TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI

KHOA CÔNG NGHỆ THÔNG TIN

======\*\*\*======

****

BÁO CÁO THỰC NGHIỆM MÔN HỌC ĐỒ HỌA MÁY TÍNH

ĐỀ TÀI

MÔ PHỎNG GIAN HÀNG BÁN CÁC LOẠI Ô TÔ ĐỒ CHƠI TRẺ EM BẰNG OPENGL KHẢ LẬP TRÌNH VÀ VISUAL C++

|  |  |
| --- | --- |
| GVHD: | TS. Vũ Minh Yến |
| Nhóm 3 | Lớp 20233IT6010001 |
| Thành viên: | 1. Đoàn Khắc Minh - 2021601536 |
|  | 2. Nguyễn Văn Băng - 2018607487 |
|  | 3. Nguyễn Hữu Bắc - 2021600159 |

Hà Nội, tháng 02 năm 2024

# MỤC LỤC

[MỤC LỤC 1](#_Toc159965513)

[MỞ ĐẦU 3](#_Toc159965514)

[Chương 1: XÁC ĐỊNH VÀ PHÂN TÍCH BÀI TOÁN 4](#_Toc159965515)

[1.1 Giới thiệu đề tài 4](#_Toc159965516)

[1.2 Mô tả các đối tượng cần thiết 4](#_Toc159965517)

[1.3. Mô tả kịch bản chương trình 5](#_Toc159965518)

[1.3.1 Yêu cầu của chương trình 5](#_Toc159965519)

[1.3.2 Mô tả chức năng của chương trình 5](#_Toc159965520)

[Chương 2: CÀI ĐẶT CHƯƠNG TRÌNH 6](#_Toc159965521)

[2.1. Kỹ thuật tạo mô hình ô tô 6](#_Toc159965522)

[2.2 Kỹ thuật tạo mô hình tên lửa 25](#_Toc159965523)

[2.3 Kỹ thuật tạo mô hình trống 27](#_Toc159965524)

[2.4 Kỹ thuật tạo mô hình robot 29](#_Toc159965525)

[2.5 Kỹ thuật tạo mô hình dao 35](#_Toc159965526)

[2.6 Kỹ thuật tạo mô hình bàn thanh toán, máy tính 40](#_Toc159965527)

[2.7 Kỹ thuật điều khiển camera 43](#_Toc159965528)

[2.8 Kỹ thuật chiếu sáng 44](#_Toc159965529)

[Chương 3: KẾT QUẢ 46](#_Toc159965530)

[3.1 Mô hình ô tô 46](#_Toc159965531)

[3.2 Mô hình tên lửa 47](#_Toc159965532)

[3.3 Mô hình trống 48](#_Toc159965533)

[3.4 Mô hình robot 48](#_Toc159965534)

[3.5 Mô hình dao 48](#_Toc159965535)

[3.6 Mô hình bàn thanh toán, máy tính 49](#_Toc159965536)

[3.7 Mô hình nhân viên bán hàng 49](#_Toc159965537)

[3.8 Mô hình cửa ra vào 50](#_Toc159965538)

# 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 gian hàng bán các mô hình ô tô đồ chơi trẻ em 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 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!

# XÁC ĐỊNH VÀ PHÂN TÍCH BÀI TOÁN

## 1.1 Giới thiệu đề tài

Tên đề tài: Mô phỏng gian hàng bán các mô hình ô tô đồ chơi trẻ em bằng OpenGL khả lập trình và Visual C++

## 1.2 Mô tả các đối tượng cần thiết

#### 1.2.1 Gian hàng

Mô hình gồm 4 tường, 1 trần nhà, 1 nền nhà

#### 1.2.2 Tủ trưng bày

Dùng để trưng bày đồ vật

#### 1.2.3 Bàn thanh toán

Dùng để trưng bày tên lửa (có thể chuyển động)

#### 1.2.4 Máy tính

#### 1.2.5 Dao

#### 1.2.6 Ô tô

Có thể di chuyển đồng thời bánh xe quay

Nâng hạ thùng xe, mở cửa

Xe cẩu phần máy cẩu xoay trục Oy, cần cẩu lên xuống …

#### 1.2.7 Tên lửa

Có thể xoay tròn

#### 1.2.8 Robot

#### 1.2.9 Trống

## 1.3. Mô tả kịch bản chương trình

### 1.3.1 Yêu cầu của chương trình

* Mô phỏng được gian hàng bán các mô hình đồ chơi gồm có các mô hình sau: bàn ghế thanh toán, tủ trưng bày, ô tô, tên lửa, robot, trống, dao, ...
* Mô phỏng được chuyển động của các đối tượng cụ thể: Ô tô di chuyển, tên lửa quay, xe tải đổ thùng, xe cẩu xoay, ...

### 1.3.2 Mô tả chức năng của chương trình

* Xe ô tô: Phím ‘>’ và ‘<’ để di chuyển
* Xe tải:   
  + Phím ‘u’ ‘U’ ‘i’ ‘I’ để mở đóng cửa   
  + Phím ‘1’ ‘2’ nâng hạ thùng xe  
  + Phím ‘[’ ‘]’ di chuyển xe
* Xe cẩu  
  + Phím ‘=’ ‘-’ di chuyển xe  
  + Phím ‘5’ ‘6’ ‘%’ ‘^’ mở cửa xe  
  + Phím ‘0’ ‘)’ mở cửa cẩu, ‘9’ ‘(‘ xoay cẩu, ‘8’ ‘\*’ nâng cẩu
* Tên lửa: Phím ‘>’ và ‘<’ để xoay
* Cánh cửa: ‘o’ ‘O’ mở đóng cửa

# CÀI ĐẶT CHƯƠNG TRÌNH

## 2.1. Kỹ thuật tạo mô hình ô tô

#### - Ô tô 1

GLfloat wheelRotZ = 0.0;

void drawCabin(mat4 carCtm) {

//khung xe

// upper

initReflection(0.0, 0.54509807, 0.27058825, 0.4, 0.4, 0.4, 0.9, 0.9, 0.9); // SpringGreen

glUniformMatrix4fv(model, 1, GL\_TRUE, carCtm \* Translate(0.0, 3.0, 0.0) \* Scale(4.0, 0.1, 4.0));

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

// right sheet

glUniformMatrix4fv(model, 1, GL\_TRUE, carCtm \* Translate(0.0, 0.45, 1.95) \* Scale(4.0, 5.0, 0.1));

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

// left sheet

glUniformMatrix4fv(model, 1, GL\_TRUE, carCtm \* Translate(0.0, 0.45, -1.95) \* Scale(4.0, 5.0, 0.1));

