| **TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**  **KHOA CÔNG NGHỆ THÔNG TIN**  **---------------------------------------**  Logo.png  **BÁO CÁO THÍ NGHIỆM / THỰC NGHIỆM**  **HỌC PHẦN ĐỒ HỌA MÁY TÍNH**  **ĐỀ TÀI: MÔ PHỎNG PHÒNG BẾP BẰNG OPENGL KHẢ LẬP TRÌNH VÀ VISUAL C++**    **GVHD: TS. Vũ Minh Yến**  **Lớp: 20233IT6010001 Khóa: K16**  **Nhóm: 07**  **Họ và tên: Nguyễn Đức Anh - 2021607023**  **Nguyễn Duy Đại  - 2021608319**  **Trần Văn Duy - 2021603837**  **Chu Quốc Khánh - 2021605406**  **Mã Phương Nga - 2020602285**  **Hà Nội – Năm 2024** |
| --- |

**MỤC LỤC**

[**CHƯƠNG 1. XÁC ĐỊNH VÀ PHÂN TÍCH BÀI TOÁN 5**](#_gjdgxs)

[1.1. Bài toán. 5](#_30j0zll)

[1.2. Mô tả các đối tượng cần thiết kế. 5](#_1fob9te)

[1.3. Mô tả bố cục khung cảnh chung. 9](#_xkfe6fnetj7g)

[1.4. Mô tả kịch bản. 11](#_2et92p0)

[**CHƯƠNG 2: CÀI ĐẶT CHƯƠNG TRÌNH 12**](#_srlp7tefmd86)

[2.1. Kỹ thuật tạo khung phòng bếp 12](#_w1bmqmirbntl)

[2.2. Kỹ thuật tạo hình trụ tròn 14](#_f8gmalpoj0n3)

[2.3. Kỹ thuật tạo cửa ra vào và cửa sổ 16](#_ukycw8miohy6)

[2.4. Kỹ thuật tạo đèn điện, quạt trần, tivi có hoạt ảnh chuyển động 19](#_rk57r3tv6bfz)

[2.5. Kỹ thuật tạo bếp 25](#_ad99jrdg2z57)

[2.6. Kỹ thuật tạo bàn ghế 35](#_dehnqey2k4fp)

[2.7. Kỹ thuật tạo tủ lạnh, đồng hồ và thùng rác 39](#_cwutp7wm8p52)

[2.8. Kỹ thuật di chuyển cam 46](#_9ur8setbeue5)

[2.9. Kỹ thuật chiếu sáng 49](#_kfzti9vdqb0g)

[**CHƯƠNG 3. KẾT QUẢ ĐẠT ĐƯỢC 52**](#_slc4flfzzx2g)

[3.1. Cảnh quan 52](#)

[3.2. Mô hình cửa chính 52](#)

[3.3. Mô hình cửa sổ 54](#)

[3.4. Mô hình bộ bàn ghế 56](#)

[3.5. Mô hình tủ bếp 57](#)

[3.6. Mô hình vòi nước 60](#)

[3.7. Mô hình tủ đựng đồ 61](#)

[3.8. Mô hình cái quạt 61](#)

[3.9. Mô hình tủ lạnh 62](#)

[3.10. Mô hình bóng đèn 66](#)

[3.11. Mô hình thùng rác 67](#)

[3.12. Mô hình tivi 68](#)

[3.13. Mô hình đồng hồ 69](#)

[3.14. Điều khiển camera 69](#)

[**KẾT LUẬN 70**](#)

[**TÀI LIỆU THAM KHẢO 71**](#)

**MỞ ĐẦU**

Đồ họa máy tính là một lĩnh vực của khoa học máy tính nghiên cứu về cơ sở toán học, các thuật toán cũng như các kĩ thuật để cho phép tạo, hiển thị và điều khiển hình ảnh trên màn hình máy tính.

Đồ họa máy tính ngày nay được ứng dụng rất rộng rãi trong nhiều lĩnh vực khoa học, kỹ thuật, nghệ thuật, kinh doanh, quản lý, … Các ứng dụng đồ họa rất đa dạng, phong phú và phát triển liên tục không ngừng. Ngày nay, hầu như không có chương trình ứng dụng nào mà không sử dụng kĩ thuật đồ họa để làm tăng tính hấp dẫn của mình.

Việc mô tả dữ liệu thông qua các hình ảnh và màu sắc đa dạng của nó, các chương trình đồ họa thường thu hút người sử dụng bởi tính thân thiện, dễ dùng,… kích thích khả năng sáng tạo và nâng cao năng suất làm việc.

Là sinh viên khoa Công nghệ thông tin trường Đại học Công Nghiệp Hà Nội. Chúng em có cơ hội được tiếp xúc với môn học Đồ họa máy tính. Với những kiến thức đã được học và được sự hướng dẫn của cô Vũ Minh Yến chúng em đã thực hiện đề tài "Mô phỏng phòng bếp bằng OpenGL khả lập trình và Visual C++". Trong quá trình thực hiện nghiên cứu chúng em đã sử dụng thư viện đồ họa OpenGL đang được sử dụng rộng rãi hiện nay.

Có lẽ rằng chương trình và báo cáo của chúng em chưa được chuyên nghiệp, hoàn chỉnh nhất, còn có những thiếu sót. Vì thế nhóm em rất mong thầy cô và các bạn có thể góp ý để nhóm em xây dựng đề đạt kết quả tốt nhất có thể.

Chúng em xin trân trọng cảm ơn!

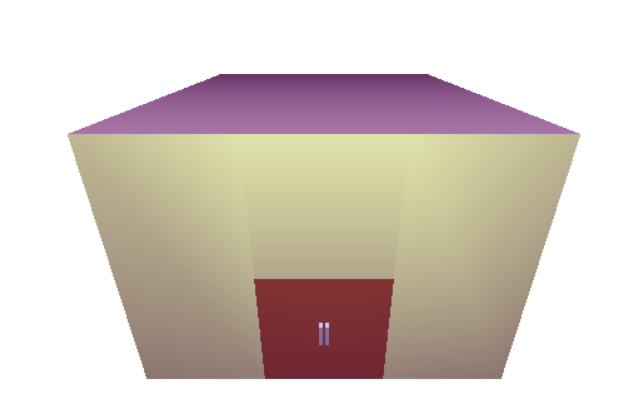
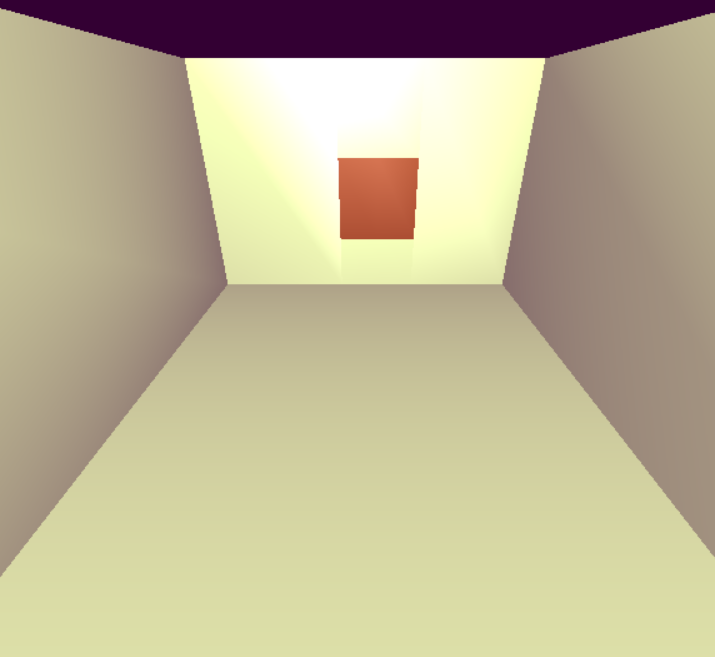
# **CHƯƠNG 1. XÁC ĐỊNH VÀ PHÂN TÍCH BÀI TOÁN**

## **1.1. Bài toán.**

* Tên đề tài: Mô phỏng phòng bếp bằng OpenGL khả lập trình và Visual C++.
* Phòng bếp là nơi rất gần gũi với mọi người, trong phòng bếp thì có các vật dụng cần thiết như, tủ lạnh, bàn ghế, tủ bát, bếp gas…Với đề tài trên nhóm chúng em sẽ mô phỏng một căn phòng bếp với những đồ dùng và thiết bị cần có bên trong.

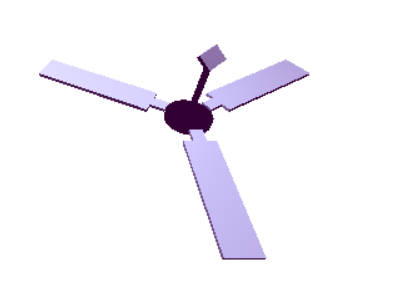
## **1.2. Mô tả các đối tượng cần thiết kế.**

* **Khung căn phòng :**
* Mô hình gồm 9 tường (tường trái , tường phải, 3 tường trước, 4 tường sau), 1 trần nhà, 1 sàn nhà.
* Cửa: Cửa ra vào có tay nắm, cửa sổ có thể mở/đóng.

*Hình 1.1. Mô hình căn phòng.*

* **Quạt trần :**
* Mô hình gồm 3 cánh quạt, 1 trụ quay.
* Có khả năng di chuyển cánh quạt quay 360 độ, bật/ tắt quạt.



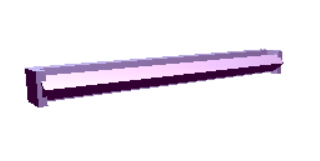
*Hình 1.2. Mô hình quạt trần.*

* **Tivi :**
* Gồm thanh đỡ và màn hình.
* Có thể bật/tắt . Khi bật trên màn hình xuất hiện hình mặt trời và một ô tô.

