẽ.
- Điều hòa: Là thiết bị kiểm soát nhiệt độ và độ ẩm, được gắn trên tường của phòng làm việc. Điều hòa có một cánh gió vẫy có thể đóng mở.
- Đồng hồ: Được treo trên tường phòng làm việc để có thể hiển thị và theo dõi thời gian. Trên mặt đồng hồ có 2 thanh kim đồng hồ quay 360 độ theo thời gian.

- Máy tính xách tay: Máy tính xách tay làm việc đặt trên bàn làm việc, có thể gập lên gập xuống (đóng/mở) để làm việc, học tập và giải trí.

1.3. Mô tả bố cục khung cảnh chung

Hình 1. 1: Bố cục khung cảnh chung

1.4. Mô tả kịch bản của chương trình

Ban đầu, camera của người nhìn sẽ bắt đầu từ vị trí ở bên ngoài cửa phòng làm việc như *Hình 1.1*, sau đó chúng ta sẽ di chuyển tầm nhìn của người dùng bằng các phím *"w, s, a, d"* để thay đổi góc nhìn tiến, lùi, sang trái hoặc sang phải sang phải và sử dụng chuột để di chuyển thay đổi góc nhìn camera. Đồng thời, cũng có thể sử dụng các phím *",”* để đưa camera lên trên hoặc phím *"i"* để đưa camera xuống dưới tùy theo nhu cầu thay đổi góc nhìn.

Người dùng sẽ sử dụng các tương tác với chuột và bàn phím như trên để di chuyển camera về phía phòng làm việc, khi tới trước cánh cửa chính, ta sẽ sử dụng phím *"/"* để mở và phím *"?"* để đóng cửa chính của phòng làm việc.

Khi di chuyển vào bên trong phòng làm việc, ta sẽ thấy các đồ dùng, vật dụng nội thất như: bàn ghế sofa, bàn làm việc, điều hòa, đồng hồ, cửa sổ, tủ, camera… Và bằng cách sử dụng các phím trên bàn phím ta có thể sử dụng các đồ dùng, vật dụng đó. Ngoài ra, điều kiện bên ngoài như ánh sáng, thời tiết, … hay việc phóng to, thu nhỏ cũng được mô tả, thiết kế đối với mô hình phòng việc. Cụ thể:

Phím	**Hành động / Chuyển động**
/	Mở cửa chính phòng làm việc
?	Đóng cửa chính phòng làm việc
6	Mở cửa sổ
7	Đóng cửa sổ
c	Bật/tắt bóng đèn
v	Trời sáng/tối
r	Di chuyển ghế làm việc sang phải
f	Di chuyển ghế làm việc sang trái
u	Di chuyển ghế làm việc lên trên (lại gần cửa chính)
p	Di chuyển ghế làm việc lùi xuống (ra xa cửa chính)
t	Mở cánh tủ tài liệu
T	Đóng cánh tủ tài liệu
Q	Bật/tắt quạt trần
m	Mở laptop trên bàn làm việc

M	Đóng laptop trên bàn làm việc
1	Bật/tắt điều hòa trong phòng
R	Di chuyển ghế sofa sang phải
F	Di chuyển ghế sofa sang trái
U	Di chuyển ghế sofa lên trên (lại gần cửa chính)
P	Di chuyển ghế sofa lùi xuống (ra xa cửa chính)
2	Bật/tắt camera
+	Phóng to vật thể
-	Thu nhỏ vật thể
g	Tịnh tiến vật thể về gần camera
h	Tịnh tiến vật thể ra xa camera

1.5. Phần mềm sử dụng

1.5.1. Visual Studio 2022

Visual Studio 2022 là phiên bản mới nhất trong dòng công cụ phát triển tích hợp (IDE) của Microsoft, được thiết kế để hỗ trợ các nhà phát triển trong việc xây dựng, kiểm thử, và triển khai phần mềm trên nhiều nền tảng khác nhau. Đây là một trong những công cụ phổ biến nhất hiện nay dành cho lập trình viên, đặc biệt trong phát triển ứng dụng Windows, web, và đồ họa.

Visual Studio 2022 cung cấp môi trường lý tưởng để phát triển ứng dụng sử dụng ngôn ngữ C++, với:

- Trình biên dịch C++ tiêu chuẩn: Tuân thủ các tiêu chuẩn C++ mới nhất (C++17, C++20).

- Công cụ IntelliSense: Gợi ý mã thông minh, giúp tăng năng suất lập trình.
- Hỗ trợ công cụ gỡ lỗi (Debugger): Xem giá trị biến, kiểm tra luồng chương trình, và xử lý lỗi hiệu quả.
- Hỗ trợ OpenGL: Cung cấp thư viện và công cụ giúp tích hợp OpenGL để phát triển đồ họa 3D.

Visual Studio 2022 là phiên bản 64-bit đầu tiên, giúp IDE hoạt động mượt mà hơn khi xử lý các dự án lớn. Giảm thiểu tình trạng chậm trễ và treo máy khi mở các dự án phức tạp.

Visual Studio 2022 còn có tính năng biên dịch nền (còn gọi là biên dịch gia tăng). Như mã đang được viết, Visual Studio biên dịch nó trong nền để cung cấp thông tin phản hồi về cú pháp và biên dịch lỗi, được đánh dấu bằng một gạch dưới gợn sóng màu đỏ. Biên dịch nền không tạo ra mã thực thi, vì nó đòi hỏi một trình biên dịch khác hơn là để sử dụng tạo ra mã thực thi. Biên dịch nền ban đầu được giới thiệu với Microsoft Visual Basic nhưng bây giờ đã được mở rộng cho tất cả các ngôn ngữ.

1.5.2. Thư viện GLEW

GLEW (OpenGL Extension Wrangler Library) là một thư viện mã nguồn mở được thiết kế để quản lý và sử dụng các phần mở rộng của OpenGL. Nó cung cấp cách tiếp cận nhất quán và đơn giản để truy cập các tính năng OpenGL tiên tiến mà không cần phải tự tay quản lý các hàm phần mở rộng.

- Tính năng chính:
  - Quản lý phần mở rộng OpenGL: GLEW hỗ trợ kiểm tra và sử dụng các tính năng bổ sung của OpenGL do các trình điều khiển đồ họa cung cấp.
  - Hỗ trợ đa nền tảng: Thư viện hoạt động trên Windows, macOS, và Linux.
  - Tự động tải hàm OpenGL: GLEW tự động tải các hàm cần thiết để sử dụng các phiên bản và phần mở rộng của OpenGL.
- Trong đề tài "Xây dựng chương trình mô phỏng phòng làm việc sử dụng OpenGL khả lập trình và Visual C++", GLEW đóng vai trò:

- Khởi tạo các hàm OpenGL hiện đại: Cung cấp khả năng sử dụng các hàm OpenGL mới như shader, buffer object, và các tính năng nâng cao khác.
- Tăng cường khả năng tương thích: Cho phép chương trình hoạt động trên các hệ thống với trình điều khiển đồ họa khác nhau.
- Tích hợp dễ dàng với Visual Studio: GLEW được sử dụng để thiết lập môi trường lập trình đồ họa nhanh chóng và hiệu quả.

- Ưu điểm:
  - Giảm bớt sự phức tạp trong việc quản lý các phần mở rộng OpenGL.
  - Tăng tính ổn định và khả năng mở rộng của chương trình đồ họa.

1.5.3.Thư viện FreeGLUT

FreeGLUT là một phiên bản mã nguồn mở của thư viện GLUT (OpenGL Utility Toolkit), được phát triển để thay thế GLUT và cung cấp các công cụ cần thiết để tạo giao diện đồ họa và quản lý vòng đời của ứng dụng OpenGL.

- Tính năng chính
  - Quản lý cửa sổ OpenGL: FreeGLUT giúp tạo và quản lý cửa sổ hiển thị OpenGL.
  - Xử lý sự kiện đầu vào: Hỗ trợ các sự kiện từ bàn phím, chuột, và các thiết bị ngoại vi khác.
  - Hỗ trợ vòng lặp chính: FreeGLUT cung cấp một vòng lặp chính cho ứng dụng, giúp dễ dàng quản lý và cập nhật khung hình.
- FreeGLUT đóng vai trò quan trọng trong việc:
  - Tạo giao diện hiển thị mô phỏng: Tạo cửa sổ và thiết lập môi trường làm việc để hiển thị các mô hình 3D của phòng làm việc.
  - Xử lý tương tác người dùng: Hỗ trợ nhận sự kiện từ bàn phím hoặc chuột để điều khiển camera hoặc tương tác với các đối tượng trong không gian mô phỏng.
  - Quản lý vòng lặp đồ họa: Cung cấp cơ chế để liên tục vẽ lại và cập nhật mô hình trong chương trình.