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

// bottom sheet

glUniformMatrix4fv(model, 1, GL\_TRUE, carCtm \* Translate(-5.0, -2.0, 0.0) \* Scale(14.0, 0.1, 4.0));

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

// front sheet

glUniformMatrix4fv(model, 1, GL\_TRUE, carCtm \* Translate(1.95, 0.45, 0.0) \* Scale(0.1, 5.0, 4.0));

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

// back sheet

glUniformMatrix4fv(model, 1, GL\_TRUE, carCtm \* Translate(-1.95, 0.45, 0.0) \* Scale(0.1, 5.0, 4.0));

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

//kính trước và 2 bên

// right glass

initReflection(0.9411765, 1.0, 1.0, 0.2, 0.2, 0.2, 1.0, 1.0, 1.0); // Azure1

glUniformMatrix4fv(model, 1, GL\_TRUE, carCtm \* Translate(0.5, 1.55, 1.96) \* Scale(2.0, 1.5, 0.1));

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

// left glass

glUniformMatrix4fv(model, 1, GL\_TRUE, carCtm \* Translate(0.5, 1.55, -1.96) \* Scale(2.0, 1.5, 0.1));

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

// front glass

glUniformMatrix4fv(model, 1, GL\_TRUE, carCtm \* Translate(1.96, 1.7, 0.0) \* Scale(0.1, 1.8, 3.0));

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

// Front decor highlight

initReflection(0.0, 0.0, 0.0, 0.4, 0.4, 0.4, 0.8, 0.8, 0.8); // Black

glUniformMatrix4fv(model, 1, GL\_TRUE, carCtm \* Translate(2.0, -1.8, 0.0) \* Scale(0.3, 0.5, 4.0)); //thanh dưới mặt trước cabin

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

// Front decor Id board

initReflection(1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0); // White

glUniformMatrix4fv(model, 1, GL\_TRUE, carCtm \* Translate(2.01, -1.8, 0.0) \* Scale(0.3, 0.5, 1.0)); // ở giữa thanh dưới cabin

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

// Front decor lights Đèn ô tô

for (int i = 0; i < 360; i++) {

initReflection(1.0, 0.75686276, 0.14509805, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0); // Goldenrod1

glUniformMatrix4fv(model, 1, GL\_TRUE, carCtm \* Translate(1.96, -1.0, 1.3) \* RotateY(90) \* RotateZ(i \* 1.0) \* Scale(0.4, 0.4, 0.1));

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

}

for (int i = 0; i < 360; i++) {

glUniformMatrix4fv(model, 1, GL\_TRUE, carCtm \* Translate(1.96, -1.0, -1.3) \* RotateY(90) \* RotateZ(i \* 1.0) \* Scale(0.4, 0.4, 0.1));

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

}

// fornt decor Radiators

// các thanh ngang mặt trc cabin

initReflection(0.0, 0.0, 0.0, 0.4, 0.4, 0.4, 0.8, 0.8, 0.8); // Black

glUniformMatrix4fv(model, 1, GL\_TRUE, carCtm \* Translate(2.01, 0.3, 0.0) \* Scale(0.1, 0.05, 2.3));

glDrawArrays(GL\_TRIANGLES, 36 \* 1, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, carCtm \* Translate(2.01, 0.15, 0.0) \* Scale(0.1, 0.05, 2.3));

glDrawArrays(GL\_TRIANGLES, 36 \* 1, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, carCtm \* Translate(2.01, 0.0, 0.0) \* Scale(0.1, 0.05, 2.3));

glDrawArrays(GL\_TRIANGLES, 36 \* 1, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, carCtm \* Translate(2.01, -0.15, 0.0) \* Scale(0.1, 0.05, 2.3));

glDrawArrays(GL\_TRIANGLES, 36 \* 1, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, carCtm \* Translate(2.01, -0.3, 0.0) \* Scale(0.1, 0.05, 2.3));

glDrawArrays(GL\_TRIANGLES, 36 \* 1, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, carCtm \* Translate(2.01, -0.45, 0.0) \* Scale(0.1, 0.05, 2.3));

glDrawArrays(GL\_TRIANGLES, 36 \* 1, POINTS\_PER\_CUBE);

}

void drawTrunk(mat4 carCtm) {

drawCabin(carCtm);

// lower

initReflection(0.93333334, 0.2509804, 0.0, 0.3, 0.0, 0.0, 1.0, 1.0, 1.0); // OrangeRed2 (238 64 0)

glUniformMatrix4fv(model, 1, GL\_TRUE, carCtm \* Translate(-7.25, -1.8, 0.0) \* Scale(9.5, 0.3, 4.0));

glDrawArrays(GL\_TRIANGLES, 36 \* 3, POINTS\_PER\_CUBE);

// right

glUniformMatrix4fv(model, 1, GL\_TRUE, carCtm \* Translate(-7.25, -0.3, 1.85) \* Scale(9.5, 3.0, 0.3));

glDrawArrays(GL\_TRIANGLES, 36 \* 2, POINTS\_PER\_CUBE);

// left

glUniformMatrix4fv(model, 1, GL\_TRUE, carCtm \* Translate(-7.25, -0.3, -1.85) \* Scale(9.5, 3.0, 0.3));

glDrawArrays(GL\_TRIANGLES, 36 \* 2, POINTS\_PER\_CUBE);

// top

glUniformMatrix4fv(model, 1, GL\_TRUE, carCtm \* Translate(-2.65, -0.3, 0.0) \* Scale(0.3, 3.0, 4.0));

glDrawArrays(GL\_TRIANGLES, 36 \* 1, POINTS\_PER\_CUBE);

// bot

glUniformMatrix4fv(model, 1, GL\_TRUE, carCtm \* Translate(-11.85, -0.3, 0.0) \* Scale(0.3, 3.0, 4.0));

glDrawArrays(GL\_TRIANGLES, 36 \* 1, POINTS\_PER\_CUBE);

}

void drawWheel(mat4 carCtm, int mode) { // Use mode to determine whether the wheels should be rotated in Z Axis or not (we use it to distinguish the motion object and garniture object)

// front wheel

GLfloat factor = (mode == 1) ? 1.0 : 0.0;

initReflection(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0); // black

for (int i = 0; i < 360; i++) {

glUniformMatrix4fv(model, 1, GL\_TRUE, carCtm \* Translate(0.0, -2.0, 1.61) \* RotateZ(i \* 1.0) \* RotateZ(wheelRotZ \* factor) \* Scale(1.0, 1.7, 0.8)); // 0.35

glDrawArrays(GL\_TRIANGLES, 36 \* 7, POINTS\_PER\_CUBE);