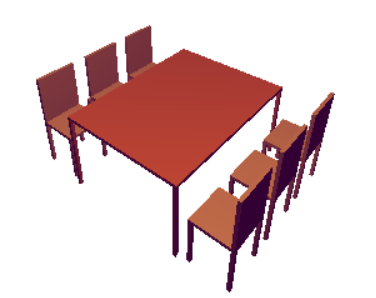
*Hình 1.3. Mô hình tivi khi tắt, bật.*

* **Bóng đèn :** Gồm bóng đèn hình trụ, các thanh đỡ.



*Hình 1.4. Mô hình đèn điện*.

* **Bộ bàn ghế :**
* Gồm 1 bàn, 6 ghế.
* Có thể kéo ra, kéo vào các hàng ghế.



*Hình 1.5. Mô hình bộ bàn ăn.*

* **Tủ lạnh :**
* Gồm có 2 ngăn : ngăn đông và ngăn mát. Có cửa có thể đóng/mở.
* Ngăn kéo ở tủ dưới có thể kéo ra/vào.



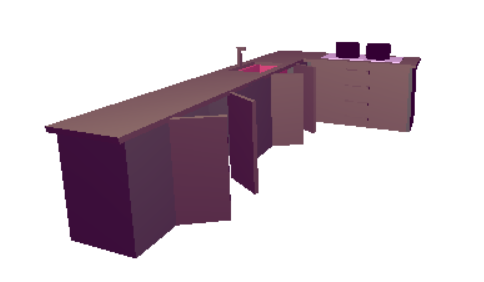
*Hình 1.5. Mô hình tủ lạnh.*

* **Đồng hồ :**
* Gồm: Kim giờ, kim phút, kim giây và mặt đồng hồ.
* 3 kim giờ, phút, giây có thể di chuyển với tốc độ tương đương đồng hồ thật.



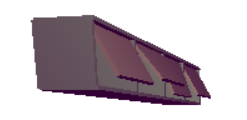
*Hình 1.6. Mô hình đồng hồ.*

* **Tủ bếp dưới :**
* Gồm 1 khung tủ có 2 bộ cửa, mặt trên có bồn rửa và vòi nước.
* Cửa có thể đóng/mở, vòi nước có thể xoay.
* Nếu cửa tủ bếp điện đang mở, thì sẽ không mở được cửa tủ bếp dưới.



*Hình 1.6. Mô hình tủ bếp dưới, tủ bếp điện.*

* **Tủ bếp điện :**
* Gồm khung tủ với 4 hộc tủ, mặt trên có mặt bếp từ và 2 xoong.
* Bốn hộc tủ kèm tay nắm có thể kéo ra , kéo vào.
* **Tủ bếp trên :**
* Gồm khung tủ, 4 ngăn tủ.
* Các cánh cửa tủ có thể đóng/mở từng cái một.



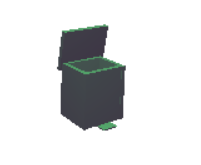
*Hình 1.7. Mô hình tủ bếp trên.*

* **Hút mùi :**
* Gồm một ống khí và mặt hút mùi.



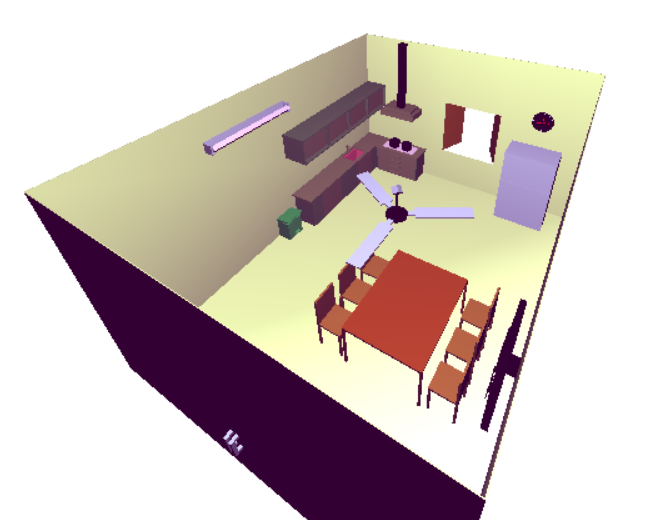
*Hình 1.8. Mô hình hút mùi.*

* **Thùng rác :**
* Thùng rác có nắp và chân đạp.
* Khi chân đạp di chuyển thì đồng thời nắp thùng rác mở ra/ đóng lại.

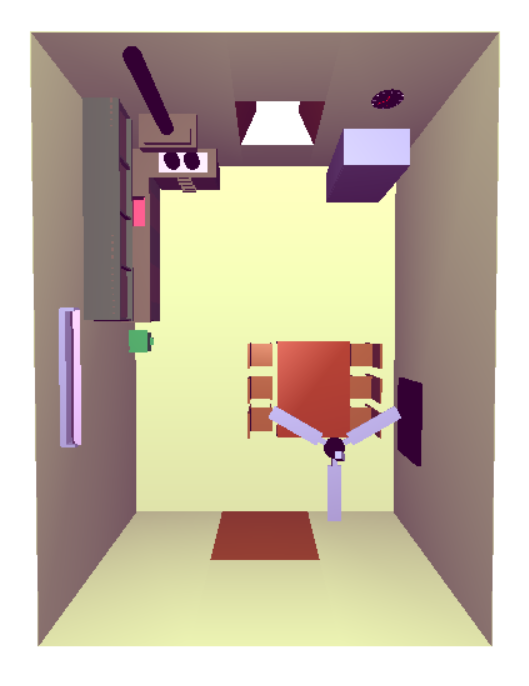


*Hình 1.9. Mô hình thùng rác.*

## **1.3. Mô tả bố cục khung cảnh chung.**

****

*Hình 1.10. Khung cảnh phòng bếp*.



*Hình 1.11. Khung cảnh chiếu từ trên xuống.*



*Hình 1.12. Khung cảnh chiếu bằng.*

## **1.4. Mô tả kịch bản.**

Chương trình mô phỏng góc nhìn của người vào phòng bếp.

Dưới đây là mô tả chi tiết của kịch bản:

* Chạy chương trình : Người dùng khởi động ứng dụng và màn hình hiển thị màn hình đứng trước cửa của phòng bếp.
* Thiết lập gian phòng bếp: Trong phòng gồm nhiều đối tượng: bàn ghế, tủ bếp, tủ lạnh, quạt trần, tivi, đồng hồ , thùng rác, hút mùi. Mỗi đối tượng được mô hình hóa và tạo trong môi trường 3D sử dụng các hình học và đồ họa của OpenGL.
* Tương tác với các đối tượng : Người dùng có thể tương tác với quạt trận(bật/ tắt quạt), kéo ghế, mở cửa sổ, cửa tủ, đóng/ mở nắp thùng rác, bật/ tắt tivi...

# 

# 

# 

# **CHƯƠNG 2: CÀI ĐẶT CHƯƠNG TRÌNH**

## **2.1. Kỹ thuật tạo khung phòng bếp**

mat4 model\_cuaChinh, model\_cuaSo, model\_quat;

GLfloat goccua, gocso = 65, quayQuat;

void sanNha() {

mat4 sannha = Scale(6, 0.01, 8);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* sannha);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void tranNha() {

mat4 sannha = Translate(0, 4, 0) \* Scale(6, 0.01, 8);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* sannha);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void tuongTrai() {

mat4 tuongtrai = Translate(-3, 2, 0) \* Scale(0.01, 4, 8);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* tuongtrai);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void tuongPhai() {

mat4 tuongphai = Translate(3, 2, 0) \* Scale(0.01, 4, 8);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* tuongphai);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void tuongTruoc1() {

mat4 tuongtruoc = Translate(-2, 2, 4) \* Scale(2, 4, 0.01);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* tuongtruoc);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void tuongTruoc2() {

mat4 tuongtruoc = Translate(2, 2, 4) \* Scale(2, 4, 0.01);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* tuongtruoc);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void tuongTruoc3() {

mat4 tuongtruoc = Translate(0, 3, 4) \* Scale(2, 2, 0.01);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* tuongtruoc);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void tuongSau1() {

mat4 tuongsau = Translate(-1.75, 2, -4) \* Scale(2.5, 4, 0.01);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* tuongsau);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void tuongSau2() {

mat4 tuongsau = Translate(2, 2, -4) \* Scale(2, 4, 0.01);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* tuongsau);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void tuongSau3() {

mat4 tuongsau = Translate(0.25, 0.5, -4) \* Scale(1.5, 1, 0.01);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* tuongsau);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void tuongSau4() {

mat4 tuongsau = Translate(0.25, 3.25, -4) \* Scale(1.5, 1.5, 0.01);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* tuongsau);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void khungPhong() {

//mau(225, 239, 148);

mau(225, 296, 155);

sanNha();

tuongTrai();

tuongPhai();

tuongTruoc1();

tuongTruoc2();

tuongTruoc3();

tuongSau1();

tuongSau2();

tuongSau3();

tuongSau4();

mau(250, 250, 250);

tranNha();

}

## **2.2. Kỹ thuật tạo hình trụ tròn**

void drawCylinder(float radius, float height, int numSegments) {

float segmentAngle = 2.0f \* M\_PI / numSegments;

// Vẽ thành hình trụ

glBegin(GL\_TRIANGLE\_STRIP);

for (int i = 0; i <= numSegments; i++) {

float angle = i \* segmentAngle;

// Tính toán các tọa độ x, y của các điểm trên đường viền hình float x = radius \* cos(angle);

float y = radius \* sin(angle);

// Vẽ đỉnh dưới của hình trụ

glVertex3f(x, y, 0.0f);

// Vẽ đỉnh trên của hình trụ

glVertex3f(x, y, height);

}

glEnd();