- Ưu điểm
    - Giao diện lập trình đơn giản, dễ sử dụng.
    - Tương thích với nhiều nền tảng và tích hợp tốt với các thư viện khác như GLEW.
    - Hỗ trợ phát triển nhanh các ứng dụng đồ họa 3D cơ bản và nâng cao.

CHƯƠNG II. CÀI ĐẶT CHƯƠNG TRÌNH

2.1. Kỹ thuật tạo mô hình phòng làm việc

Sử dụng hình khối cube để tạo ra các mô hình tương tự như các bức tường như trong thực tế. Từ 4 bức tường, 1 trần nhà và 1 sàn nhà được ghép lại với nhau trở thành một căn phòng làm việc với không gian thoáng đãng và thoải mái.

```
void room() {
        mvstack.push(model_mat_cpp);
        // trần nhà
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(0.0, 2.9, -3.5)) *
    scale(vec3(5.0, 0.01, 7.0));
        setInt("color", 14);
        cube();
        model_mat_cpp = mvstack.pop();
        // tường trái
        mvstack.push(model_mat_cpp);
        setInt("color", 15);
        model_mat_cpp = model_mat_cpp * translate(vec3(0.0, 0.9, 0)) *
    scale(vec3(5.0, 4.0, 0.01));
        cube();
        model_mat_cpp = mvstack.pop();
        // sàn
        mvstack.push(model_mat_cpp);
        setInt("color", 15);
        model_mat_cpp = model_mat_cpp * translate(vec3(0.0, -1.1, -3.5)) *
    scale(vec3(5.0, 0.01, 7.0));
        cube();
```

```
        model_mat_cpp = mvstack.pop();

        // mặt trước
        //mảnh tường to thứ nhất
        mvstack.push(model_mat_cpp);
        setInt("color", 15);
        model_mat_cpp = model_mat_cpp * translate(vec3(-2.5, 1.65, -3.5)) *
    scale(vec3(0.01, 2.5, 7.0));
        cube();
        model_mat_cpp = mvstack.pop();
        //mảnh tường nhỏ bên trái
        mvstack.push(model_mat_cpp);
        setInt("color", 15);
        model_mat_cpp = model_mat_cpp * translate(vec3(-2.5, -0.35, -0.5))
    * scale(vec3(0.01, 1.5, 1));
        cube();
        model_mat_cpp = mvstack.pop();
        //mảnh tường nhỏ bên phải
        mvstack.push(model_mat_cpp);
        setInt("color", 15);
        model_mat_cpp = model_mat_cpp * translate(vec3(-2.5, -0.35, -4.5))
    * scale(vec3(0.01, 1.5, 5));
        cube();
        model_mat_cpp = mvstack.pop();
        model_mat_cpp = mvstack.pop();
        // mặt sau
        mvstack.push(model_mat_cpp);
        setInt("color", 15);
```

```
    model_mat_cpp = model_mat_cpp * translate(vec3(2.5, 0.9, -3.5)) *
scale(vec3(0.01, 4.0, 7.0));
    cube();
    model_mat_cpp = mvstack.pop();

    // tường phải
//  mảnh tường to trên
    mvstack.push(model_mat_cpp);
    setInt("color", 15);
    model_mat_cpp = model_mat_cpp * translate(vec3(0.0, 2.15, -7)) *
scale(vec3(5.0, 1.5, 0.01));
    cube();
    model_mat_cpp = mvstack.pop();
    //mảnh tường to dưới
    mvstack.push(model_mat_cpp);
    setInt("color", 15);
    model_mat_cpp = model_mat_cpp * translate(vec3(0.0,-0.35, -7)) *
scale(vec3(5.0, 1.5, 0.01));
    cube();
    model_mat_cpp = mvstack.pop();
    //mảnh tường to trái
    mvstack.push(model_mat_cpp);
    setInt("color", 15);
    model_mat_cpp = model_mat_cpp * translate(vec3(1, 0.9, -7)) *
scale(vec3(3, 1.0, 0.01));
    cube();
    model_mat_cpp = mvstack.pop();
    // mảnh tường nhỏ  phải
    mvstack.push(model_mat_cpp);
```

```
        setInt("color", 15);
        model_mat_cpp = model_mat_cpp * translate(vec3(-2, 0.9, -7)) *
    scale(vec3(1, 1.0, 0.01));
        cube();
        model_mat_cpp = mvstack.pop();
    }
```

2.2. Kỹ thuật tạo mô hình cánh cửa chính

Cánh cửa chính được tạo bằng các đối tượng hình học như hình khối cube. Kết hợp với các phép biến đổi như translate, rotate để tạo động tác mở cửa và đóng cửa.

Cửa chính có các khung bản lề và tay nắm cửa, tạo cảm giác dễ dàng đóng mở.

```
void door() {
        mvstack.push(model_mat_cpp);
        // Xoay cánh cửa
        mvstack.push(model_mat_cpp);
        setInt("color", 18);
        model_mat_cpp = model_mat_cpp
                * translate(vec3(-2.5, -0.35, -1.5))
                * translate(vec3(-0.025, 0.0, 0.5))
                * rotate_y(door_angle)
                * translate(vec3(0.025, 0.0, -0.5))
                * scale(vec3(0.05, 1.5, 1.0));
        cube();  // Vẽ cánh cửa
```

```
        model_mat_cpp = mvstack.pop();
        // Tay nắm cửa hình chữ nhật
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp
                * translate(vec3(-2.53, -0.3, -1))
                * rotate_y(door_angle)
                * translate(vec3(0.01, 0.0, -0.4))
                * translate(vec3(0.01, 0.0, -0.4))
                * scale(vec3(0.14, 0.4, 0.05));
        setInt("color", 2); // Tay nắm có màu khác
        cube();  // Vẽ tay nắm
        model_mat_cpp = mvstack.pop();
        // Khung cửa trái
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(-2.5, -0.35, -1)) *
scale(vec3(0.05, 1.5, 0.05));  // Khung trái, cao hơn cánh cửa
        setInt("color", 1);
        cube();
        model_mat_cpp = mvstack.pop();
        // Khung cửa phải
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(-2.5, -0.35, -2)) *
scale(vec3(0.05, 1.5, 0.05));  // Khung phải
        setInt("color", 1);
```

```
        cube();
        model_mat_cpp = mvstack.pop();
        // Khung cửa trên
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(-2.5, 0.4, -1.5)) *
scale(vec3(0.05, 0.05, 1.0)); // Khung trên
        setInt("color", 1);
        cube();
        model_mat_cpp = mvstack.pop();
        model_mat_cpp = mvstack.pop();
}
```

2.3. Kỹ thuật tạo mô hình cửa sổ

Cửa sổ được xây dựng tương tự như cánh cửa chính. Tuy nhiên, vật liệu được làm bằng kính trong suốt, tạo cảm giác chân thực.

```
void windown(float tmp) {
        //khung tren
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(1.6, 0.275, -0.9))
* scale(vec3(1.05, 0.05, 0.05));
        setInt("color", 1);
        cube();
        //khung duoi
        model_mat_cpp = mvstack.pop();
```

```
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(1.6, -0.35, -0.9))
* scale(vec3(1.05, 0.05, 0.05));
        setInt("color", 1);
        cube();
        model_mat_cpp = mvstack.pop();
        //khung trai
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(2.1, -0.05, -0.9))
* scale(vec3(0.05, 0.65, 0.05));
        setInt("color", 1);
        cube();
        model_mat_cpp = mvstack.pop();
        //khung phai
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(1.05, -0.04, -0.9))
* scale(vec3(0.05, 0.67, 0.05));
        setInt("color", 1);
        cube();
        model_mat_cpp = mvstack.pop();
        //kinh
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp
                * translate(vec3(1.58, -0.04, -0.9))
```

```
                * translate(vec3(0.0, 0.3, 0))
                * rotate_x(tmp)
                * translate(vec3(0.0, -0.3, 0))
                * scale(vec3(1.0, 0.6, 0.02));
        setInt("color", 9);
        cube();
        model_mat_cpp = mvstack.pop();
        //tay nam
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp
                * translate(vec3(1.55, 0.25, -0.89))
                * rotate_x(tmp)
                * translate(vec3(0.0, -0.25, 0.0))
                * translate(vec3(0.0, -0.15, 0.0))
                * scale(vec3(0.1, 0.02, 0.02));
        setInt("color", 1);
        cube();
        model_mat_cpp = mvstack.pop();
}
```