}

initReflection(0.8117647, 0.8117647, 0.8117647, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0); // gray81

for (int i = 0; i < 360; i += 60) {

glUniformMatrix4fv(model, 1, GL\_TRUE, carCtm \* Translate(0.0, -2.0, 1.62) \* RotateZ(i \* 1.0) \* RotateZ(wheelRotZ \* factor) \* Scale(0.5, 0.85, 0.8)); // 0.35

glDrawArrays(GL\_TRIANGLES, 36 \* 1, POINTS\_PER\_CUBE);

}

initReflection(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0); // black

for (int i = 0; i < 360; i++) {

glUniformMatrix4fv(model, 1, GL\_TRUE, carCtm \* Translate(0.0, -2.0, -1.61) \* RotateZ(i \* 1.0) \* RotateZ(wheelRotZ \* factor) \* Scale(1.0, 1.7, 0.8));

glDrawArrays(GL\_TRIANGLES, 36 \* 7, POINTS\_PER\_CUBE);

}

initReflection(0.8117647, 0.8117647, 0.8117647, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0); // gray81

for (int i = 0; i < 360; i += 60) {

glUniformMatrix4fv(model, 1, GL\_TRUE, carCtm \* Translate(0.0, -2.0, -1.62) \* RotateZ(i \* 1.0) \* RotateZ(wheelRotZ \* factor) \* Scale(0.5, 0.85, 0.8)); // 0.35

glDrawArrays(GL\_TRIANGLES, 36 \* 1, POINTS\_PER\_CUBE);

}

// back wheel

initReflection(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0); // black

for (int i = 0; i < 360; i++) {

glUniformMatrix4fv(model, 1, GL\_TRUE, carCtm \* Translate(-10.0, -2.0, 1.61) \* RotateZ(i \* 1.0) \* RotateZ(wheelRotZ \* factor) \* Scale(1.0, 1.7, 0.8)); // 0.35

glDrawArrays(GL\_TRIANGLES, 36 \* 7, POINTS\_PER\_CUBE);

}

initReflection(0.8117647, 0.8117647, 0.8117647, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0); // gray81

for (int i = 0; i < 360; i += 60) {

glUniformMatrix4fv(model, 1, GL\_TRUE, carCtm \* Translate(-10.0, -2.0, 1.62) \* RotateZ(i \* 1.0) \* RotateZ(wheelRotZ \* factor) \* Scale(0.5, 0.85, 0.8)); // 0.35

glDrawArrays(GL\_TRIANGLES, 36 \* 1, POINTS\_PER\_CUBE);

}

initReflection(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0); // black

for (int i = 0; i < 360; i++) {

glUniformMatrix4fv(model, 1, GL\_TRUE, carCtm \* Translate(-10.0, -2.0, -1.61) \* RotateZ(i \* 1.0) \* RotateZ(wheelRotZ \* factor) \* Scale(1.0, 1.7, 0.8));

glDrawArrays(GL\_TRIANGLES, 36 \* 7, POINTS\_PER\_CUBE);

}

initReflection(0.8117647, 0.8117647, 0.8117647, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0); // gray81

for (int i = 0; i < 360; i += 60) {

glUniformMatrix4fv(model, 1, GL\_TRUE, carCtm \* Translate(-10.0, -2.0, -1.62) \* RotateZ(i \* 1.0) \* RotateZ(wheelRotZ \* factor) \* Scale(0.5, 0.85, 0.8)); // 0.35

glDrawArrays(GL\_TRIANGLES, 36 \* 1, POINTS\_PER\_CUBE);

}

#### - Xe tải

mat4 model\_xetai;

void khungxe()

{

initReflection(0.0, 0.54509807, 0.27058825, 0.4, 0.4, 0.4, 0.9, 0.9, 0.9);

mat4 instance = Translate(0, 0, 0) \* Scale(8, 0.2, 2);

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

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

instance = Translate(0, -0.2, 0) \* Scale(8, 0.5, 2);

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

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

instance = Translate(-3.95, 1, 1) \* Scale(0.1, 2, 0.04);

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

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

instance = Translate(-3.95, 1, -1) \* Scale(0.1, 2, 0.04);

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

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

instance = Translate(-1.98, 1, 0) \* Scale(0.04, 2, 2);

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

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

instance = Translate(-3.98, 0.6, 0) \* Scale(0.04, 1.2, 2);

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

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

instance = Translate(-3, 2, 0) \* Scale(2, 0.04, 2);

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

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

}

mat4 model\_cuaxe1, model\_cuaxe2;

void cuaxe1()

{

mat4 instance = Translate(0.95, 0, 0) \* Scale(1.9, 1.2, 0.04);

glUniformMatrix4fv(model, 1, GL\_TRUE, model\_cuaxe1 \* instance);

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

}

void cuaxe2()

{

mat4 instance = Translate(0.95, 0, 0) \* Scale(1.9, 1.2, 0.04);

glUniformMatrix4fv(model, 1, GL\_TRUE, model\_cuaxe2 \* instance);

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

}

mat4 model\_thungxe;

void khungthungxe()

{

initReflection(0.93333334, 0.2509804, 0.0, 0.3, 0.0, 0.0, 1.0, 1.0, 1.0);

mat4 instance = Translate(-2.8, 0, 0) \* Scale(5.6, 0.1, 2);

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

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

instance = Translate(-2.8, 1, 0.95) \* Scale(5.6, 2, 0.04);

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

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

instance = Translate(-2.8, 1, -0.95) \* Scale(5.6, 2, 0.04);

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

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

instance = Translate(-5.6, 1, 0) \* Scale(0.04, 2, 2);

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

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

}

mat4 model\_napthung;

void napthung()

{

mat4 instance = Translate(0, -1, 0) \* Scale(0.04, 2, 2);

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

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

}

GLfloat gocthung, gocnapthung;

GLfloat goccua1, goccua2;

mat4 model\_banhxe;

void banhxe()

{

initReflection(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0);

mat4 instance = Translate(0, 0, -1)\* Scale(0.8, 0.8, 0.04);

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

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

instance = Translate(0, 0, 1) \* Scale(0.8, 0.8, 0.04);

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

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

}

GLfloat banhxequay,xetaidc;

void xetai()