// Vẽ đáy dưới của hình trụ

glBegin(GL\_TRIANGLE\_FAN);

glVertex3f(0.0f, 0.0f, 0.0f);

for (int i = 0; i <= numSegments; i++) {

float angle = i \* segmentAngle;

float x = radius \* cos(angle);

float y = radius \* sin(angle);

glVertex3f(x, y, 0.0f);

}

glEnd();

// Vẽ đáy trên của hình trụ

glBegin(GL\_TRIANGLE\_FAN);

glVertex3f(0.0f, 0.0f, height);

for (int i = 0; i <= numSegments; i++) {

float angle = i \* segmentAngle;

float x = radius \* cos(angle);

float y = radius \* sin(angle);

glVertex3f(x, y, height);

}

glEnd();

}

void hinhtru() {

mat4 instance = Scale(0.5, 0.5, 0.5) \* Translate(-1, -1, 0);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* instance);

drawCylinder(1.0f, 1.0f, 100); // Vẽ hình trụ tròn

}

## **2.3. Kỹ thuật tạo cửa ra vào và cửa sổ**

void cuaTrai()

{

mat4 cuatrai = Translate(0.5, 0, 0) \* Scale(1, 2, 0.01);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_cuaChinh \* cuatrai);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void cuaPhai()

{

mat4 cuaphai = Translate(-0.5, 0, 0)\* Scale(1, 2, 0.01);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE,model\_cuaChinh \* cuaphai);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void cuaSoTrai()

{

mat4 cuasotrai = Translate(-0.75, 0, 0) \* RotateY(gocso) \* Translate(0.375, 0, 0) \* Scale(0.75, 1.5, 0.01);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_cuaSo \* cuasotrai);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void cuaSoPhai()

{

mat4 cuasophai = Translate(0.75, 0, 0) \* RotateY(-gocso) \* Translate(-0.375, 0, 0) \* Scale(0.75, 1.5, 0.01);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_cuaSo \* cuasophai);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void tayNamCua1(GLfloat traiPhai)

{

mat4 cuatrai = Translate(0.95 \* traiPhai, 0.2, 0.05) \* Scale(0.05, 0.05, 0.1);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_cuaChinh \* cuatrai);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void tayNamCua2(GLfloat traiPhai)

{

mat4 cuatrai = Translate(0.95 \* traiPhai, -0.1, 0.05) \* Scale(0.05, 0.05, 0.1);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_cuaChinh \* cuatrai);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void tayNamCua3(GLfloat traiPhai)

{

mat4 cuatrai = Translate(0.95 \* traiPhai, 0.05, 0.1) \* Scale(0.05, 0.35, 0.05);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_cuaChinh \* cuatrai);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void cuaPhong()

{

mau(139, 90,0);

model\_cuaChinh = model \*Translate(-1, 1, 4) \* RotateY(goccua);

cuaTrai();

mau(176, 224, 230);

tayNamCua1(1);

tayNamCua2(1);

tayNamCua3(1);

mau(139, 90, 0);

model\_cuaChinh = model \* Translate(1, 1, 4) \* RotateY(-goccua);

cuaPhai();

mau(176, 224, 230);

tayNamCua1(-1);

tayNamCua2(-1);

tayNamCua3(-1);

mau(139, 90, 0);

model\_cuaSo = model \* Translate(0.25, 1.75, -4);

cuaSoTrai();

cuaSoPhai();

}

## **2.4. Kỹ thuật tạo đèn điện, quạt trần, tivi có hoạt ảnh chuyển động**

// bóng đèn

void thanhDen()

{

mat4 thanhden = Translate(-2.95, 3.3, 0.5) \* Scale(0.1, 0.2, 2);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* thanhden);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void thanhDenTrai()

{

mat4 thanhden = Translate(-2.875, 3.3, 1.5) \* Scale(0.2, 0.2, 0.05);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* thanhden);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void thanhDenPhai()

{

mat4 thanhden = Translate(-2.875, 3.3, -0.5) \* Scale(0.2, 0.2, 0.05);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* thanhden);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void bongDen() {

mat4 instance = Translate(-2.8, 3.3, 0.5) \* RotateZ(45) \* Scale(0.1, 0.1, 2);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* instance);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

bool batDen = true;

void denDien()

{

//mau(255, 255, 255);

mau(176, 224, 230);

thanhDen();

thanhDenTrai();

thanhDenPhai();

if (batDen)

mau(220, 230, 240);

else

mau(110, 120, 130);

bongDen();

}

//quat tran

void canh(float x)

{

mat4 instance = RotateY(x) \* Translate(0, 0, 0.65) \* Scale(0.2, 0.02, 0.8);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_quat \* instance);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void bocanh()

{

canh(0);

canh(120);

canh(240);

}

void canhNho(float x)

{

mat4 instance = RotateY(x) \* Translate(0, 0, 0.2) \* Scale(0.08, 0.02, 0.1);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_quat \* instance);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void bocanhNho()

{

canhNho(0);

canhNho(120);

canhNho(240);

}

GLboolean checked = false;

void canh\_quay()

{

if (checked) {

quayQuat += 25;

glutPostRedisplay();

}

}

void trucQuay() {

mat4 instance = Translate(0,0, 0) \* RotateX(90);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_quat \* instance);

drawCylinder(0.15, 0.04, 100); // Vẽ hình trụ tròn

}

void trucOng() {

mat4 instance = Translate(0, 0.4, 0) \* RotateX(90);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_quat \* instance);

drawCylinder(0.02, 0.4, 100); // Vẽ hình trụ tròn

}

void trucTren()

{

mat4 instance = Translate(0, 0.4, 0) \* RotateY(45) \* RotateX(35) \* RotateZ(45) \* Scale(0.1, 0.1, 0.1);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_quat \* instance);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void quatTran() {

//mau(250, 250, 250);

mau(176, 224, 230);

model\_quat = model \* Translate(1, 3.5, 1.5);

trucQuay();

trucOng();

trucTren();

model\_quat \*= RotateY(quayQuat);

bocanhNho();

bocanh();

canh\_quay();

}

// Tivi và hoạt ảnh

void thanhTivi()

{

mat4 thanh = Translate(2.9, 1.5, 1.5) \* Scale(0.2, 0.2, 0.2);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* thanh);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void tivi() {

mat4 thanh = Translate(2.8, 1.5, 1.5) \* Scale(0.1, 1, 1.5);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* thanh);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

GLfloat xee;

mat4 xe;

bool batTV = false;

void banhXe(GLfloat y, GLfloat z)

{

mat4 instance = Translate(2.72, y, z) \* RotateY(90);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, xe \* instance);

drawCylinder(0.1, 0.02, 100); // Vẽ hình trụ tròn

}

void mattroi()

{

mat4 instance = Translate(2.72, 1.86, 2.1) \* RotateY(90);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* instance);

drawCylinder(0.1, 0.02, 100); // Vẽ hình trụ tròn

}

void tiaMatTroi(GLfloat y, GLfloat z, GLfloat a)

{

mat4 thanh = Translate(2.72, y, z) \* RotateX(a) \* Scale(0.015, 0.015, 0.15);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* thanh);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void thanXe()

{

mat4 thanh = Translate(2.72, 1.45, 1.3) \* Scale(0.02, 0.15, 0.7);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, xe \* thanh);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void thanXe2()

{

mat4 thanh = Translate(2.72, 1.55, 1.3) \* Scale(0.02, 0.12, 0.5);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, xe \* thanh);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void trangTri()

{

if (batTV)mau(0, 0, 255);

else

{

mau(0, 0, 0);

xee = 0;

}

thanhTivi();

tivi();

mau(0, 0, 0);

xe = model \* Translate(0, 0, xee);

banhXe(1.3, 1.1);

banhXe(1.3, 1.5);

thanXe();

thanXe2();

if (batTV && xee < 0.6)

{

xee += 0.0005;

glutPostRedisplay();

}

mattroi();

tiaMatTroi(1.86, 1.94, 0);

tiaMatTroi(1.92, 1.98, 40);

tiaMatTroi(1.8, 1.98, -40);

tiaMatTroi(1.75, 2.05, -60);

tiaMatTroi(1.72, 2.12, -90);

tiaMatTroi(1.75, 2.18, -120);

}

## **2.5. Kỹ thuật tạo bếp**

mat4 canhtu\_m;

mat4 bep\_m;

mat4 tubep\_m;

float tubepxyz[4] = { 0,0,0,0 };

void canhturuabat(GLfloat x, GLfloat y, GLfloat z, GLfloat lr, GLfloat sx, GLfloat sy, GLfloat sz) {

mat4 temp = Scale(sx,sy,sz) \* Translate(0, 0, lr);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* canhtu\_m \* temp);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void matbep() {

mat4 temp = Scale(1.4, 0.04, 0.6);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* temp);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void khungtubep() {

// thanh phai

mat4 temp = Translate(0.64, -0.42, 0) \* Scale(0.04, 0.8, 0.52);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* temp);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

//thanh trai

temp = Translate(-0.64, -0.42, 0) \* Scale(0.04, 0.8, 0.52);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* temp);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

// thanh sau

temp = Translate(0, -0.42, 0.24) \* Scale(1.24, 0.8, 0.04);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* temp);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void beptu() {

mat4 temp = Translate(0, 0.01, 0) \* Scale(1, 0.04, 0.4);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* temp);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void taytu() {

mat4 temp = Translate(0, 0.098, -0.32) \* Scale(0.3, 0.04, 0.04);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* tubep\_m \* temp);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

temp = Translate(-0.13, 0.098, -0.28) \* Scale(0.04, 0.04, 0.04);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* tubep\_m \* temp);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

temp = Translate(0.13, 0.098, -0.28) \* Scale(0.04, 0.04, 0.04);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* tubep\_m \* temp);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void xoong(GLfloat x, GLfloat y, GLfloat z) {

mat4 instance = Translate(x, y + 0.25 - 0.04, z) \* RotateX(90) \* Scale(0.5, 0.5, 0.5);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* instance);

drawCylinder(0.3f, 0.4f, 100); // Vẽ hình trụ tròn

}

//void cacloaihop(GLfloat x, GLfloat y, GLfloat z) {

// mat4 instance = Translate(x, y , z) \* RotateX(90) \* Scale(0.5, 0.5, 0.5);

// glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* instance);

// drawCylinder(0.2f, 0.3f, 100); // Vẽ hình trụ tròn

//}

void tubep() {

// mat duoi

mat4 temp = Scale(1.24, 0.04, 0.52);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* tubep\_m \* temp);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

// mat trai

temp = Translate(-0.6, 0.196 / 2, 0) \* Scale(0.04, 0.196, 0.52);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* tubep\_m \* temp);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

// mat phai

temp = Translate(0.6, +0.196 / 2, 0) \* Scale(0.04, 0.196, 0.52);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* tubep\_m \* temp);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

// mat truoc

temp = Translate(0, +0.196 / 2, 0.24) \* Scale(1.24, 0.196, 0.04);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* tubep\_m \* temp);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

// mat sau

temp = Translate(0, +0.196 / 2, -0.24) \* Scale(1.24, 0.196, 0.04);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* tubep\_m \* temp);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

taytu();

}

void botubep() {

tubep\_m = Translate(0, -0.2, 0) \* Translate(0, 0, tubepxyz[0]);

tubep();

tubep\_m = Translate(0, -0.4, 0) \* Translate(0, 0, tubepxyz[1]);

tubep();

tubep\_m = Translate(0, -0.6, 0) \* Translate(0, 0, tubepxyz[2]);

tubep();

tubep\_m = Translate(0, -0.8, 0) \* Translate(0, 0, tubepxyz[3]);

tubep();

}

void mattudungdo() {

mat4 temp = Translate(1, 0, -0.8 - 2.3 + 0.6) \* Scale(0.6, 0.04, 1.6);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* temp);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void khungtudungdo() {

mat4 temp = Translate(1 + 0.24, -0.4, -0.8 - 2.3 + 0.6 + 0.04) \* Scale(0.04, 0.8, 1.6);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* temp);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

temp = Translate(1 + 0.24 - 0.52, -0.4, -0.8 - 2.3 + 0.04) \* Scale(0.04, 0.8, 0.4);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* temp);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

temp = Translate(1 + 0.24 - 0.52, -0.4, -0.8 - 2.3 + 0.04 + 0.4 + 0.72 + 0.08) \* Scale(0.04, 0.8, 0.4);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* temp);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

temp = Translate(0.24 + 0.7 + 0.04, -0.4, 0 - 3.7 + 0.6 - 0.1 - 0.04) \* Scale(0.48, 0.8, 0.04);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* temp);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

int alpha = 0;

void hecanhtu() {

canhtu\_m = Translate(1 + 0.24 - 0.52, -0.4, -0.8 - 2.3 + 0.04 + 0.4 + 0.72 + 0.08 - 0.2) \* RotateY(alpha);

canhturuabat(0, 0, 0, -0.5, 0.04, 0.8, 0.4);

canhtu\_m = Translate(1 + 0.24 - 0.52, -0.4, -0.8 - 2.3 + 0.04 + 0.4 + 0.72 + 0.08 - 0.2 - 0.72 - 0.08) \* RotateY(-alpha);

canhturuabat(0, 0, 0, 0.5, 0.04, 0.8, 0.4);

}

mat4 bonruabat\_m;

mat4 canhtubonruabat\_m;

mat4 voirua\_m;

int voirua\_q = 0;

void voirua\_thanh1() {

mat4 temp = Scale(0.04, 0.2, 0.04) \* Translate(-0.5, 0.5, 0);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* voirua\_m \* temp);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void voirua\_thanh2() {

mat4 temp = Translate(-0.05, 0.22, 0) \* Scale(0.1, 0.04, 0.04);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* voirua\_m \* temp);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void voirua(GLfloat x, GLfloat y, GLfloat z) {

voirua\_m = Translate(1.25, 0, -1) ;

voirua\_thanh1();

voirua\_m = voirua\_m \* RotateY(voirua\_q);

voirua\_thanh2();

}

void bonruabat(GLfloat x, GLfloat y, GLfloat z) {

// mat duoi

mat4 temp = Translate(1.2 + x, 0 + y, 0 + z) \* Scale(0.4, 0.04, 0.8);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* temp);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

//mat trai

temp = Translate(1.2 + x, 0.2 + y, -0.38 + z) \* Scale(0.4, 0.36, 0.04);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* temp);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

// mat phai

temp = Translate(1.2 + x, 0.2 + y, 0.38 + z) \* Scale(0.4, 0.36, 0.04);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* temp);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

temp = Translate(1.2 + 0.18 + x, 0.2 + y, 0 + z) \* Scale(0.04, 0.36, 0.72);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* temp);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

temp = Translate(1.2 - 0.18 + x, 0.2 + y, 0 + z) \* Scale(0.04, 0.36, 0.72);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* temp);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void matkhuruabat(GLfloat x, GLfloat y, GLfloat z) {

mat4 temp = Translate(1 + x, 0 + y, -0.1 + 0.5 + z) \* Scale(0.6, 0.04, 1);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* temp);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

temp = Translate(1 - 0.25 + x, 0 + y, -0.1 + 0.5 - 0.6 + z) \* Scale(0.1, 0.04, 1.2);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* temp);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

temp = Translate(1 + 0.25 + x, 0 + y, -0.1 + 0.5 - 0.6 + z) \* Scale(0.1, 0.04, 1.2);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* temp);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

temp = Translate(1 + x, 0 + y, -0.1 - 0.8 + z) \* Scale(0.6, 0.04, 0.4);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* temp);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void mattubat(GLfloat x, GLfloat y, GLfloat z) {

mat4 temp = Translate(1 + x, 0 + y, -0.1 + 0.5 + z) \* Scale(0.6, 0.04, 3);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* temp);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void hecanhturuabat() {

canhtu\_m = Translate(0.68 + 0.56 - 0.52, -0.4, 0.18 - 0.12 - 0.8 + 0.18 + 0.02) \* RotateY(alpha);

canhturuabat(0, 0, 0, -0.5, 0.04, 0.8, 0.36);

/\* canh trai\*/

canhtu\_m = Translate(0.68 + 0.56 - 0.52, -0.4, -0.54 - 0.12 - 0.8 + 0.18 + 0.02) \* RotateY(-alpha);

canhturuabat(0, 0, -0.18, 0.5, 0.04, 0.8, 0.36);

}

void khuturuabat() {

mat4 temp = Translate(0.68 + 0.56, -0.4, -1.92 / 2 + 0.3 - 0.04) \* Scale(0.04, 0.8, 1.92);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* temp);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

temp = Translate(0.68 + 0.56 - 0.52, -0.4, -0.8 / 2 + 0.3 - 0.04) \* Scale(0.04, 0.8, 0.8);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* temp);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

temp = Translate(0.68 + 0.56 - 0.52, -0.4, -1.52 - 0.4 / 2 + 0.3 - 0.04) \* Scale(0.04, 0.8, 0.4);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* temp);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

temp = Translate(0.96, -0.4, 0.24) \* Scale(0.6, 0.8, 0.04);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* temp);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

temp = Translate(0.98, -0.4, -1.28 - 0.02 - 0.3 - 0.04) \* Scale(0.48, 0.8, 0.04);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model \* bep\_m \* temp);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void khunaunuong() {

mau(100, 120, 60);

matbep();

mau(70, 120, 60);

khungtubep();

mau(100, 120, 60);

botubep();

mau(250,250,250);

beptu();

xoong(0.2, 0, 0);

xoong(-0.2, 0, 0);

//cacloaihop(x, y, z);

}

void tudungdo() {

mau(100, 120, 60);

mattudungdo();

mau(70, 120, 60);

khungtudungdo();

mau(100, 120, 60);

hecanhtu();

}

void khuruabat() {

mau(100, 120, 60);

matkhuruabat(0, 0, -0.6);

mau(230, 100, 100);

bonruabat(-0.2, -0.4, -1);

mau(70, 120, 60);

khuturuabat();

mau(100, 120, 60);

hecanhturuabat();

voirua(0, 0, 0);

}

void bep() {

bep\_m = Translate(-1.7,0.85,-3.7) \* RotateY(180);

khunaunuong();

//mau(200, 100, 10);

khuruabat();

tudungdo();

}

## **2.6. Kỹ thuật tạo bàn ghế**

mat4 model\_ban, model\_ghe;

GLfloat keoGhePhai, keoGheTrai;

void matBan()

{

mat4 matban = Scale(1.5, 0.04, 2);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_ban \* matban);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void chanBan(GLfloat n, GLfloat m)

{

mat4 matban = Translate(n, -0.5, m) \* Scale(0.04, 0.96, 0.04);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_ban \* matban);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void matGhe(GLfloat b, GLfloat n, GLfloat m)

{

mat4 matghe = Translate(b, n, m) \* Scale(0.5, 0.04, 0.5);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_ghe \* matghe);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void lungGhe(GLfloat b, GLfloat n, GLfloat m)

{

mat4 matghe = Translate(b, n, m) \* Scale(0.04, 0.6, 0.5);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_ghe \* matghe);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void chanGhe(GLfloat b, GLfloat n, GLfloat m)

{

mat4 matghe = Translate(b, n, m) \* Scale(0.04, 0.58, 0.04);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_ghe \* matghe);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void boBanGhe()