Các bộ phận của cửa sổ sử dụng hàm windows và truyền vào các tham số phù hợp để tạo nên từng bộ phận của cửa sổ.

```
void windowns(float tx, float ty, float tz, float sx, float sy, float sz, float tmp,
int j) {
        mvstack.push(model_mat_cpp);
```

```
        model_mat_cpp = model_mat_cpp * translate(vec3(tx, ty, tz)) *
scale(vec3(sx, sy, sz)) * rotate_y(360 / j);
        windown(tmp);
        model_mat_cpp = mvstack.pop();
}
```

2.4. Kỹ thuật tạo mô hình bóng đèn

Sử dụng các hình khối cube để tạo ra từng bộ phận của khung chứa và bóng đèn tương tự như mô hình cánh cửa.

```
void light() {
        mvstack.push(model_mat_cpp);
        //bong
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(1.35, 3, 0)) *
    scale(vec3(0.05, 0.05, 1.86));
        setInt("color", 98);
        cube();
        model_mat_cpp = mvstack.pop();
        //de
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(1.42, 3, 0)) *
    scale(vec3(0.03, 0.1, 2));
        setInt("color", 1);
        cube();
        model_mat_cpp = mvstack.pop();
```

```
        //tai phai
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(1.35, 3,- 0.95)) *
    scale(vec3(0.1, 0.1, 0.07));
        setInt("color", 1);
        cube();
        model_mat_cpp = mvstack.pop();
        //tai trai
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(1.35, 3, 0.95)) *
    scale(vec3(0.1, 0.1, 0.07));
        setInt("color", 1);
        cube();
        model_mat_cpp = mvstack.pop();
        model_mat_cpp = mvstack.pop();
}
```

Các bộ phận của khung và bóng đèn sử dụng hàm lights và truyền vào hàm các tham số phù hợp để tạo nên từng bộ phận của bóng đèn.

```
void lights(float tx, float ty, float tz, float sx, float sy, float sz, int tmp) {
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(tx, ty, tz)) *
scale(vec3(sx, sy, sz)) * rotate_z(90.0f * tmp);
        light();
        model_mat_cpp = mvstack.pop();
}
```

2.5. Kỹ thuật tạo mô hình camera

Camera được thiết lập để theo dõi, quan sát phòng làm việc với góc xoay là camera_angle – vừa đủ để quan sát được cả căn phòng bằng phép biến đổi xoay. Biến cờ để bật/tắt camera là khi vào phòng là true, vì vậy khi vào phòng camera sẽ được bật.

```
void camera() {
        mvstack.push(model_mat_cpp);
        // Base
        mvstack.push(model_mat_cpp);
        setInt("color", 99);
        model_mat_cpp = model_mat_cpp * scale(vec3(0.03, 0.03, 0.12));
        cube();
        model_mat_cpp = mvstack.pop();
        // camera
        mvstack.push(model_mat_cpp);
        setInt("color", 2);
        model_mat_cpp = model_mat_cpp * rotate_y(camera_angle) *
    translate(vec3(0.0, 0.0, 0.1)) * scale(vec3(0.1, 0.1, 0.2));
        cube();
        model_mat_cpp = mvstack.pop();
        model_mat_cpp = mvstack.pop();
}
```

Các bộ phận của camera sử dụng hàm cameras và truyền vào hàm các tham số phù hợp để tạo nên từng bộ phận của camera.

```
void cameras(float tx, float ty, float tz, float sx, float sy, float sz) {
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(tx, ty, tz)) *
scale(vec3(sx, sy, sz));
        camera();
        model_mat_cpp = mvstack.pop();
}
```

2.6. Kỹ thuật tạo mô hình bàn ghế làm việc

Phòng làm việc luôn phải có bàn và ghế làm việc. Sử dụng các hình khối cube để tạo ra các bộ phận của bàn ghế làm việc.

```
void chair() {
        mvstack.push(model_mat_cpp);
        //mat ghe
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(0.0, 0.0, 0.08)) *
    scale(vec3(0.6, 0.05, 0.77));
        setInt("color", 1);
        cube();
        model_mat_cpp = mvstack.pop();
        //tua lung
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(0.0, 0.3, -0.275))
    * scale(vec3(0.5, 0.6, 0.02));
        setInt("color", 1);
```

```
    cube();
    model_mat_cpp = mvstack.pop();
    //tay vin trai
    mvstack.push(model_mat_cpp);
    model_mat_cpp = model_mat_cpp * translate(vec3(-0.275, 0.325, -0.275)) * scale(vec3(0.05, 0.6, 0.05));
    setInt("color", 2);
    cube();
    model_mat_cpp = mvstack.pop();
    //tay vin phai
    mvstack.push(model_mat_cpp);
    model_mat_cpp = model_mat_cpp * translate(vec3(0.275, 0.325, -0.275)) * scale(vec3(0.05, 0.6, 0.05));
    setInt("color", 2);
    cube();
    model_mat_cpp = mvstack.pop();
    //chan phai tren
    mvstack.push(model_mat_cpp);
    model_mat_cpp = model_mat_cpp * translate(vec3(0.275, -0.325, 0.375)) * scale(vec3(0.05, 0.6, 0.05));
    setInt("color", 2);
    cube();
    model_mat_cpp = mvstack.pop();
    //chan trai tren
```

```
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(-0.275, -0.325,
    0.375)) * scale(vec3(0.05, 0.6, 0.05));
        setInt("color", 2);
        cube();
        model_mat_cpp = mvstack.pop();
        //chan phai duoi
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(0.275, -0.325, -
    0.275)) * scale(vec3(0.05, 0.6, 0.05));
        setInt("color", 2);
        cube();
        model_mat_cpp = mvstack.pop();
        //chan trai duoi
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(-0.275, -0.325, -
    0.275)) * scale(vec3(0.05, 0.6, 0.05));
        setInt("color", 2);
        cube();
        model_mat_cpp = mvstack.pop();
        model_mat_cpp = mvstack.pop();
}
void matban() {
        mvstack.push(model_mat_cpp);
```

```
        setInt("color", 1);


        mat4 instance = identity_mat4();
        instance = translate(vec3(0.0, chieucaoban, chieurongmat / 2.0)) *
scale(vec3(chieudaimat, dodaymatban, chieurongmat));


        mat4 model_torso = model_mat_cpp * instance;
        glUniformMatrix4fv(model_mat_location, 1, GL_FALSE,
model_torso.m);
        glDrawArrays(GL_TRIANGLES, 0, 36);
        model_mat_cpp = mvstack.pop();
}
void chanban() {
        mvstack.push(model_mat_cpp);
        setInt("color", 2);
        mat4 instance = identity_mat4();
        instance = translate(vec3(0, chieucaoban / 2.0, 0)) *
scale(vec3(dodaychanban, chieucaoban, dodaychanban));
        mat4 model_left_upper_arm = model_mat_cpp * instance;
        glUniformMatrix4fv(model_mat_location, 1, GL_FALSE,
model_left_upper_arm.m);
        glDrawArrays(GL_TRIANGLES, 0, 36);
        model_mat_cpp = mvstack.pop();
}
```

```
void nganban()
{
	mvstack.push(model_mat_cpp);
	setInt("color", 1);
	mat4 instance = identity_mat4();
	instance = translate(vec3(0.25, chieucaongan - 1, chieurongmat / 2.0)) * scale(vec3((chieudaimat - (dodaychanban * 2.0)) / 2, dodaymatban, chieurongmat - (dodaychanban * 2.0)));
	mat4 model_torso = model_mat_cpp * instance;
	glUniformMatrix4fv(model_mat_location, 1, GL_FALSE, model_torso.m);
	glDrawArrays(GL_TRIANGLES, 0, 36);
	model_mat_cpp = mvstack.pop();
}
void tamchanban() {
	mvstack.push(model_mat_cpp);
	setInt("color", 1);
	mat4 instance = identity_mat4();
	instance = translate(vec3(0.25, chieucaoban - 0.14, chieurongmat / 2 - 0.3)) * scale(vec3((chieudaimat - (dodaychanban * 2.0)) / 2, 0.28, dodaymatban));
	mat4 model_torso = model_mat_cpp * instance;
	glUniformMatrix4fv(model_mat_location, 1, GL_FALSE, model_torso.m);
```

```
        glDrawArrays(GL_TRIANGLES, 0, 36);
        model_mat_cpp = mvstack.pop();
}
```