{

model\_xetai = Translate(xetaidc, 0, 0) \* Translate(0.5, 1.2, -0.8) \* Scale(0.1, 0.1, 0.1);

khungxe();

model\_cuaxe1 = model\_xetai \* Translate(-3.9, 0.7, 1) \* RotateY(goccua1); //--

cuaxe1();

model\_cuaxe2 = model\_xetai \* Translate(-3.9, 0.7, -1) \* RotateY(goccua2); //++

cuaxe2();

model\_thungxe = model\_xetai \* Translate(4, 0.1, 0) \* RotateZ(gocthung);

khungthungxe();

model\_napthung = model\_thungxe \* Translate(0, 2, 0) \* RotateZ(gocnapthung);

napthung();

model\_banhxe = model\_xetai \* Translate(0, -0.5, 0) \* RotateZ(banhxequay);

banhxe();

model\_banhxe = model\_xetai \* Translate(3, -0.5, 0) \* RotateZ(banhxequay);

banhxe();

model\_banhxe = model\_xetai \* Translate(-3, -0.5, 0) \* RotateZ(banhxequay);

banhxe();

}

#### - Xe cẩu

GLfloat banhxecau, xecaudc, goccuac1, goccuac2;

mat4 model\_sauxecau;

void sauxecau()

{

initReflection(0.93333334, 0.2509804, 0.0, 0.3, 0.0, 0.0, 1.0, 1.0, 1.0);

mat4 instance = Translate(0, 0, 0) \* Scale(5, 0.5, 2);

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

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

instance = Translate(2, 0.5, 0.5) \* Scale(1, 0.5, 1);

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

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

instance = Translate(0, 1, 0.02) \* Scale(3, 1.5, 0.04);

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

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

instance = Translate(0, 1, 1.02) \* Scale(3, 1.5, 0.04);

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

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

instance = Translate(-2, 0.5, 0.02) \* Scale(1, 0.5, 0.04);

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

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

instance = Translate(-2, 1.7, 0.02) \* Scale(1, 0.1, 0.04);

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

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

instance = Translate(-2.48, 0.5, 0.5) \* Scale(0.04, 0.5, 1);

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

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

instance = Translate(0.5, 1, 0.5) \* Scale(2, 1.5, 1);

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

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

instance = Translate(-1.5, 1.73, 0.5) \* Scale(2, 0.04, 1);

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

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

instance = Translate(-2.48, 1.63, 0.5) \* Scale(0.04, 0.2, 1);

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

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

instance = Translate(-2.46, 1, 0.02) \* Scale(0.1, 1.5, 0.04);

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

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

instance = Translate(-2.46, 1, 0.98) \* Scale(0.1, 1.5, 0.04);

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

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

instance = Translate(2, 0.45, -0.15) \* Scale(0.4, 0.4, 0.3);

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

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

instance = Translate(2, 0.45, -0.85) \* Scale(0.4, 0.4, 0.3);

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

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

}

mat4 model\_caucuaxe, model\_caucuaxe2;

void caucuaxe()

{

initReflection(0.7764706, 0.8862745, 1.0, 0.0, 0.0, 0.0, 0.6, 0.3, 0.3);

mat4 instance = Translate(-2.5, 0, 0) \* Scale(5, 0.4, 0.4);

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

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

}

void caucuaxe2()

{

mat4 instance = Scale(4, 0.2, 0.2);

glUniformMatrix4fv(model, 1, GL\_TRUE, model\_caucuaxe2 \* instance);

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

instance = Translate(-1.95, -0.4, 0) \* Scale(0.1, 0.8, 0.1);

glUniformMatrix4fv(model, 1, GL\_TRUE, model\_caucuaxe2 \* instance);

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

instance = Translate(-1.7, -0.6, 0) \* Scale(0.1, 0.2, 0.1);

glUniformMatrix4fv(model, 1, GL\_TRUE, model\_caucuaxe2 \* instance);

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

instance = Translate(-1.8, -0.75, 0) \* Scale(0.3, 0.1, 0.1);

glUniformMatrix4fv(model, 1, GL\_TRUE, model\_caucuaxe2 \* instance);

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

}

mat4 model\_cuasaucau;

void cuasaucau()

{

mat4 instance = Translate(0.45, 0, 0) \* Scale(0.9, 0.5, 0.04);

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

glDrawArrays(GL\_TRIANGLES, 0, POINTS\_PER\_CUBE);

}

GLfloat goccuacau, banhxequay1;

GLfloat goccau, cau2 = -3.2, goccau1 = 0;

void xecau()

{

model\_xetai = Translate(xecaudc, 0, 0) \* Translate(0.5, 0.73, -0.8) \*Scale(0.1, 0.1, 0.1);

khungxe();

model\_cuaxe1 = model\_xetai \* Translate(-3.9, 0.7, 1) \* RotateY(goccuac1); //--

cuaxe1();

model\_cuaxe2 = model\_xetai \* Translate(-3.9, 0.7, -1) \* RotateY(goccuac2); //++

cuaxe2();

model\_sauxecau = model\_xetai \* Translate(1.5, 0.35, 0) \* RotateY(goccau);

sauxecau();

model\_cuasaucau = model\_sauxecau \* Translate(-2.41, 0.5, 0.98) \* RotateY(goccuacau);

cuasaucau();

model\_caucuaxe = model\_sauxecau \* Translate(2, 0.5, -0.5) \* RotateZ(goccau1);

caucuaxe();

model\_caucuaxe2 = model\_caucuaxe \* Translate(cau2, 0, 0);

caucuaxe2();

model\_banhxe = model\_xetai \* Translate(0, -0.5, 0) \* RotateZ(banhxequay1);

banhxe();

model\_banhxe = model\_xetai \* Translate(3, -0.5, 0) \* RotateZ(banhxequay1);

banhxe();

model\_banhxe = model\_xetai \* Translate(-3, -0.5, 0) \* RotateZ(banhxequay1);

banhxe();

}

## 2.2 Kỹ thuật tạo mô hình tên lửa

// Rocket

GLfloat rocketRotY = 0.0;

void drawRocket(mat4 rocketCtm, int mode) {

GLfloat factor = (mode == 1) ? 1.0 : 0.0;

// lower SlateGray1 (198 226 255)

initReflection(0.7764706, 0.8862745, 1.0, 0.0, 0.0, 0.0, 0.6, 0.3, 0.3);

for (int i = 0; i < 360; i++) {

glUniformMatrix4fv(model, 1, GL\_TRUE, rocketCtm \* RotateZ(45 \* factor) \* Translate(0.0, -1.5, 0.0) \* RotateY(i \* 1.0 + rocketRotY \* factor) \* Scale(5.0, 3.0, 5.0));

glDrawArrays(GL\_TRIANGLES, 36 \* 15, POINTS\_PER\_CUBE);

}

//// mid body

for (int i = 0; i < 360; i++) {

glUniformMatrix4fv(model, 1, GL\_TRUE, rocketCtm \* RotateZ(45 \* factor) \* RotateY(i \* 1.0 + rocketRotY \* factor) \* Scale(7.0, 9.0, 7.0));

glDrawArrays(GL\_TRIANGLES, 594, 18);