{

//mau(139, 129 ,76);

mau(139 ,69 ,0);

model\_ban = model \* Translate(1, 1, 1);

matBan();

chanBan(0.73, 0.98);

chanBan(-0.73, 0.98);

chanBan(0.73, -0.98);

chanBan(-0.73, -0.98);

mau(139 ,105 ,20);

model\_ghe = model\_ban \* Translate(keoGhePhai, 0, 0);

matGhe(1.1, -0.4, 0.7);

lungGhe(1.33, -0.08, 0.7);

chanGhe(0.87, -0.7, 0.93);

chanGhe(1.33, -0.7, 0.93);

chanGhe(0.87, -0.7, 0.47);

chanGhe(1.33, -0.7, 0.47);

matGhe(1.1, -0.4, 0);

lungGhe(1.33, -0.08, 0);

chanGhe(0.87, -0.7, 0.23);

chanGhe(1.33, -0.7, 0.23);

chanGhe(0.87, -0.7, -0.23);

chanGhe(1.33, -0.7, -0.23);

matGhe(1.1, -0.4, -0.7);

lungGhe(1.33, -0.08, -0.7);

chanGhe(0.87, -0.7, -0.47);

chanGhe(1.33, -0.7, -0.47);

chanGhe(0.87, -0.7, -0.93);

chanGhe(1.33, -0.7, -0.93);

model\_ghe = model\_ban \* Translate(keoGheTrai, 0, 0);

matGhe(-1.1, -0.4, 0.7);

lungGhe(-1.33, -0.08, 0.7);

chanGhe(-0.87, -0.7, 0.93);

chanGhe(-1.33, -0.7, 0.93);

chanGhe(-0.87, -0.7, 0.47);

chanGhe(-1.33, -0.7, 0.47);

matGhe(-1.1, -0.4, 0);

lungGhe(-1.33, -0.08, 0);

chanGhe(-0.87, -0.7, 0.23);

chanGhe(-1.33, -0.7, 0.23);

chanGhe(-0.87, -0.7, 0.27);

chanGhe(-1.33, -0.7, 0.27);

matGhe(-1.1, -0.4, -0.7);

lungGhe(-1.33, -0.08, -0.7);

chanGhe(-0.87, -0.7, -0.47);

chanGhe(-1.33, -0.7, -0.47);

chanGhe(-0.87, -0.7, -0.93);

chanGhe(-1.33, -0.7, -0.93);

}

## **2.7. Kỹ thuật tạo tủ lạnh, đồng hồ và thùng rác**

**//**Tủ lạnh

mat4 model\_tu, model\_hocTu, model\_canhLanh;

GLfloat keoLanh, canhDuoi, canhTren;

void lanhSau()

{

mat4 canh = Scale(1.2, 2, 0.04);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_tu \* canh);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void lanhGiua(GLfloat a, GLfloat b)

{

mat4 canh = Translate(0, a, 0.32) \* Scale(1.12, b, 0.6);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_tu \* canh);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void lanhCanh(GLfloat a)

{

mat4 canh = Translate(a, 0, 0.32) \* Scale(0.04, 2, 0.6);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_tu \* canh);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void hocSau(GLfloat a)

{

mat4 canh = Translate(0, -0.8, a) \* Scale(1.12, 0.32, 0.02);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_hocTu \* canh);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void hocDuoi()

{

mat4 canh = Translate(0, -0.95, 0.3) \* Scale(1.12, 0.02, 0.52);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_hocTu \* canh);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void hocCanh(GLfloat a)

{

mat4 canh = Translate(a, -0.8, 0.3) \* Scale(0.02, 0.32, 0.5);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_hocTu \* canh);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void hocThanh()

{

mat4 canh = Translate(0, -0.65, 0.58) \* Scale(1.12, 0.04, 0.04);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_hocTu \* canh);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void hoc()

{

model\_hocTu = model\_tu \* Translate(0, 0, keoLanh);

hocDuoi();

hocSau(0.05);

hocSau(0.55);

hocCanh(0.55);

hocCanh(-0.55);

hocThanh();

}

void canhLanhDuoi()

{

mat4 cua = Translate(1.18, 0, 0) \* RotateY(canhDuoi) \* Translate(-0.59, 0, 0) \* Scale(1.18, 1.18, 0.04);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_canhLanh \* cua);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void canhLanhTren()

{

mat4 cua = Translate(1.18, 0.5, 0) \* RotateY(canhTren) \* Translate(-0.59, 0.5, 0) \* Scale(1.18, 0.8, 0.04);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_canhLanh \* cua);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void khayDa()

{

mat4 canh = Translate(1.18, 0.5, 0) \* Scale(1.2, 2, 0.04);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_tu \* canh);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

//Dong ho

mat4 model\_dongho, model\_donghog, model\_donghop, model\_donghoh;

GLfloat giay,phut=-60,gio=90;

void quayDongHo()

{

giay -= 0.1;

phut -= 0.1/60;

gio -= 0.1 / (60 \* 60);

glutPostRedisplay();

}

void DongHo() {

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_dongho );

drawCylinder(0.25, 0.02, 100); // Vẽ hình trụ tròn

}

void kimDongHo()

{

mat4 canh = Translate(0.22, 0, 0.02) \* Scale(0.06, 0.016, 0.02);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_dongho \* canh);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

canh = Translate(-0.22, 0, 0.02) \* Scale(0.06, 0.016, 0.02);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_dongho \* canh);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

canh = Translate(0, 0.22, 0.02) \* Scale(0.016, 0.06, 0.02);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_dongho \* canh);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

canh = Translate(0,-0.22, 0.02) \* Scale(0.016, 0.06, 0.02);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_dongho \* canh);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void kimDongHoNho(GLfloat x,GLfloat y)

{

mat4 canh = Translate(x, y, 0.02) \* Scale(0.02, 0.02, 0.02);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_dongho \* canh);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void kimgiay()

{

mat4 canh = Translate(0.08, 0, 0.02) \* Scale(0.16, 0.01, 0.02);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_donghog \* canh);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void kimphut()

{

mat4 canh = Translate(0.075, 0, 0.02) \* Scale(0.13, 0.013, 0.02);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_donghop \* canh);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void kimgio()

{

mat4 canh = Translate(0.05, 0, 0.02) \* Scale(0.1, 0.016, 0.02);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_donghoh \* canh);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void tuLanh()

{

mau(176, 224, 230);

model\_tu = model \* Translate(2, 1, -3.8);

lanhSau();

lanhCanh(-0.58);

lanhCanh(0.58);

lanhGiua(-0.98, 0.04);

lanhGiua(0.2, 0.04);

lanhGiua(0.98, 0.04);

model\_canhLanh = model\_tu \* Translate(-0.58, -0.4, 0.62);

canhLanhDuoi();

canhLanhTren();

mau(254, 254, 254);

lanhGiua(-0.2, 0.01);

lanhGiua(-0.58, 0.01);

lanhGiua(0.7, 0.02);

hoc();

model\_dongho = model \* Translate(2, 2.6, -3.98);

DongHo();

mau(255, 255, 255);

kimDongHo();

kimDongHoNho(0.2, 0.12);

kimDongHoNho(0.12, 0.195);

kimDongHoNho(-0.2, 0.12);

kimDongHoNho(-0.12, 0.195);

kimDongHoNho(-0.2, -0.12);

kimDongHoNho(-0.12, -0.2);

kimDongHoNho(0.2, -0.12);

kimDongHoNho(0.12, -0.2);

mau(255, 0, 0);

model\_donghog = model\_dongho \* RotateZ(giay);

kimgiay();

model\_donghop = model\_dongho \* RotateZ(phut);

kimphut();

model\_donghoh = model\_dongho \* RotateZ(gio);

kimgio();

quayDongHo();

//thùng rác

void khungThungRac()

{

mat4 instance = Translate(-4.2, -0.3, 3.4) \* Scale(0.35, 0.5, 0.04);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_mat \* instance);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

instance = Translate(-4.2, -0.3, 3) \* Scale(0.35, 0.5, 0.04);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_mat \* instance);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

instance = Translate(-4.35, -0.3, 3.2) \* Scale(0.04, 0.5, 0.4);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_mat \* instance);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

instance = Translate(-4.05, -0.3, 3.2) \* Scale(0.04, 0.5, 0.4);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_mat \* instance);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void napThung()

{

mat4 instance = Translate(0.15, 0.08, 0) \* Scale(0.38, 0.04, 0.5);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_rac \* instance);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void dapChan()

{

mat4 instance = Scale(0.1, 0.01, 0.15);

glUniformMatrix4fv(model\_loc, 1, GL\_TRUE, model\_rac \* instance);

glDrawArrays(GL\_TRIANGLES, 0, NumPoints);

}

void thungRac()

{

mau(37, 190, 63);

khungThungRac();

model\_rac = model \* Translate(-2.89, 0.505, 0.01) \* RotateZ(alpha3);

napThung();

model\_rac = model \* Translate(-2.58, chanRac, 0.01);

dapChan();

}

## **2.8. Kỹ thuật di chuyển cam**

* Camera được tạo bằng hàm LookAt với phép chiếu Frustum. Với vị trí trước chính giữa cửa hàng với chiều cao 1.7. Nhìn thẳng.
* Các biến eye, at có các biến toàn cục quản lý. Để cho phép thay đổi camera.
* Thêm biến toàn cục là kc lưu trữ khoảng cách giữa eye và at theo chiều xz.