Các bộ phận của bàn ghế làm việc sử dụng hàm chairs và hàm veban, sau đó truyền vào hàm các tham số phù hợp để tạo nên từng bộ phận của bàn và ghế làm việc. Đồng thời tạo các biến số tăng giảm giá trị, tạo nên chuyển động tịnh tiến cho chiếc ghế.

```
void chairs(float tx, float ty, float tz, float sx, float sy, float sz, int j) {
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(work_chairx, 0.0, work_chairz)) * translate(vec3(tx, ty, tz)) * scale(vec3(sx, sy, sz)) * rotate_y(360 / j);
        chair();
        model_mat_cpp = mvstack.pop();
}

void veban(float tx, float ty, float tz, float sx, float sy, float sz, GLfloat ngankeo) {
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(tx, ty, tz)) * scale(vec3(sx, sy, sz));
        ban::veban();
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(0, 0, ngankeo));
        ban::nganban();
        model_mat_cpp = mvstack.pop();
        model_mat_cpp = mvstack.pop();
}
```

2.7. Kỹ thuật tạo mô hình tủ tài liệu

Tương tự như mặt bàn hay ngăn bàn,… các thanh gỗ của tủ tài liệu được tạo nên dễ dàng. Cánh tủ cũng được tạo nên bằng chất liệu là kính.

```
void san() {
        mvstack.push(model_mat_cpp);
        setInt("color", 1);
        model_mat_cpp = model_mat_cpp * scale(vec3(chieungangtu,
    dodaytu, chieurongtu));
        cube();
        model_mat_cpp = mvstack.pop();
}
void canhbentu() {
        mvstack.push(model_mat_cpp);
        setInt("color", 1);
        model_mat_cpp = model_mat_cpp * scale(vec3(dodaytu, chieucaotu,
    chieurongtu));
        cube();
        model_mat_cpp = mvstack.pop();
}
void matlungtu() {
        mvstack.push(model_mat_cpp);
        setInt("color", 1);
        model_mat_cpp = model_mat_cpp * translate(vec3(0, chieucaotu /
    2.0, chieurongtu / 2.0)) * scale(vec3(chieungangtu, chieucaotu, dodaytu));
```

```
        cube();
        model_mat_cpp = mvstack.pop();
}
void canhtu() {
        float chieucaocanh = chieucaotu - dodaytu;
        mvstack.push(model_mat_cpp);
        setInt("color", 17);
        model_mat_cpp = model_mat_cpp * scale(vec3(chieungangtu / 2.0,
    chieucaocanh, dodaytu));
        cube();
        model_mat_cpp = mvstack.pop();
        mvstack.push(model_mat_cpp);
        setInt("color", 17);
        model_mat_cpp = model_mat_cpp * scale(vec3(chieungangtu / 2.0 -
    0.04, chieucaocanh - 0.03, dodaytu)) * translate(vec3(0, 0, 0.01));
        cube();
        model_mat_cpp = mvstack.pop();
}
```

Các bộ phận của tủ tài liệu sử dụng hàm draw_tu và truyền vào hàm các tham số phù hợp để tạo ghép nên từng bộ phận của tủ chứa tài liệu. Cùng với đó là sử dụng hàm xoay trên trục y để tạo nên hiệu ứng đóng mở cánh tủ.

```
void draw_tu(float tx, float ty, float tz, float sx, float sy, float sz, GLfloat
angle_canh1)
{
        mvstack.push(model_mat_cpp);
```

```
        model_mat_cpp = model_mat_cpp * translate(vec3(tx, ty, tz)) *
scale(vec3(sx, sy, sz));
        tu::vetu();
        // canh tu 1 - canh trai
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(0.4, 0.6, -0.24)) *
rotate_y(angle_canh1) * translate(vec3(-0.2, 0, 0));
        tu::canhtu();
        model_mat_cpp = mvstack.pop();
        // canh tu 2 - canh phai
        GLfloat angle_canh2 = angle_canh1 * (-1);
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(-0.4, 0.6, -0.24)) *
rotate_y(angle_canh2) * translate(vec3(0.2, 0, 0));
        tu::canhtu();
        model_mat_cpp = mvstack.pop();


        model_mat_cpp = mvstack.pop();
}
```

2.8. Kỹ thuật tạo mô hình đồng hồ

Đồng hồ được tạo nên từ mặt đồng hồ, trục, kim giờ, kim phút và các viền gỗ bao bọc mặt đồng hồ.

```
void mat() {
        mvstack.push(model_mat_cpp);
        setInt("color", 98);
        model_mat_cpp = model_mat_cpp * scale(vec3(1, 1, 0.1));
```

```
        cube();
        model_mat_cpp = mvstack.pop();
}
void vien() {
        mvstack.push(model_mat_cpp);
        setInt("color", 1);
        model_mat_cpp = model_mat_cpp * scale(vec3(0.1, 1, 0.1));
        cube();
        model_mat_cpp = mvstack.pop();
}
void hour_index() {
        mvstack.push(model_mat_cpp);
        setInt("color", 99);
        model_mat_cpp = model_mat_cpp * scale(vec3(0.02, 0.06, 0.01));
        cube();
        model_mat_cpp = mvstack.pop();
}
void truc() {
        mvstack.push(model_mat_cpp);
        setInt("color", 99);
        model_mat_cpp = model_mat_cpp * scale(vec3(0.06, 0.06, 0.12));
        cube();
        model_mat_cpp = mvstack.pop();
```

```
}
void kim() {
        mvstack.push(model_mat_cpp);
        setInt("color", 3);
        model_mat_cpp = model_mat_cpp * translate(vec3(0, 0.11, 0.08)) *
scale(vec3(0.02, 0.25, 0.02));
        cube();
        model_mat_cpp = mvstack.pop();
}
```

Các bộ phận đó được ghép lại với nhau bằng hàm vedongdo và các tham số, phép biến đổi xoay, tạo nên hiệu ứng quay kim đồng hồ 360 độ theo thời gian.

```
void vedongho(float tx, float ty, float tz, float sx, float sy, float sz) {
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(tx, ty, tz)) *
scale(vec3(sx, sy, sz));
        mvstack.push(model_mat_cpp);
        dongho::mat();
        model_mat_cpp = mvstack.pop();
        //ve vien 1
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(0.45, 0, 0.05));
        dongho::vien();
        model_mat_cpp = mvstack.pop();
        //ve vien 2
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(-0.45, 0, 0.05));
```

```
        dongho::vien();
        model_mat_cpp = mvstack.pop();
        //ve vien 3
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(0, 0.45, 0.05)) *
rotate_y(90) * rotate_x(90);
        dongho::vien();
        model_mat_cpp = mvstack.pop();
        //ve vien 4
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(0, -0.45, 0.05)) *
rotate_y(90) * rotate_x(90);
        dongho::vien();
        model_mat_cpp = mvstack.pop();
        //index
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(0, 0.34, 0.06));
        dongho::hour_index();
        model_mat_cpp = mvstack.pop();

        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(0.34, 0, 0.06)) *
rotate_z(90);
        dongho::hour_index();
        model_mat_cpp = mvstack.pop();

        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(-0.34, 0, 0.06)) *
rotate_z(-90);
```

```
        dongho::hour_index();
        model_mat_cpp = mvstack.pop();

        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(0, -0.34, 0.06));
        dongho::hour_index();
        model_mat_cpp = mvstack.pop();
        //truc
        dongho::truc();
        //kim gio
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * rotate_z(hour_angle) *
scale(vec3(1.4, 1, 1));
        dongho::kim();
        model_mat_cpp = mvstack.pop();
        //kim phut
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * rotate_z(min_angle) *
scale(vec3(1, 1.4, 1));
        dongho::kim();
        model_mat_cpp = mvstack.pop();

        model_mat_cpp = mvstack.pop();
```