}

// upper IndianRed4 (139 58 58)

initReflection(0.54509807, 0.22745098, 0.22745098, 0.1, 0.1, 0.1, 0.0, 0.0, 0.0);

for (int i = 0; i < 360; i++) {

glUniformMatrix4fv(model, 1, GL\_TRUE, rocketCtm \* RotateZ(45 \* factor) \* Translate(0.0, 5.45, 0.0) \* RotateY(i \* 1.0 + rocketRotY \* factor) \* Scale(3.9, 1.9, 3.9));

glDrawArrays(GL\_TRIANGLES, 576, 18);

}

// wings

glUniformMatrix4fv(model, 1, GL\_TRUE, rocketCtm \* RotateZ(45 \* factor) \* RotateY(0.0 + rocketRotY \* factor) \* Translate(-3.5, -1.5, 0.0) \* Scale(3.0, 3.0, 1.0));

glDrawArrays(GL\_TRIANGLES, 576, 18);

glUniformMatrix4fv(model, 1, GL\_TRUE, rocketCtm \* RotateZ(45 \* factor) \* RotateY(120 + rocketRotY \* factor) \* Translate(-3.5, -1.5, 0.0) \* Scale(3.0, 3.0, 1.0));

glDrawArrays(GL\_TRIANGLES, 576, 18);

glUniformMatrix4fv(model, 1, GL\_TRUE, rocketCtm \* RotateZ(45 \* factor) \* RotateY(240 + rocketRotY \* factor) \* Translate(-3.5, -1.5, 0.0) \* Scale(3.0, 3.0, 1.0));

glDrawArrays(GL\_TRIANGLES, 576, 18);

// window DeepSkyBlue4 (0 104 139)

initReflection(0.0, 0.40784314, 0.54509807, 0.7, 0.7, 0.7, 1.0, 1.0, 1.0);

for (int i = 0; i < 360; i++) {

glUniformMatrix4fv(model, 1, GL\_TRUE, rocketCtm \* RotateZ(45 \* factor) \* RotateY(-30 + rocketRotY \* factor) \* Translate(0.0, 3.0, 2.65) \* RotateX(90) \* RotateY(i \* 1.0) \* Scale(1.0, 1.0, 1.0));

glDrawArrays(GL\_TRIANGLES, 36, 36);

}

}

## 2.3 Kỹ thuật tạo mô hình trống

void drawDrum(mat4 drumCtm) {

// surface LightYellow1 (255 255 224)

initReflection(1.0, 1.0, 0.8784314, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0);

for (int i = 0; i < 360; i++) {

glUniformMatrix4fv(model, 1, GL\_TRUE, drumCtm \* Translate(0.0, 0.4, 0.0) \* RotateY(i \* 1.0) \* Scale(4.0, 0.8, 4.0));

glDrawArrays(GL\_TRIANGLES, 0, 36);

}

// rope Tan (210 180 140)

initReflection(0.8235294, 0.7058824, 0.54901963, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0);

for (int i = 0; i < 360; i++) {

glUniformMatrix4fv(model, 1, GL\_TRUE, drumCtm \* Translate(0.0, 0.3, 0.0) \* RotateY(i \* 1.0) \* Scale(4.1, 0.2, 4.1));

glDrawArrays(GL\_TRIANGLES, 36, 36);

}

// body Turquoise2(0 229 238)

initReflection(0.0, 0.8980392, 0.93333334, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0);

for (int i = 0; i < 360; i++) {

glUniformMatrix4fv(model, 1, GL\_TRUE, drumCtm \* Translate(0.0, -0.9, 0.0) \* RotateY(i \* 1.0) \* Scale(4.0, 1.8, 4.0));

glDrawArrays(GL\_TRIANGLES, 72, 36);

}

// drumstick AntiqueGold (221 196 136)

initReflection(0.8666667, 0.76862746, 0.53333336, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0);

glUniformMatrix4fv(model, 1, GL\_TRUE, drumCtm \* Translate(0.0, 0.95, 1.0) \* Scale(3.0, 0.25, 0.25));

glDrawArrays(GL\_TRIANGLES, 36, 36);

initReflection(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0);

glUniformMatrix4fv(model, 1, GL\_TRUE, drumCtm \* Translate(1.375, 0.95, 1.0) \* Scale(0.5, 0.5, 0.5)); // mau den

glDrawArrays(GL\_TRIANGLES, 72, 36);

initReflection(0.8666667, 0.76862746, 0.53333336, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0);

glUniformMatrix4fv(model, 1, GL\_TRUE, drumCtm \* Translate(0.0, 0.95, -1.0) \* Scale(3.0, 0.25, 0.25));

glDrawArrays(GL\_TRIANGLES, 36, 36);

initReflection(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0);

glUniformMatrix4fv(model, 1, GL\_TRUE, drumCtm \* Translate(1.375, 0.95, -1.0) \* Scale(0.5, 0.5, 0.5)); // mau den

glDrawArrays(GL\_TRIANGLES, 72, 36);

}

## 2.4 Kỹ thuật tạo mô hình robot

// robot

void drawRobotHead(mat4 tranCtm) {

// head DarkSlateBlue (72 61 139)

initReflection(0.28235295, 0.23921569, 0.54509807, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0);

for (int i = 0; i < 360; i++) {

glUniformMatrix4fv(model, 1, GL\_TRUE, tranCtm \* RotateY(i \* 1.0) \* Scale(1.0, 1.0, 1.0));

glDrawArrays(GL\_TRIANGLES, 612, 108);

}

// Scarves Tomato3 (205 79 57)

initReflection(0.8039216, 0.30980393, 0.22352941, 0.1, 0.2, 0.2, 0.15, 0.2, 0.1);

glUniformMatrix4fv(model, 1, GL\_TRUE, tranCtm \* Translate(0.0, -0.1, 0.0) \* RotateX(90) \* Scale(1.0, 1.0, 1.0));

glDrawArrays(GL\_TRIANGLES, 612, 108);

glUniformMatrix4fv(model, 1, GL\_TRUE, tranCtm \* Translate(0.0, -0.1, 0.0) \* RotateX(90) \* RotateZ(90) \* Scale(1.0, 1.0, 1.0));

glDrawArrays(GL\_TRIANGLES, 612, 108);

glUniformMatrix4fv(model, 1, GL\_TRUE, tranCtm \* Translate(0.0, -0.1, 0.0) \* RotateX(90) \* RotateZ(180) \* Scale(1.0, 1.0, 1.0));

glDrawArrays(GL\_TRIANGLES, 612, 108);

glUniformMatrix4fv(model, 1, GL\_TRUE, tranCtm \* Translate(0.0, -0.1, 0.0) \* RotateX(90) \* RotateZ(270) \* Scale(1.0, 1.0, 1.0));

glDrawArrays(GL\_TRIANGLES, 612, 108);