GLfloat xxx, yyy, zzz; // di chuyển cam

GLfloat l = -1, r = 1, bottom = -1, top = 1, zNear = 1, zFar = 18; // projection

GLfloat anpha, beta, gama; // góc xoay

void gocNhin()

{

vec4 eye = vec4(0 + xxx, 1.5 + yyy, 8 + zzz, 1);

vec4 at = vec4(0 + xxx, 0.5 + yyy, 3 + zzz, 1);

vec4 up = vec4(0, 1, 0, 1);

mat4 u = LookAt(eye, at, up);

glUniformMatrix4fv(view\_loc, 1, GL\_TRUE, u);

mat4 p = Frustum(l, r, bottom, top, zNear, zFar);

glUniformMatrix4fv(projection\_loc, 1, GL\_TRUE, p);

}

- 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 gốc toạ độ (0 0 0)

- Tham số up là vector chỉ hướng của camera, thường là (0 1 0)

- Sử dụng phép chiếu phối cảng Frustum:

+ Vật càng gần camera thì trông sẽ càng to hơn và càng xa camera

thì vật càng nhỏ

+ Giá trị càng tăng thì vùng chiếu càng mở rộng và vật cũng nhỏ

dần

+ NearFar quyết định đến các vật ở xa được trông thấy hoặc các

vật ở gần bị mất đi khi nó thay đổi giá trị // di chuyển cam //

case 'w':

yyy++;

break;

case 's':

yyy--;

break;

case 'a':

xxx--;

break;

case 'd':

xxx++;

break;

// tiến lùi cam //

case 'U':

zzz++;

break;

case 'u':

zzz--;

break;

// xoay phòng //

case 'i':

anpha += 10;

if (anpha > 360) anpha -= 360;

break;

case 'I':

anpha -= 10;

if (anpha < -360) anpha += 360;

break;

case 'o':

beta += 10;

if (beta > 360) beta -= 360;

break;

case 'O':

beta -= 10;

if (beta < -360) beta += 360;

break;

case 'p':

gama += 10;

if (gama > 360) gama -= 360;

break;

case 'P':

gama -= 10;

if (gama < -360) gama += 360;

break;

## **2.9. Kỹ thuật chiếu sáng**

- Khởi tạo các tham số chiếu sáng và tô bóng, mỗi tham số tương

ứng với các cấu trúc của chiếu sáng tô bóng:

+ light\_position: thiết lập vị trí của nguồn sáng

+ light\_ambient: thiết lập màu cho ánh sáng môi trường xung quanh (ở

đây là màu ghi) tương ứng trong công thức trên là Ia

+ light\_diffuse: màu của nguồn sáng, ánh sáng khuếch tán đi là màu gì,

tương ứng trong công thức trên là Id

+ light\_specular: màu của ánh sáng chói, tương ứng Is, với độ trong

suốt là 1.0 ở cuối

+ material\_ambient: hệ số phản xạ của môi trường, tương ứng Ka

+ material\_diffuse: hệ số phản xạ khuếch tán của vật, ở đây nó phản xạ

màu mà mình truyền vào, chính cái này xác định màu sắc của đối

tượng, tương ứng Kd

+ material\_specular: hệ số phản xạ chói sáng, ở đây phản xạ ra màu

trắng, tương ứng Ks

+ material\_shininess: hệ số bóng sáng

+ ambient\_product: theo công thức thì đây là tích của Ia \* Ka

+ diffuse\_product: Id \* Kd

+ specular\_product: Is \* Ks

void mau(GLfloat a, GLfloat b, GLfloat c) {

/\* Khởi tạo các tham số chiếu sáng - tô bóng\*/

point4 light\_position(0.0, 2.0, 0.0, 1.0);

color4 light\_ambient(0.2, 0.2, 0.2, 1.0);

color4 light\_diffuse(mauAnhSang, mauAnhSang, mauAnhSang, 1.0);

color4 light\_specular(choiSang, choiSang, choiSang, 1.0);

color4 material\_ambient(1.0, 0.0, 1.0, 1.0);

color4 material\_diffuse(a / 255.0, b / 255.0, c / 255.0, 1.0);

color4 material\_specular(255.0 / 255.0, 255.0 / 255.0, 255 / 255.0, 1.0);

float material\_shininess = 100.0;

color4 ambient\_product = light\_ambient \* material\_ambient;

color4 diffuse\_product = light\_diffuse \* material\_diffuse;

color4 specular\_product = light\_specular \* material\_specular;

glUniform4fv(glGetUniformLocation(program, "AmbientProduct"), 1, ambient\_product);

glUniform4fv(glGetUniformLocation(program, "DiffuseProduct"), 1, diffuse\_product);

glUniform4fv(glGetUniformLocation(program, "SpecularProduct"), 1, specular\_product);

glUniform4fv(glGetUniformLocation(program, "LightPosition"), 1, light\_position);

glUniform1f(glGetUniformLocation(program, "Shininess"), material\_shininess);

}

void mau(GLfloat a, GLfloat b, GLfloat c) {

/\* Khởi tạo các tham số chiếu sáng - tô bóng\*/

point4 light\_position(0.0, 2.0, 0.0, 1.0);

color4 light\_ambient(0.2, 0.2, 0.2, 1.0);

color4 light\_diffuse(mauAnhSang, mauAnhSang, mauAnhSang, 1.0);

color4 light\_specular(choiSang, choiSang, choiSang, 1.0);

color4 material\_ambient(1.0, 0.0, 1.0, 1.0);

color4 material\_diffuse(a / 255.0, b / 255.0, c / 255.0, 1.0);

color4 material\_specular(255.0 / 255.0, 255.0 / 255.0, 255 / 255.0, 1.0);

float material\_shininess = 100.0;

color4 ambient\_product = light\_ambient \* material\_ambient;

color4 diffuse\_product = light\_diffuse \* material\_diffuse;

color4 specular\_product = light\_specular \* material\_specular;

glUniform4fv(glGetUniformLocation(program, "AmbientProduct"), 1, ambient\_product);

glUniform4fv(glGetUniformLocation(program, "DiffuseProduct"), 1, diffuse\_product);

glUniform4fv(glGetUniformLocation(program, "SpecularProduct"), 1, specular\_product);

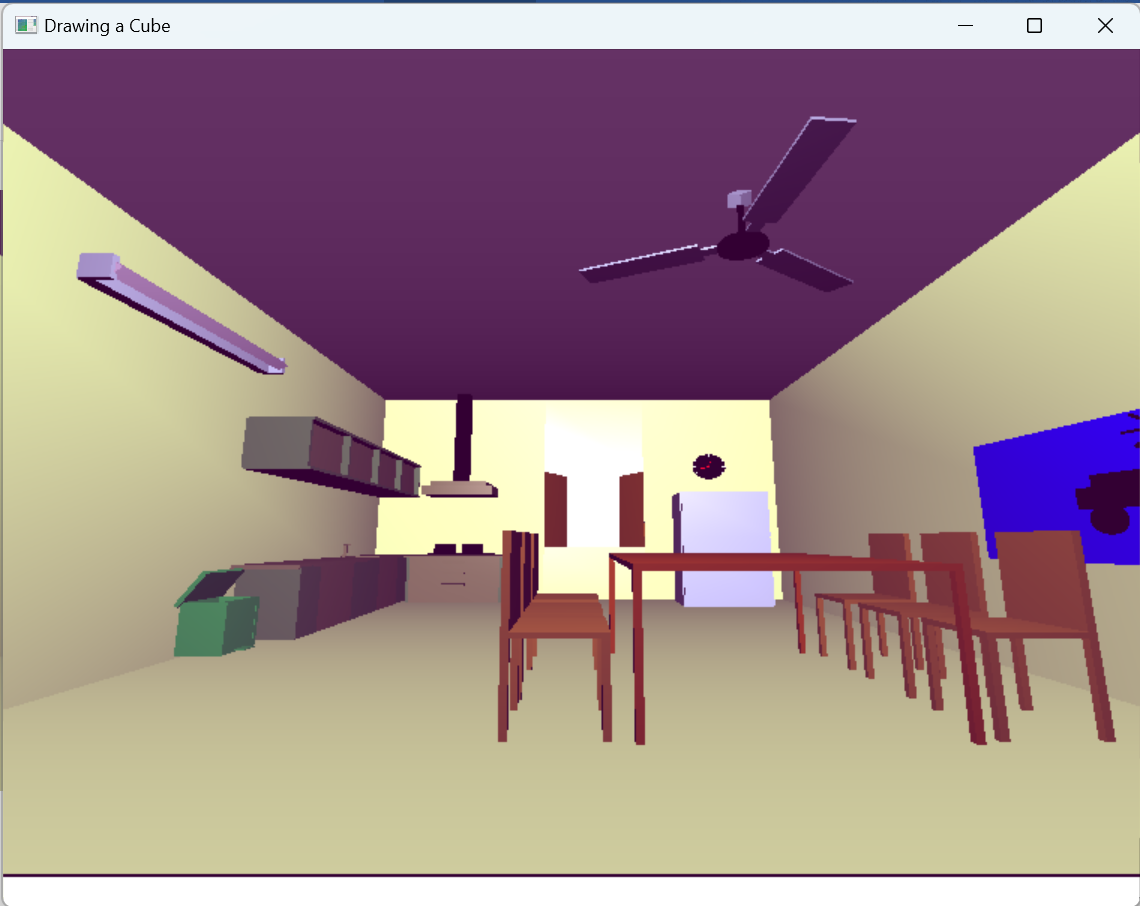
glUniform4fv(glGetUniformLocation(program, "LightPosition"), 1, light\_position);

glUniform1f(glGetUniformLocation(program, "Shininess"), material\_shininess);