2.9. Kỹ thuật tạo mô hình điều hòa

Điều hòa gồm các hộp khung và cửa thông gió có thể điều hòa lên xuống thông qua phép xoay.

```
void conditioner() {
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * scale(vec3(0.2, 0.2, 0.2));
    //hộp to
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(11, -0.5, -9)) *
    scale(vec3(8.4, 2.3, 2));
        setInt("color", 98);
        cube();
        model_mat_cpp = mvstack.pop();
        //hộp nhỏ
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(11, -1.8, -9)) *
    scale(vec3(8.4, 0.5, 1.6));
        setInt("color", 98);
        cube();
        model_mat_cpp = mvstack.pop();
        //hộp dưới
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(11, -2, -9)) *
    scale(vec3(8.4, 0.1, 2));
        setInt("color", 98);
        cube();
        model_mat_cpp = mvstack.pop();
```

```
        //canh ben trai
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(6.8, -0.67, -9)) *
scale(vec3(0.01, 2.7, 2));
        setInt("color", 2);
        cube();
        model_mat_cpp = mvstack.pop();
        //canh ben phai
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(15.2, -0.67, -9)) *
scale(vec3(0.01, 2.7, 2));
        setInt("color", 2);
        cube();
        model_mat_cpp = mvstack.pop();
        //cua
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp
                * translate(vec3(11, -1.82, -8.0))
                * translate(vec3(0.0, -0.25, 0.0))
                * rotate_x(angle_conditioner)
                * translate(vec3(0.0, 0.25, 0.0))
                * scale(vec3(8.4, 0.4, 0.01));
        mvstack.push(model_mat_cpp);
        setInt("color", 2);
```

```
        cube();
        model_mat_cpp = mvstack.pop();
        model_mat_cpp = mvstack.pop();
        model_mat_cpp = mvstack.pop();
}
```

Tương tự như các mô hình khác, hàm dieuhoa có chức năng tạo nên một mô hình điều hòa hoàn chỉnh.

```
void dieuhoa(float qx, float qy, float qz) {
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(qx, qy, qz));
        conditioner();
        model_mat_cpp = mvstack.pop();
}
```

2.10. Kỹ thuật tạo mô hình máy tính xách tay

Máy tính xách tay được tạo nên từ 2 mảnh ghép tương tự như trong thực tế. Màn hình có thể gập lên xuống thông qua chức năng của phép biến đổi xoay.

```
void laptop() {
        mvstack.push(model_mat_cpp);
        //ban phim
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(0.2, 0.6, 0.4)) *
    scale(vec3(0.12, 0.02, 0.27));
        setInt("color", 10);
```

```
        cube();
        model_mat_cpp = mvstack.pop();
        //man hinh
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp
                * translate(vec3(0.2, 0.62, 0.4))
               * translate(vec3(-0.06, 0.0, 0.0))
                * rotate_z(lap_angle)
                * translate(vec3(0.06, 0.0, 0.0))
                * scale(vec3(0.12, 0.01, 0.27));
        setInt("color", 10);
        cube();
        model_mat_cpp = mvstack.pop();
        model_mat_cpp = mvstack.pop();
}
```

2.11. Kỹ thuật tạo mô hình bàn ghế sofa

```
void matbansofa() {
        mvstack.push(model_mat_cpp);
        setInt("color", 2);
        mat4 instance = identity_mat4();
        instance = translate(vec3(0.0, 0.5, 0.6/ 2.0)) * scale(vec3(1, 0.05, 0.6));
        mat4 model_torso = model_mat_cpp * instance;
```

```
        glUniformMatrix4fv(model_mat_location, 1, GL_FALSE, model_torso.m);
        glDrawArrays(GL_TRIANGLES, 0, 36);
        model_mat_cpp = mvstack.pop();
}
void chanbansofa() {
        mvstack.push(model_mat_cpp);
        setInt("color", 10);
        mat4 instance = identity_mat4();
        instance = translate(vec3(0, 0.5 / 2.0, 0)) * scale(vec3(0.17, 0.5, 0.4));
        mat4 model_left_upper_arm = model_mat_cpp * instance;
        glUniformMatrix4fv(model_mat_location, 1, GL_FALSE, model_left_upper_arm.m);
        glDrawArrays(GL_TRIANGLES, 0, 36);
        model_mat_cpp = mvstack.pop();
}
void drawSofa() {
        mvstack.push(model_mat_cpp);
        //ngồi
        setInt("color", 20);
        model_mat_cpp = model_mat_cpp * translate(vec3(0, 0.1, 0)) * scale(vec3(1.5, 0.2, 0.7));
        cube();
        model_mat_cpp = mvstack.pop();
```

```
        // đệm ngồi
        mvstack.push(model_mat_cpp);
        setInt("color", 19);
        model_mat_cpp = model_mat_cpp * translate(vec3(0, 0.25, 0.05)) *
scale(vec3(1.3, 0.1, 0.6));
        cube();
        model_mat_cpp = mvstack.pop();
        // tựa lưng
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(0, 0.4, -0.3)) *
scale(vec3(1.5, 0.72, 0.1));
        setInt("color", 20);
        cube();
        model_mat_cpp = mvstack.pop();
        // vịn trái
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(-0.7, 0.35, 0)) *
scale(vec3(0.1, 0.35, 0.7));
        setInt("color", 20);
        cube();
        model_mat_cpp = mvstack.pop();
        // vịn phải
        mvstack.push(model_mat_cpp);
```

```
        model_mat_cpp = model_mat_cpp * translate(vec3(0.7, 0.35, 0)) *
scale(vec3(0.1, 0.35, 0.7));
        setInt("color", 20);
        cube();
        model_mat_cpp = mvstack.pop();
}
```

Các bộ phận của bàn ghế sofa được ghép lại với nhau tương tự như bàn ghế làm việc thông qua hàm sofa và hàm vebansofa.

```
void sofa(float tx, float ty, float tz, float sx, float sy, float sz, int j) {
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(ghe_sofax, -0.1,
ghe_sofaz)) * translate(vec3(tx, ty, tz)) * scale(vec3(sx, sy, sz)) * rotate_y(-
180 / j);
        drawSofa();
        model_mat_cpp = mvstack.pop();
}

void vebansofa(float tx, float ty, float tz, float sx, float sy, float sz,int j) {
                mvstack.push(model_mat_cpp);
                model_mat_cpp = model_mat_cpp * translate(vec3(tx, ty, tz)) *
scale(vec3(sx, sy, sz)) * rotate_y(-180 / j);
                bansofa::vebansofa();
                model_mat_cpp = mvstack.pop();
}
```

2.12. Kỹ thuật tạo mô hình quạt trần

Bao gồm các bộ phận như trụ quạt, thân quạt và 4 cánh quạt. Phép biến đổi scale với tỉ lệ 1:1 tạo ra 4 cánh quạt như nhau đã được sử dụng. Cùng với

phép biến đổi tỉ lệ là phép xoay làm cho các cánh quạt quay 360 độ với tốc độ nhanh.

```
void fan_blade_x() {
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(0, 0, 0))
                * scale(vec3(2.0f, 0.1f, 0.6f)) * rotate_x(30);
        setInt("color", 16);
        cube();
        model_mat_cpp = mvstack.pop();
}
void fan_blade_z() {
        mvstack.push(model_mat_cpp);
        setInt("color", 16);
        model_mat_cpp = model_mat_cpp * translate(vec3(0, 0, 0))
                * scale(vec3(0.6f, 0.1f, 2.0f)) * rotate_z(30);
        cube();
        model_mat_cpp = mvstack.pop();
}
void fan_base() {
        mvstack.push(model_mat_cpp);
        setInt("color", 16);
        model_mat_cpp = model_mat_cpp * translate(vec3(0, 0, 0))
                * scale(vec3(0.8f, 0.2f, 0.8f));
```

```
        cube();
        model_mat_cpp = mvstack.pop();
}
```

Sử dụng hàm quat và truyền vào các tham số để ghép các bộ phận tạo nên mô hình quạt trần.

```
void quat(float qx, float qy, float qz) {
        mvstack.push(model_mat_cpp);
        model_mat_cpp = model_mat_cpp * translate(vec3(qx, qy, qz));
        fan::ceiling_fan();
        model_mat_cpp = mvstack.pop();
}
```

2.13. Kỹ thuật điều khiển camera

- Tham số eye là vị trí mắt nhìn của camera.
- Tham số at là vị trí mắt nhìn của camera nhìn tới, ở đây vị trí mắt nhìn tới là tọa độ (cameraX + sin(yaw), cameraY - pitch, cameraZ - cos(yaw) - 0.3f).
- Tham số up là vector chỉ hướng của camera, thường là (0.0, 1.0, 0.0).
- Sử dụng phép chiếu phối cảnh perspective:
  + Các đối tượng ở xa trông nhỏ hơn so với các đối tượng ở gần, tạo cảm giác chiều sâu.
  + Giá trị fov càng lớn thì góc nhìn càng rộng hơn và ngược lại giá trị nhỏ thì làm cho góc nhìn hẹp hơn.
  + aspect là tỉ lệ khung hình của màn hình, được tính bằng chiều rộng/chiều cao.
  + znear là khoảng cách từ camera đến mặt phẳng cắt gần. Các đối tượng nằm giữa mặt phẳng này và camera sẽ không được hiển thị.
  + zfar là khoảng cách từ camera đến mặt phẳng cắt xa. Các đối tượng nằm ngoài khoảng từ znear đến zfar sẽ không được hiển thị.

```
vec3 eye(cameraX, cameraY, cameraZ),
     at(cameraX + sin(yaw), cameraY - pitch, cameraZ - cos(yaw) - 0.3f),
     up(0.0f, 1.0f, 0.0f);


  view_mat_cpp = lookat(eye, at, up);
  projection_mat_cpp = perspective(fov, aspect, znear, zfar);
  glUniformMatrix4fv(view_mat_location, 1, GL_FALSE,
  view_mat_cpp.m);
   glUniformMatrix4fv(projection_mat_location, 1, GL_FALSE,
   projection_mat_cpp.m);
```