// eye Coral1 (255 114 86)

initReflection(1.0, 0.44705883, 0.3372549, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0);

glUniformMatrix4fv(model, 1, GL\_TRUE, tranCtm \* Translate(0.0, 0.4, 0.4) \* RotateX(90) \* Scale(0.6, 0.6, 1.0));

glDrawArrays(GL\_TRIANGLES, 612, 108);

glUniformMatrix4fv(model, 1, GL\_TRUE, tranCtm \* Translate(0.0, 0.4, 0.4) \* RotateX(90) \* RotateZ(90) \* Scale(0.6, 0.6, 1.0));

glDrawArrays(GL\_TRIANGLES, 612, 108);

}

void drawRobotBody(mat4 tranBodyCtm) {

// Dark Cyan (0 128 128)

initReflection(0.0, 0.5019608, 0.5019608, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1);

glUniformMatrix4fv(model, 1, GL\_TRUE, tranBodyCtm \* Scale(4.0, 4.0, 4.0));

glDrawArrays(GL\_TRIANGLES, 720, 48);

initReflection(1.0, 0.44705883, 0.3372549, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0);

for (int i = 0; i < 360; i++) {

glUniformMatrix4fv(model, 1, GL\_TRUE, tranBodyCtm \* Translate(-1.0, 1.0, 1.9) \* RotateZ(i \* 1.0) \* Scale(0.5, 0.5, 0.5));

glDrawArrays(GL\_TRIANGLES, 0, 36);

}

}

void drawRobotArm(mat4 tranArmCtm) {

initReflection(1.0, 0.44705883, 0.3372549, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0);

glUniformMatrix4fv(model, 1, GL\_TRUE, tranArmCtm \* Translate(-0.15, -1.9, 0.0) \* Scale(1.0, 1.0, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 0, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, tranArmCtm \* Translate(-0.15, -0.9, 0.0) \* Scale(1.0, 1.0, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 0, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, tranArmCtm \* Translate(-0.13, 0.07, 0.0) \* RotateZ(-3) \* Scale(1.0, 1.0, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 0, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, tranArmCtm \* Translate(-0.05, 1.0, 0.0) \* RotateZ(-5) \* Scale(1.0, 1.0, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 0, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, tranArmCtm \* Translate(0.06, 1.95, 0.0) \* RotateZ(-7) \* Scale(1.0, 1.0, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 0, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, tranArmCtm \* Translate(0.25, 2.9, 0.0) \* RotateZ(-15) \* Scale(1.0, 1.0, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 0, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, tranArmCtm \* Translate(0.6, 3.85, 0.0) \* RotateZ(-25) \* Scale(1.0, 1.0, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 0, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, tranArmCtm \* Translate(1.1, 4.6, 0.0) \* RotateZ(-40) \* Scale(1.0, 1.0, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 0, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, tranArmCtm \* Translate(1.8, 5.3, 0.0) \* RotateZ(-50) \* Scale(1.0, 1.0, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 0, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, tranArmCtm \* Translate(2.6, 5.8, 0.0) \* RotateZ(-65) \* Scale(1.0, 1.0, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 0, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, tranArmCtm \* Translate(3.4, 6.05, 0.0) \* RotateZ(-80) \* Scale(1.0, 1.0, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 0, POINTS\_PER\_CUBE);

}

void drawRobotFinger(mat4 tranFinger) {

initReflection(0.28235295, 0.23921569, 0.54509807, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0);

glUniformMatrix4fv(model, 1, GL\_TRUE, tranFinger \* Translate(-0.45, 0.25, 0.0) \* Scale(0.3, 0.5, 1.0));

glDrawArrays(GL\_TRIANGLES, 72, 36);

glUniformMatrix4fv(model, 1, GL\_TRUE, tranFinger \* Translate(-0.4, -0.1, 0.0) \* RotateZ(20) \* Scale(0.3, 0.3, 1.0));

glDrawArrays(GL\_TRIANGLES, 72, 36);

glUniformMatrix4fv(model, 1, GL\_TRUE, tranFinger \* Translate(-0.27, -0.23, 0.0) \* RotateZ(63) \* Scale(0.3, 0.3, 1.0));

glDrawArrays(GL\_TRIANGLES, 72, 36);

glUniformMatrix4fv(model, 1, GL\_TRUE, tranFinger \* Translate(-0.1, -0.28, 0.0) \* RotateZ(80) \* Scale(0.3, 0.3, 1.0));

glDrawArrays(GL\_TRIANGLES, 72, 36);

glUniformMatrix4fv(model, 1, GL\_TRUE, tranFinger \* RotateY(180) \* Translate(-0.45, 0.25, 0.0) \* Scale(0.3, 0.5, 1.0));

glDrawArrays(GL\_TRIANGLES, 72, 36);

glUniformMatrix4fv(model, 1, GL\_TRUE, tranFinger \* RotateY(180) \* Translate(-0.4, -0.1, 0.0) \* RotateZ(20) \* Scale(0.3, 0.3, 1.0));

glDrawArrays(GL\_TRIANGLES, 72, 36);

glUniformMatrix4fv(model, 1, GL\_TRUE, tranFinger \* RotateY(180) \* Translate(-0.27, -0.23, 0.0) \* RotateZ(63) \* Scale(0.3, 0.3, 1.0));

glDrawArrays(GL\_TRIANGLES, 72, 36);

glUniformMatrix4fv(model, 1, GL\_TRUE, tranFinger \* RotateY(180) \* Translate(-0.1, -0.28, 0.0) \* RotateZ(80) \* Scale(0.3, 0.3, 1.0));

glDrawArrays(GL\_TRIANGLES, 72, 36);

}

void drawRobotFoot(mat4 tranFoot) {

// leg

initReflection(0.28235295, 0.23921569, 0.54509807, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0);

for (int i = 0; i < 360; i++) {

glUniformMatrix4fv(model, 1, GL\_TRUE, tranFoot \* RotateY(i \* 1.0) \* Scale(1.0, 4.0, 1.0));

glDrawArrays(GL\_TRIANGLES, 72, 36);

}

// foot

initReflection(1.0, 0.44705883, 0.3372549, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0);

glUniformMatrix4fv(model, 1, GL\_TRUE, tranFoot \* Translate(0.0, -1.5, 0.0) \* Scale(2.0, 2.0, 2.0));

glDrawArrays(GL\_TRIANGLES, 576, 18);