}

# **CHƯƠNG 3. KẾT QUẢ ĐẠT ĐƯỢC**

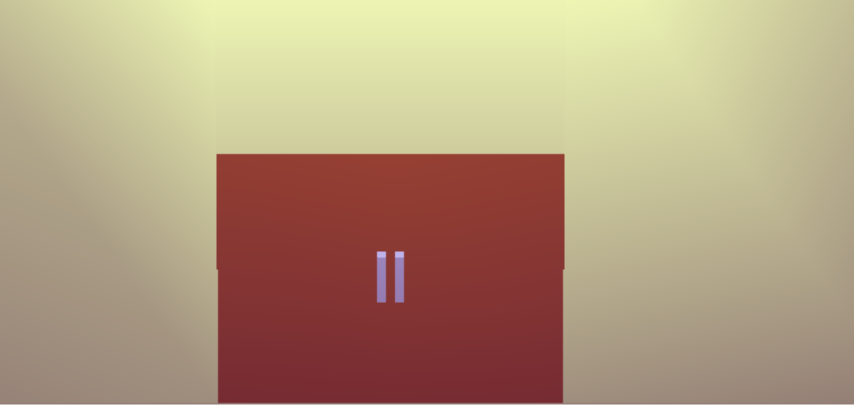
## **3.1. Cảnh quan**



*Hình 3.1: Mô hình phòng bếp hoàn chỉnh*

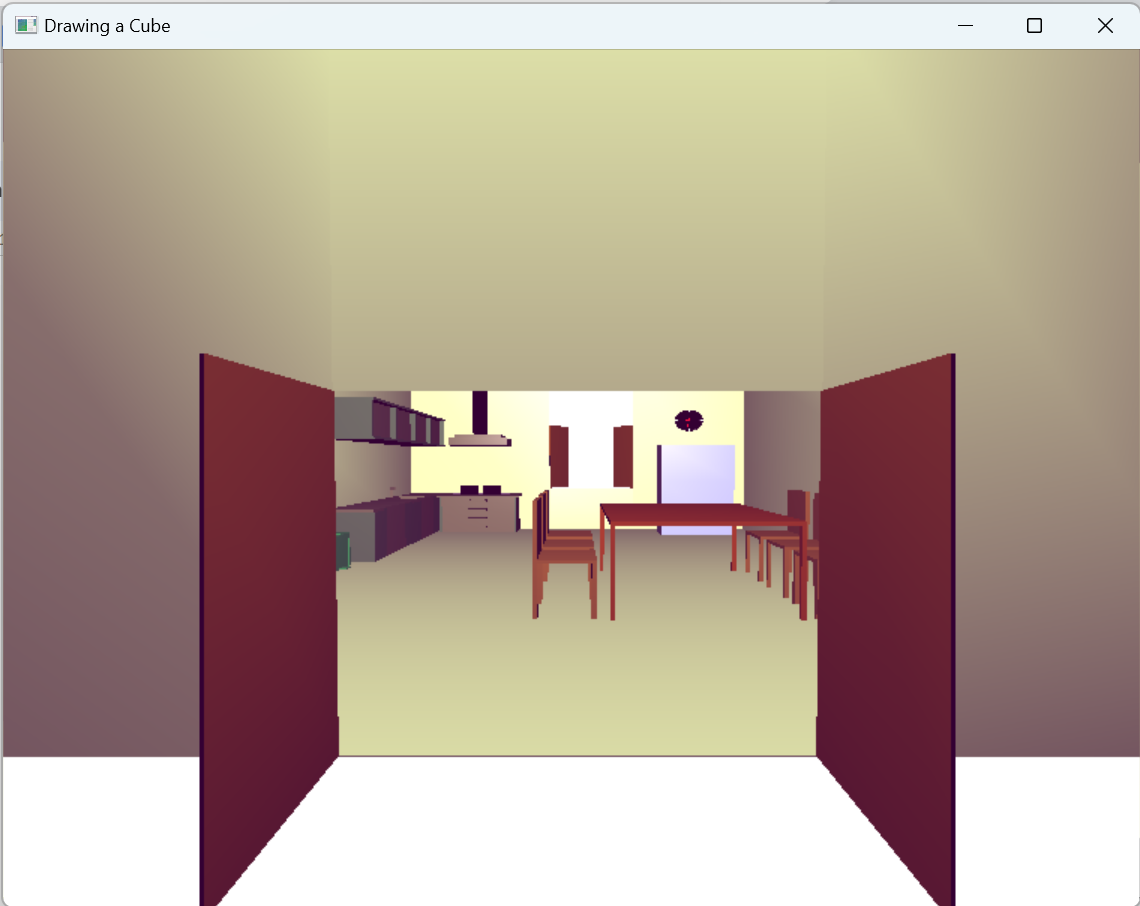
## **3.2. Mô hình cửa chính**

- Hình ảnh



*Hình 3.2.1: Mô hình cửa chính*

* Thực hiện tương tác:
* Các phím:
  + Phím j / J : Mở / Đóng cửa



*Hình 3.2.2: Mô hình mở cửa chính*

## **3.3. Mô hình cửa sổ**

- Hình ảnh:



*Hình 3.3.1: Mô hình cửa sổ*

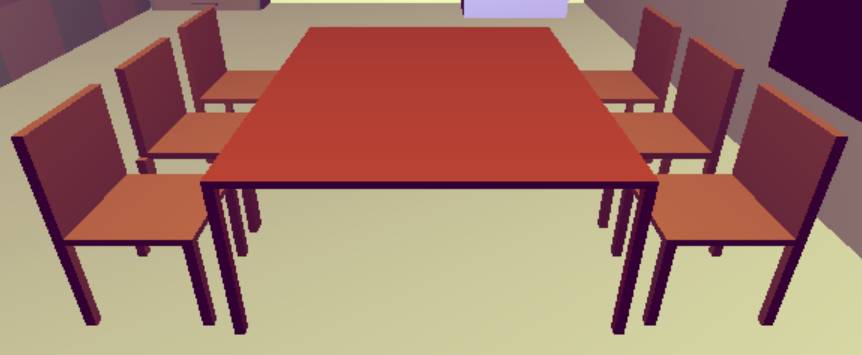
* Thực hiện tương tác:
* Các phím:
  + Phím k / K : Mở / Đóng cửa sổ



*Hình 3.3.2: Mô hình mở cửa sổ*

## **3.4. Mô hình bộ bàn ghế**

- Hình ảnh:



*Hình 3.4.1: Mô hình bộ bàn ghế*

* Thực hiện tương tác:
* Các phím:
* Phím m / M : Kéo ghế vào / ra



*Hình 3.4.2: Mô hình bộ bàn ghế*

## **3.5. Mô hình tủ bếp**

- Hình ảnh



*Hình 3.5.1: Mô hình bộ tủ bếp*

* Thực hiện tương tác:
* Các phím:
* Phím 1 / 2: Mở / Đóng ngăn tủ thứ 1
* Phím 3 / 4: Mở / Đóng ngăn tủ thứ 2
* Phím 5 / 6: Mở / Đóng ngăn tủ thứ 3
* Phím 7 / 8: Mở / Đóng ngăn tủ thứ 4



*Hình 3.5.2: Mô hình mở ngăn tủ bếp*

- Hình ảnh:



*Hình 3.5.3: Mô hình tủ bếp khu rửa bát*

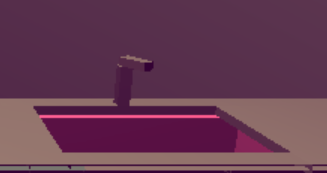
* Thực hiện tương tác:
* Các phím:
  + Phím q / Q: Mở / Đóng ngăn tủ



*Hình 3.5.3: Mô hình mở tủ bếp khu rửa bát*

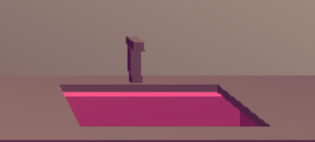
## **3.6. Mô hình vòi nước**

- Hình ảnh:



*Hình 3.6.1: Mô hình vòi nước*

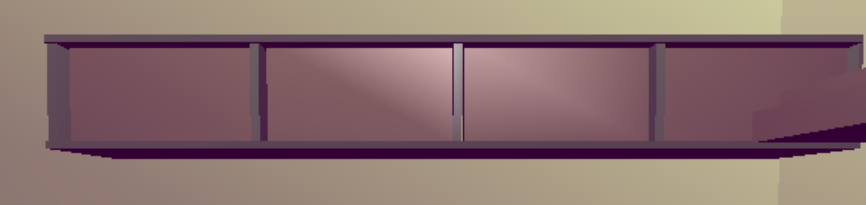
* Thực hiện tương tác:
* Các phím:
  + Phím c / C: Xoay mở / đóng vòi nước



*Hình 3.6.2: Mô hình đóng vòi nước*

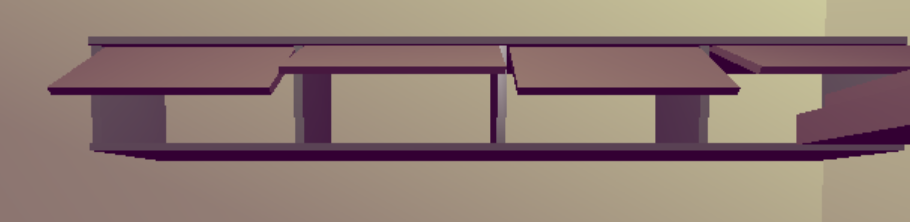
## **3.7. Mô hình tủ đựng đồ**

- Hình ảnh:



*Hình 3.7.1: Mô hình tủ đựng đồ*

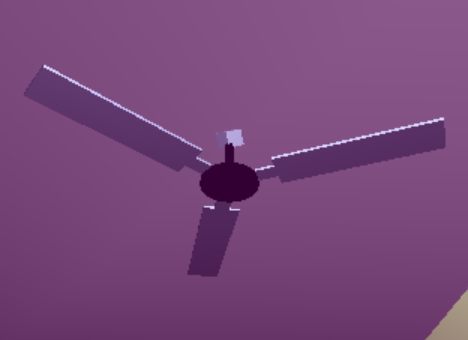
* Thực hiện tương tác:
* Các phím:
  + Phím e / E: Mở / Đóng cánh cửa tủ 1
  + Phím r / R: Mở / Đóng cánh cửa tủ 2
  + Phím t / T: Mở / Đóng cánh cửa tủ 3
  + Phím y / Y: Mở / Đóng cánh cửa tủ 4



*Hình 3.7.2: Mô hình mở cửa tủ đựng đồ*

## **3.8. Mô hình cái quạt**

- Hình ảnh:

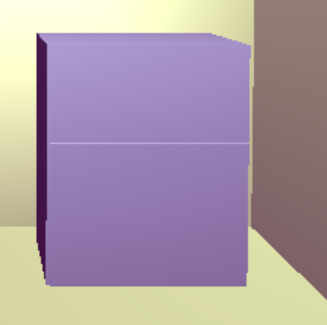


*Hình 3.8.1: Mô hình cánh quạt*

* Thực hiện tương tác:
* Các phím:
  + Phím l / L: Bật / Tắt quạt

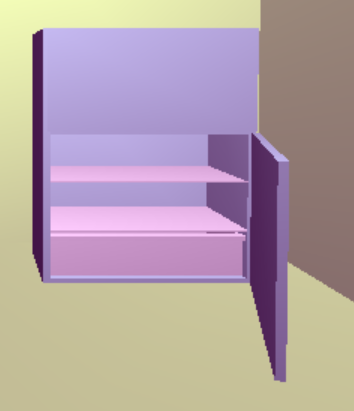
## **3.9. Mô hình tủ lạnh**

- Hình ảnh:



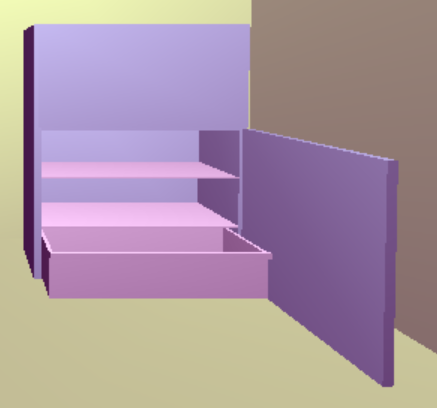
*Hình 3.9.1: Mô hình tủ lạnh*

* Thực hiện tương tác:
* Các phím:
  + Phím b / B: Mở / Đóng cánh dưới tủ lạnh



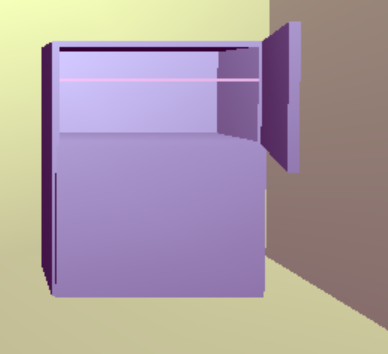
*Hình 3.9.2: Mô hình mở cánh dưới tủ lạnh*

* Phím v / V: Mở / Đóng ngăn kéo tủ lạnh



*Hình 3.9.3: Mô hình mở ngăn kéo tủ lạnh*

* Phím n / N: Mở / Đóng cánh trên tủ lạnh



*Hình 3.9.4: Mô hình mở cánh trên tủ lạnh*

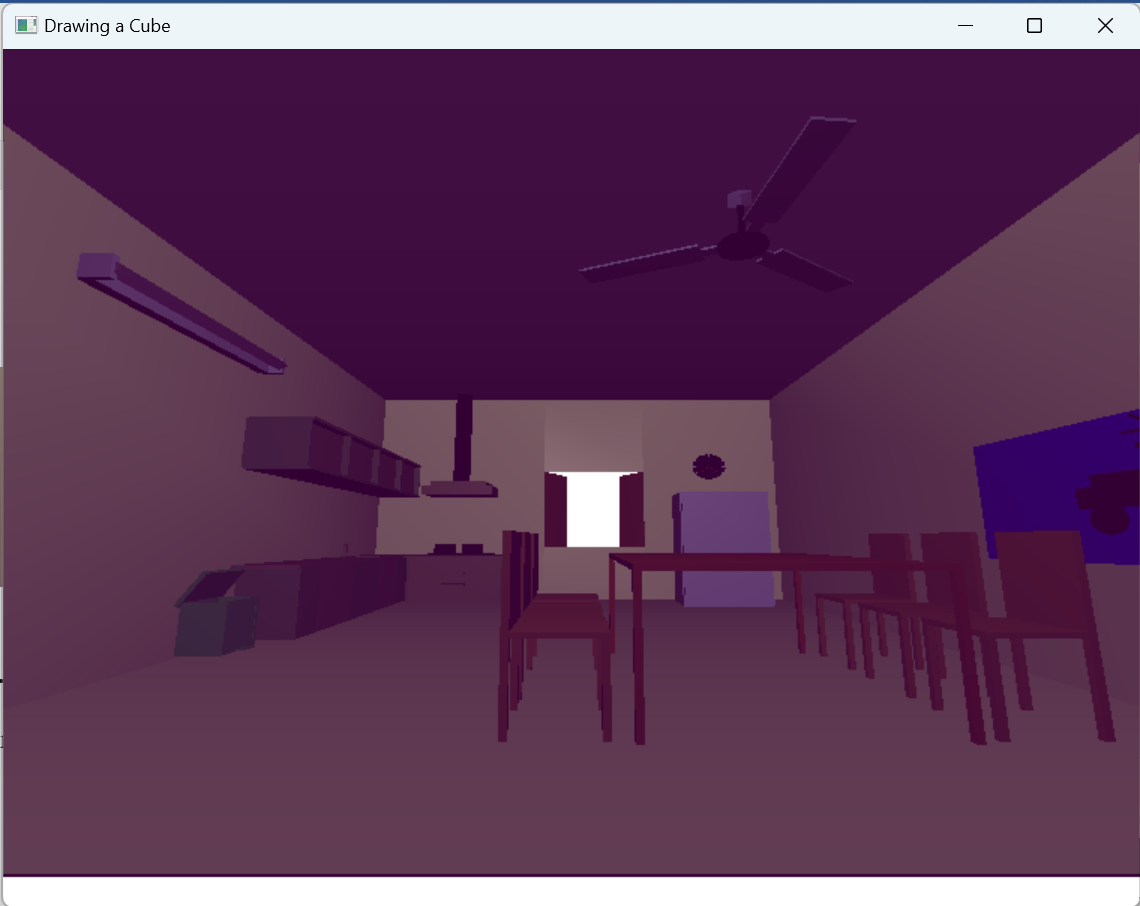
## **3.10. Mô hình bóng đèn**

- Hình ảnh:



*Hình 3.10.1: Mô hình bóng đèn*

* Thực hiện tương tác:
* Các phím:
  + Phím x: Bật / Tắt đèn



*Hình 3.10.2: Mô hình phòng bếp khi tắt điện*

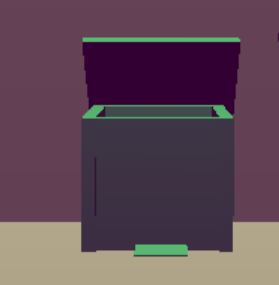
## **3.11. Mô hình thùng rác**

- Hình ảnh:



*Hình 3.11.1: Mô hình thùng rác*

* Thực hiện tương tác:
* Các phím:
  + Phím z / Z: Mở / Đóng thùng rác



*Hình 3.11.2: Mô hình mở thùng rác*

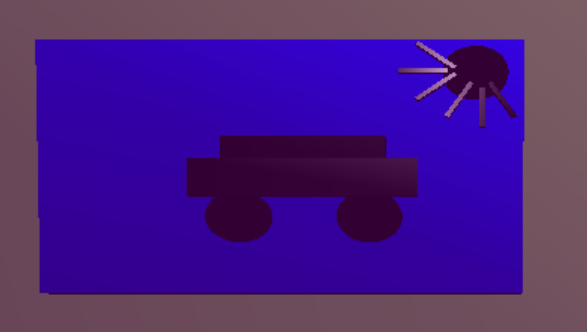
## **3.12. Mô hình tivi**

- Hình ảnh:



*Hình 3.12.1: Mô hình tivi*

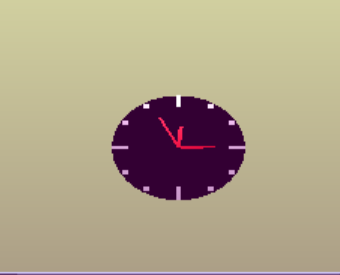
* Thực hiện tương tác:
* Các phím:
  + Phím f: Bật / Tắt tivi



*Hình 3.12.2: Mô hình bật tivi*

## **3.13. Mô hình đồng hồ**

- Hình ảnh:



*Hình 3.12.2: Mô hình đồng hồ*

## **3.14. Điều khiển camera**

Phím w, s, a, d: Di chuyển camera lên, xuống, trái, phải

Phím u, U: Tiến, lùi camera

Phím i, I, o, O, p, P: Quay camera lên, xuống, trái, phải

# **KẾT LUẬN**

Qua việc thực hiện nghiên cứu đề tài “**Mô phỏng phòng bếp bằng OpenGL khả lập trình và Visual C++**”. Nhóm chúng em đã tìm hiểu sâu hơn về môn đồ họa các khối hình. Việc thực hiện đề tài giúp cải thiện tư duy cố hữu trong code thành sáng tạo hơn. Bên cạnh đó, việc làm nghiên cứu giúp chúng em đoàn kết hơn, rèn luyện cho chúng em kỹ năng làm việc nhóm.

Cũng thông qua lần này, nhóm hình thành cách làm việc của nhóm, bắt đầu từ việc thống nhất ban đầu xây dựng một bản chung cho cả nhóm, sau đấy mỗi người được phân chia code phần riêng của mình. Việc thống nhất tỷ lệ, kích thước, màu sắc... từ ban đầu giúp quá trình code và ghép code diễn ra dễ dàng và sớm hoàn thành được mục tiêu đề ra.

Chúng em cảm ơn cô Vũ Minh Yến đã tận tình giảng dạy chúng em trong môn đồ họa máy tính. Giúp đỡ chúng em trong quá trình nghiên cứu đề tài. Cô chia sẻ những tài liệu cùng sự giảng giải nhiệt tình của cô đã giúp chúng em hiểu hơn và hoàn thành tốt bài báo cáo.

Một lần nữa chúng em xin chân thành cảm ơn!

# **TÀI LIỆU THAM KHẢO**

[1] Tham khảo giáo trình môn đồ họa máy tính

[2] Slide bài giảng và các tài liệu học tập của ThS. Vũ Minh Yến

[3] Hình ảnh các đồ vật, nhà bếp trên mạng

[4] Internet