2.14. Kỹ thuật chiếu sáng

Khởi tạo các tham số chiếu sáng và phản chiếu, mỗi tham số tương ứng với các cấu trúc của chiếu sáng phản chiếu:

- Vị trí và hướng ánh sáng:
- light_position_world_1: Vị trí của nguồn sáng trong không gian thế giới.
- light_position_eye_1: Vị trí của nguồn sáng trong không gian mắt (eye space), được tính bằng cách nhân với ma trận nhìn (view_mat_shader).
- distance_to_light_eye_1: Vector từ điểm trên bề mặt đến nguồn sáng.
- direction_to_light_eye_1: Vector đơn vị chỉ hướng từ bề mặt đến nguồn sáng.
- Quan sát và pháp tuyến:
- position_eye: Vị trí điểm trên bề mặt trong không gian mắt.
- n_eye: Pháp tuyến tại điểm trên bề mặt, đã được chuẩn hóa.
- position_viewer: Vị trí của người quan sát trong không gian mắt (thường là (0, 0, 0)).

- Thành phần ánh sáng: khuếch tán và phản chiếu.

```
if (enable_light_1) {
    vec3 light_position_world_1 = vec3(0.0, 2.27, -3.0);
    vec3 light_position_eye_1 = vec3(view_mat_shader * vec4(light_position_world_1, 1.0));
    vec3 position_viewer = vec3(0.0, 0.0, 0.0);
    vec3 n_eye = normalize(normal_eye);

    vec3 distance_to_light_eye_1 = light_position_eye_1 - position_eye;
    vec3 direction_to_light_eye_1 = normalize(distance_to_light_eye_1);
    vec3 surface_to_viewer_eye_1 = normalize(position_viewer - position_eye);

    float dot_prod_1 = max(dot(direction_to_light_eye_1, n_eye), 0.0);
    vec3 Ld_1 = vec3(0.7, 0.7, 0.7);
    vec3 Kd_1 = vec3(1.0, 1.0, 1.0);
    vec3 Id_1 = Ld_1 * Kd_1 * dot_prod_1;

    vec3 half_way_eye_1 = normalize(surface_to_viewer_eye_1 + direction_to_light_eye_1);
    float dot_prod_specular_1 = max(dot(half_way_eye_1, n_eye), 0.0);
    float specular_factor_1 = pow(dot_prod_specular_1, 20.0);
    vec3 Ls_1 = vec3(0.2, 0.2, 0.2);
    vec3 Ks_1 = vec3(1.0, 1.0, 1.0);
```

```
        vec3 Is_1 = Ls_1 * Ks_1 * specular_factor_1;


        vec3 La_1 = vec3(0.2, 0.2, 0.2);
        vec3 Ka_1 = vec3(1.0, 1.0, 1.0);
        vec3 Ia_1 = La_1 * Ka_1;


        final_color += (Is_1 + Id_1 + Ia_1);
}
else{
        vec3 light_position_world_1 = vec3(0.0, 2.27, 1.0);
        vec3 light_position_eye_1 = vec3(view_mat_shader *
    vec4(light_position_world_1, 1.0));
        vec3 position_viewer = vec3(0.0, 0.0, 0.0);
        vec3 n_eye = normalize(normal_eye);


        vec3 distance_to_light_eye_1 = light_position_eye_1 - position_eye;
        vec3 direction_to_light_eye_1 = normalize(distance_to_light_eye_1);
        vec3 surface_to_viewer_eye_1 = normalize(position_viewer -
    position_eye);


        float dot_prod_1 = max(dot(direction_to_light_eye_1, n_eye), 0.0);
        vec3 Ld_1 = vec3(0,0,0);
        vec3 Kd_1 = vec3(0,0,0);
        vec3 Id_1 = Ld_1 * Kd_1 * dot_prod_1;
```

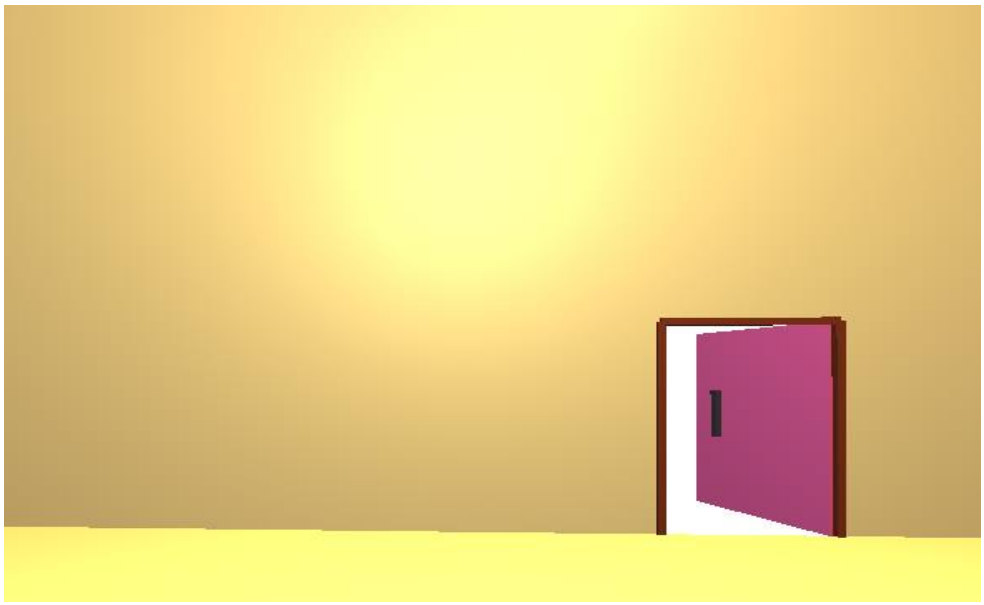
```
    vec3 half_way_eye_1 = normalize(surface_to_viewer_eye_1 +
direction_to_light_eye_1);
    float dot_prod_specular_1 = max(dot(half_way_eye_1, n_eye), 0.0);
    float specular_factor_1 = pow(dot_prod_specular_1, 20.0);
    vec3 Ls_1 = vec3(0,0,0);
    vec3 Ks_1 = vec3(0,0,0);
    vec3 Is_1 = Ls_1 * Ks_1 * specular_factor_1;

    vec3 La_1 = vec3(0,0,0);
    vec3 Ka_1 = vec3(0,0,0);
    vec3 Ia_1 = La_1 * Ka_1;

    final_color += (Is_1 + Id_1 + Ia_1);
}
```