}

void drawRobot(mat4 robotCtm) {

// head

drawRobotHead(robotCtm \* Translate(0.0, 4.35, 1.8) \* Scale(1.2, 1.2, 1.3));

// body

drawRobotBody(robotCtm \* Translate(0.0, 2.0, 2.0));

// arm

drawRobotArm(robotCtm \* Translate(-2.0, 3.0, 2.0) \* RotateZ(85) \* Scale(0.5, 0.5, 1.0));

drawRobotArm(robotCtm \* Translate(2.0, 3.0, 2.0) \* RotateY(180) \* RotateZ(85) \* Scale(0.5, 0.5, 1.0));

// fingers

drawRobotFinger(robotCtm \* Translate(-4.9, 5.5, 2.0));

drawRobotFinger(robotCtm \* Translate(4.9, 5.5, 2.0));

// foots

drawRobotFoot(robotCtm \* Translate(-1.1, -2.0, 2.0) \* Scale(0.6, 1.0, 0.6));

drawRobotFoot(robotCtm \* Translate(1.1, -2.0, 2.0) \* Scale(0.6, 1.0, 0.6));

}

## 2.5 Kỹ thuật tạo mô hình dao

//Sword

void drawSword(mat4 swordCtm) {

// handling component Orange2 (238 154 0)

initReflection(0.93333334, 0.6039216, 0.0, 0.7, 0.7, 0.7, 1.0, 1.0, 1.0);

glUniformMatrix4fv(model, 1, GL\_TRUE, swordCtm \* Translate(-0.024, -2.9, 0.0) \* Scale(1.3, 1.0, 0.6)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 12, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, swordCtm \* Translate(-0.024, -3.9, 0.0) \* Scale(1.3, 1.0, 0.6)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 12, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, swordCtm \* Translate(-0.024, -4.9, 0.0) \* Scale(1.3, 1.0, 0.6)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 12, POINTS\_PER\_CUBE);

// protector Bisque3 (205 183 158)

initReflection(0.8039216, 0.7176471, 0.61960787, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0);

glUniformMatrix4fv(model, 1, GL\_TRUE, swordCtm \* RotateY(180) \* Translate(1.05, -2.8, 0.0) \* RotateY(360) \* RotateZ(75) \* Scale(0.2, 1.0, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 12, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, swordCtm \* RotateY(180) \* Translate(1.7, -3.1, 0.0) \* RotateZ(45) \* Scale(0.2, 0.6, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 12, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, swordCtm \* RotateY(180) \* Translate(1.97, -3.45, 0.0) \* RotateZ(25) \* Scale(0.19, 0.4, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 12, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, swordCtm \* RotateY(180) \* Translate(1.05, -5.0, 0.0) \* RotateZ(-75) \* Scale(0.2, 1.0, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 12, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, swordCtm \* RotateY(180) \* Translate(1.7, -4.7, 0.0) \* RotateZ(-45) \* Scale(0.2, 0.6, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 12, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, swordCtm \* RotateY(180) \* Translate(1.97, -4.35, 0.0) \* RotateZ(-25) \* Scale(0.19, 0.4, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 12, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, swordCtm \* RotateY(180) \* Translate(2.05, -3.9, 0.0) \* Scale(0.19, 0.6, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 12, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, swordCtm \* Translate(1.0, -2.5, 0.0) \* Scale(1.0, 0.2, 0.2));

glDrawArrays(GL\_TRIANGLES, 36 \* 12, POINTS\_PER\_CUBE);

//

// blade Gray71 (181 181 181)

initReflection(0.70980394, 0.70980394, 0.70980394, 0.7, 0.7, 0.7, 1.0, 1.0, 1.0);

glUniformMatrix4fv(model, 1, GL\_TRUE, swordCtm \* Translate(-0.15, -1.9, 0.0) \* Scale(1.0, 1.0, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 0, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, swordCtm \* Translate(-0.15, -0.9, 0.0) \* Scale(1.0, 1.0, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 0, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, swordCtm \* Translate(-0.13, 0.07, 0.0) \* RotateZ(-3) \* Scale(1.0, 1.0, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 0, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, swordCtm \* Translate(-0.05, 1.0, 0.0) \* RotateZ(-5) \* Scale(1.0, 1.0, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 0, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, swordCtm \* Translate(0.06, 1.95, 0.0) \* RotateZ(-7) \* Scale(1.0, 1.0, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 0, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, swordCtm \* Translate(0.25, 2.9, 0.0) \* RotateZ(-15) \* Scale(1.0, 1.0, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 0, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, swordCtm \* Translate(0.6, 3.85, 0.0) \* RotateZ(-25) \* Scale(1.0, 1.0, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 0, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, swordCtm \* Translate(1.1, 4.6, 0.0) \* RotateZ(-40) \* Scale(1.0, 1.0, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 0, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, swordCtm \* Translate(1.8, 5.3, 0.0) \* RotateZ(-50) \* Scale(1.0, 1.0, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 0, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, swordCtm \* Translate(2.6, 5.8, 0.0) \* RotateZ(-65) \* Scale(1.0, 1.0, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 0, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, swordCtm \* Translate(3.4, 6.05, 0.0) \* RotateZ(-80) \* Scale(1.0, 1.0, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 0, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, swordCtm \* Translate(3.9, 6.12, 0.0) \* Scale(0.5, 1.0, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 0, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, swordCtm \* Translate(4.0, 5.63, 0.0) \* Scale(0.3, 1.4, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 0, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, swordCtm \* Translate(3.4, 5.25, 0.0) \* RotateZ(-70) \* Scale(1.0, 1.0, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 0, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, swordCtm \* Translate(2.7, 4.9, 0.0) \* RotateZ(-60) \* Scale(1.0, 1.0, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 0, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, swordCtm \* Translate(2.0, 4.35, 0.0) \* RotateZ(-40) \* Scale(1.0, 1.0, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 0, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, swordCtm \* Translate(1.55, 3.7, 0.0) \* RotateZ(-30) \* Scale(1.0, 1.0, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 0, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, swordCtm \* Translate(1.15, 3.0, 0.0) \* RotateZ(-25) \* Scale(1.0, 1.0, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 0, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, swordCtm \* Translate(0.75, 2.1, 0.0) \* RotateZ(-20) \* Scale(1.0, 1.0, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 0, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, swordCtm \* Translate(0.5, 1.3, 0.0) \* RotateZ(-15) \* Scale(1.0, 1.0, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 0, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, swordCtm \* Translate(0.35, 0.5, 0.0) \* RotateZ(-10) \* Scale(1.0, 1.0, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 0, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, swordCtm \* Translate(0.2, -0.5, 0.0) \* RotateZ(-7) \* Scale(1.0, 1.0, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 0, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, swordCtm \* Translate(0.1, -1.5, 0.0) \* RotateZ(-5) \* Scale(1.0, 1.0, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 0, POINTS\_PER\_CUBE);

glUniformMatrix4fv(model, 1, GL\_TRUE, swordCtm \* Translate(0.12, -1.9, 0.0) \* Scale(1.0, 1.0, 0.3)); // 0.43

glDrawArrays(GL\_TRIANGLES, 36 \* 0, POINTS\_PER\_CUBE);