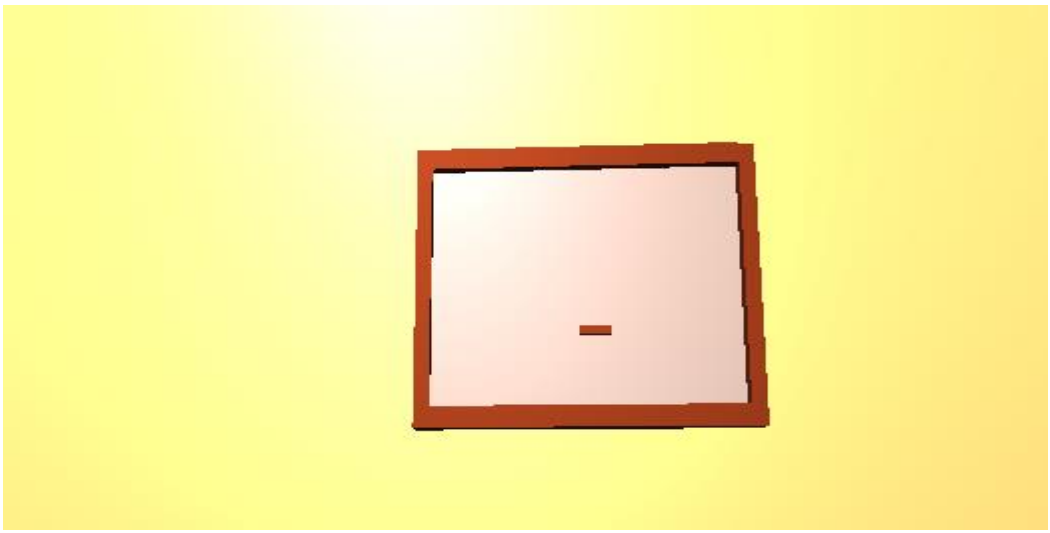
CHƯƠNG III. KẾT QUẢ ĐẠT ĐƯỢC

3.1. Mô hình cánh cửa chính



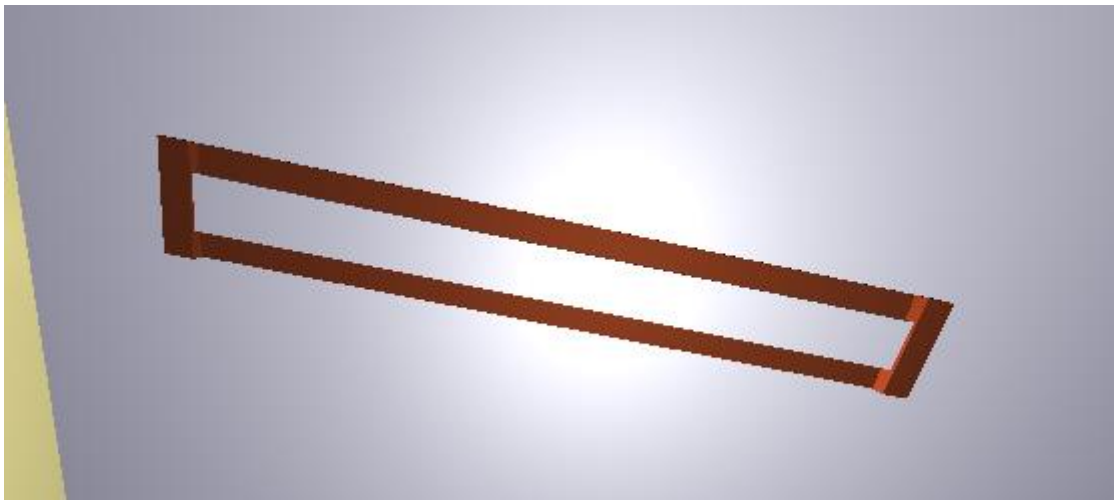
Hình 3. 1: Cánh cửa chính

3.2. Mô hình cửa sổ



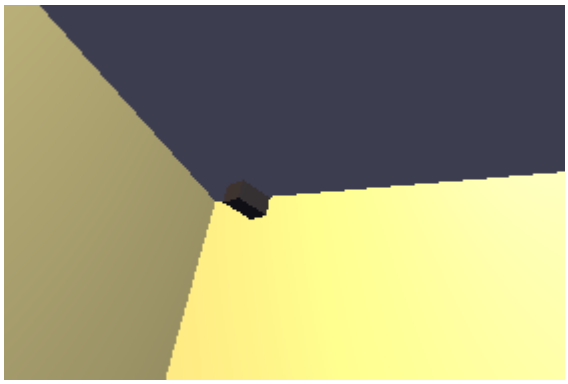
Hình 3. 2: Cửa sổ phòng làm việc

3.3. Mô hình bóng đèn



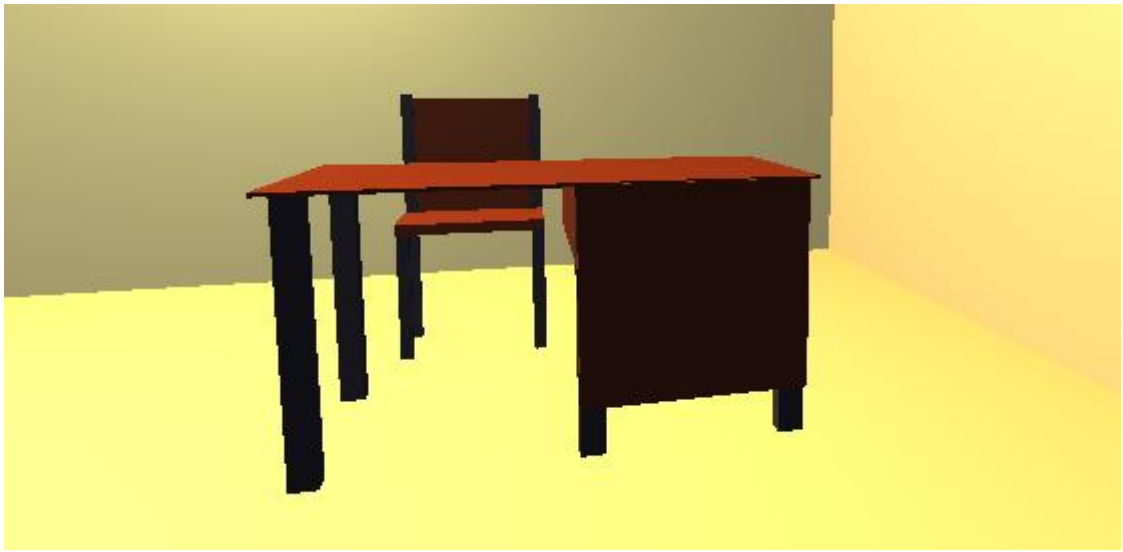
Hình 3. 3: Bóng đèn

3.4. Mô hình camera



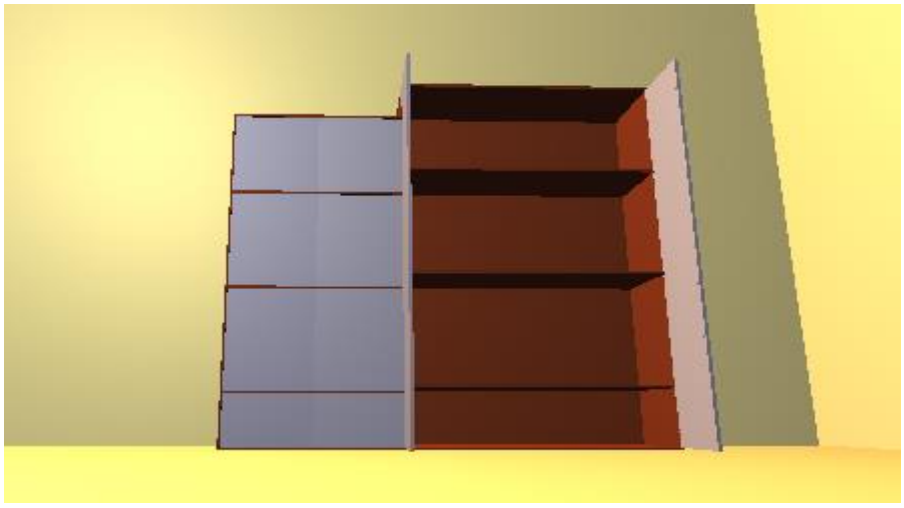
Hình 3. 4: Camera giám sát

3.5. Mô hình bàn ghế làm việc



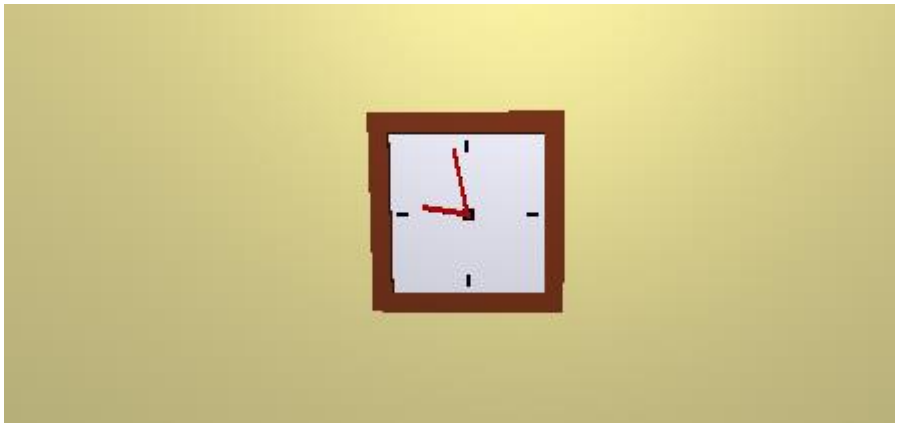
Hình 3. 5: Bàn ghế làm việc

3.6. Mô hình tủ tài liệu



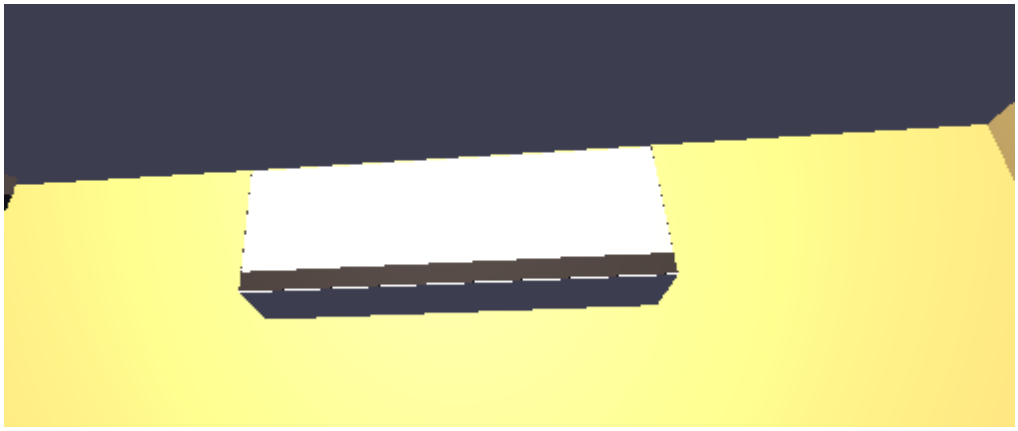
Hình 3. 6: Tủ tài liệu

3.7. Mô hình đồng hồ



Hình 3. 7: Đồng hồ

3.8. Mô hình điều hòa

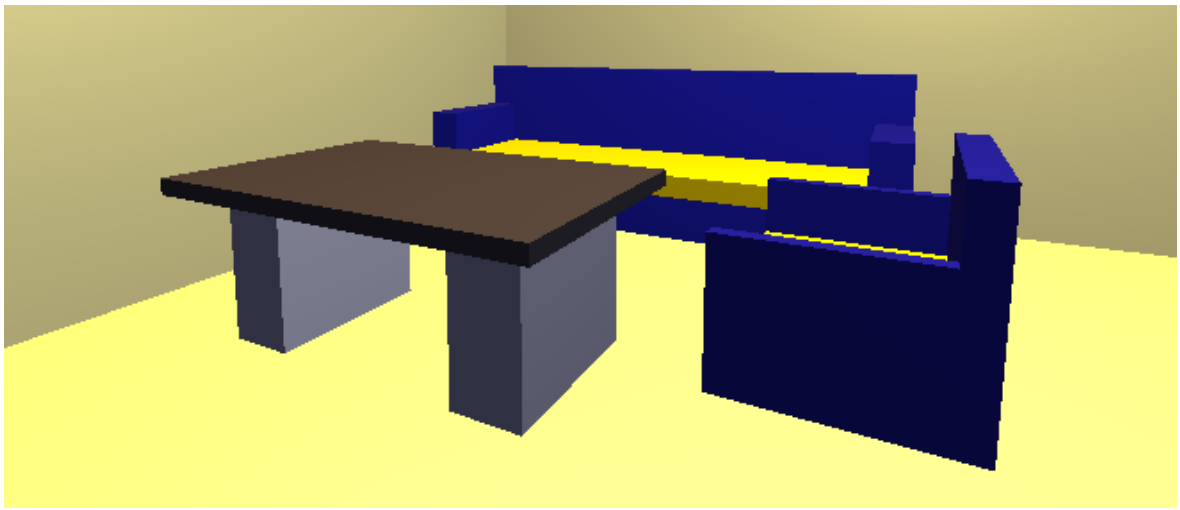


Hình 3. 8: Điều hòa

3.9. Mô hình máy tính xách tay

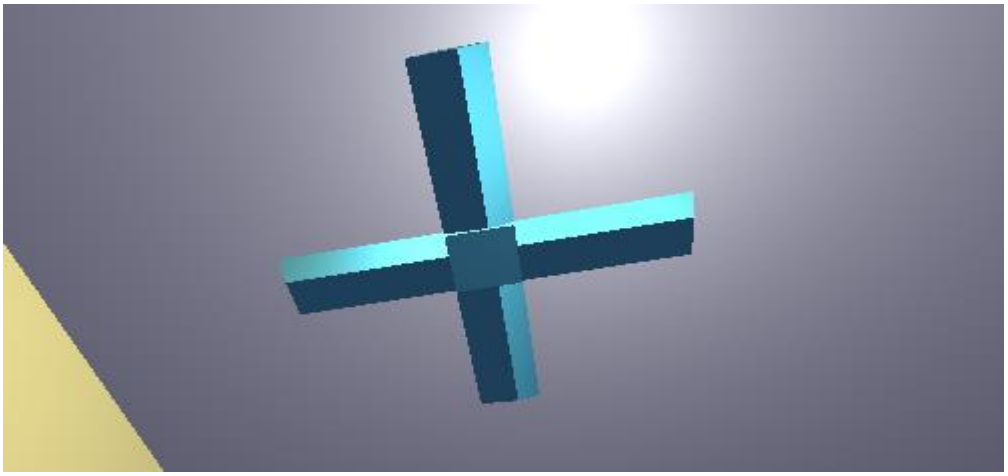
Hình 3. 9: Máy tính xách tay

3.10. Mô hình bàn ghế sofa



Hình 3. 10: Bàn ghế sofa

3.11. Mô hình quạt trần



Hình 3. 11: Quạt trần

KẾT LUẬN

Trong khuôn khổ bài tập lớn môn Đồ họa máy tính, chúng em đã nghiên cứu và hoàn thiện chương trình mô phỏng phòng làm việc sử dụng OpenGL và Visual C++. Dự án đã đạt được các mục tiêu chính đặt ra, bao gồm:

- Xây dựng không gian phòng làm việc 3D với các đối tượng cơ bản như bàn, ghế, máy tính, và các chi tiết nội thất khác, đảm bảo tính trực quan và thẩm mỹ.
- Ứng dụng các kỹ thuật đồ họa như ánh sáng, đổ bóng, kết cấu và các phép biến đổi hình học để tăng tính thực tế và sống động cho mô phỏng.
- Tương tác người dùng: Tích hợp các chức năng như xoay, tịnh tiến, phóng to/thu nhỏ và di chuyển camera, giúp người dùng quan sát không gian từ nhiều góc độ khác nhau.

Quá trình thực hiện đề tài đã giúp nhóm thành viên hiểu rõ hơn về các nguyên lý đồ họa máy tính, kỹ thuật lập trình OpenGL và cách ứng dụng các công cụ hỗ trợ phát triển như Visual C++. Đồng thời, chúng em cũng gặp phải một số khó khăn trong việc tối ưu hóa hiệu suất và quản lý tài nguyên đồ họa, nhưng qua đó tích lũy được nhiều kinh nghiệm quý báu.

Dự án không chỉ đáp ứng yêu cầu về mặt kỹ thuật mà còn là cơ hội để nhóm cải thiện kỹ năng làm việc nhóm, phân công công việc hiệu quả và xử lý các vấn đề phát sinh trong thực tế. Trong tương lai, chương trình có thể được mở rộng thêm các tính năng như: tùy chỉnh nội thất, thêm hiệu ứng thời gian thực, hoặc tích hợp với công nghệ thực tế ảo (VR) để tăng trải nghiệm người dùng.

Chúng em xin chân thành cảm ơn sự hướng dẫn tận tình của thầy Vũ Đức Huy và sự hỗ trợ từ các tài liệu tham khảo, công cụ phát triển trong suốt quá trình thực hiện bài tập lớn này.

Chúng em xin chân thành cảm ơn!

TÀI LIỆU THAM KHẢO

TÀI LIỆU TIẾNG VIỆT

[1] Vũ Minh Yến, Vũ Đức Huy, Nguyễn Phương Nga, Giáo trình Đồ họa máy tính, NXB Khoa học Kỹ thuật, 2015.

[2] Trần Giang Sơn, Đồ họa máy tính, NXB Khoa học Kỹ thuật, 2008.

[3] Vũ Đức Huy, ""Đồ họa máy tính" Đại học điện tử Đại học Công nghiệp Hà Nội," 2024.

TÀI LIỆU TIẾNG ANH

[4] OpenGL Architecture Review Board, Dave Shreiner, Graham Sellers, John Kessenich, Bill Licea-Kane, OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 4.3, Addison-Wesley, 2013.

[5] J. L. McKesson, Learning Modern 3D Graphics Programming, 2011.