}

## 2.6 Kỹ thuật tạo mô hình bàn thanh toán, máy tính

void drawComputer(mat4 tableCtm) {

// computer arm

initReflection(0.40784314, 0.5137255, 0.54509807, 0.4, 0.4, 0.4, 0.95, 0.95, 0.95); // lightblue4

for (int i = 0; i < 360; i++) {

glUniformMatrix4fv(model, 1, GL\_TRUE, tableCtm \* RotateY(i \* 1.0) \* Scale(0.7, 3.0, 0.7));

glDrawArrays(GL\_TRIANGLES, 0, 36);

}

initReflection(0.40784314, 0.5137255, 0.54509807, 0.4, 0.4, 0.4, 0.95, 0.95, 0.95); // lightblue4

for (int i = 0; i < 360; i++) {

glUniformMatrix4fv(model, 1, GL\_TRUE, tableCtm \* Translate(0.0, 1.8, -0.58) \* RotateX(-60) \* RotateY(i \* 1.0) \* Scale(0.7, 2.0, 0.7));

glDrawArrays(GL\_TRIANGLES, 0, 36);

}

initReflection(0.40784314, 0.5137255, 0.54509807, 0.4, 0.4, 0.4, 0.95, 0.95, 0.95); // lightblue4

for (int i = 0; i < 360; i++) {

glUniformMatrix4fv(model, 1, GL\_TRUE, tableCtm \* Translate(0.0, 2.3, -1.7) \* RotateX(90) \* RotateY(i \* 1.0) \* Scale(0.7, 1.0, 0.7));

glDrawArrays(GL\_TRIANGLES, 0, 36);

}

// screen

initReflection(0.0, 0.0, 0.0, 0.2, 0.2, 0.2, 0.4, 0.4, 0.4); // black

glUniformMatrix4fv(model, 1, GL\_TRUE, tableCtm \* Translate(0.0, 2.5, -2.3) \* Scale(4.0, 3.0, 0.25));

glDrawArrays(GL\_TRIANGLES, 0, 36);

}

void drawPaymentTable(mat4 tableCtm) {

// left foot

initReflection(0.54509807, 0.3529412, 0.16862746, 0.4, 0.4, 0.4, 0.7, 0.7, 0.7); // tan4

glUniformMatrix4fv(model, 1, GL\_TRUE, tableCtm \* Translate(-9.9, 0.0, 0.0) \* RotateY(0) \* Scale(0.25, 10.0, 10.0));

glDrawArrays(GL\_TRIANGLES, 0, 36);

// surface

initReflection(0.54509807, 0.3529412, 0.16862746, 0.4, 0.4, 0.4, 0.7, 0.7, 0.7); // tan4

glUniformMatrix4fv(model, 1, GL\_TRUE, tableCtm \* Translate(0.0, 4.875, 0.0) \* RotateY(0) \* Scale(20, 0.25, 10.0));

glDrawArrays(GL\_TRIANGLES, 0, 36);

// right foot

initReflection(0.54509807, 0.3529412, 0.16862746, 0.4, 0.4, 0.4, 0.7, 0.7, 0.7); // tan4

glUniformMatrix4fv(model, 1, GL\_TRUE, tableCtm \* Translate(9.9, 0.0, 0.0) \* RotateY(0) \* Scale(0.25, 10.0, 10.0));

glDrawArrays(GL\_TRIANGLES, 0, 36);

// front foot

initReflection(0.54509807, 0.3529412, 0.16862746, 0.4, 0.4, 0.4, 0.7, 0.7, 0.7); // tan4

glUniformMatrix4fv(model, 1, GL\_TRUE, tableCtm \* Translate(0.0, 0.0, 4.876) \* RotateY(0) \* Scale(20.0, 10.0, 0.25));

glDrawArrays(GL\_TRIANGLES, 0, 36);

drawComputer(tableCtm \* Translate(0.0, 6.5, 0.0));

}

## 2.7 Kỹ thuật điều khiển camera

void setupViewAndProjection() {

// camera set up

//vec4 eye = vec4(camXLoca, camYLoca, camZLoca, 1.0), at = vec4(0.0, 0.0, 0.0, 1.0), up = vec4(0.0, 1.0, 0.0, 1.0);

cameraCtm = LookAt(eye, at, up);

camChangingCtm = Translate(camXTran, camYTran, camZTran) \* RotateX(camXRot) \* RotateY(camYRot) \* RotateZ(camZRot);

glUniformMatrix4fv(camera, 1, GL\_TRUE, cameraCtm \* camChangingCtm);

// projector set up

projectorCtm = Frustum(leftLim, rightLim, botLim, topLim, nearLim, farLim);

glUniformMatrix4fv(projector, 1, GL\_TRUE, projectorCtm);

}

float centerDistance = 0.0;

float xCenterDistance = 0.0;

float beta = 0.0;

float delta = 0.0;

float zDistance = 0.0;

float xDistance = 0.0;

GLfloat side = 0.0;

void look(int direction) {

if (direction != 1 && direction != -1) return;

centerDistance = sqrt((eye.x - at.x) \* (eye.x - at.x) + (eye.z - at.z) \* (eye.z - at.z));

xCenterDistance = abs(at.x - eye.x);

beta = (side == 0.0) ? acos(xCenterDistance / centerDistance) \* 180 / 3.14159265 : side;

delta = beta + 5 \* direction;

zDistance = centerDistance \* sin(beta \* 3.14159265 / 180);

xDistance = centerDistance \* cos(delta \* 3.14159265 / 180);

at.x = eye.x - xDistance;

at.y = at.y;

at.z = eye.z - zDistance;

side = delta;

}

## 2.8 Kỹ thuật chiếu sáng

void initReflection(GLfloat r, GLfloat g, GLfloat b, GLfloat rDif, GLfloat gDif, GLfloat bDif, GLfloat rSpe, GLfloat gSpe, GLfloat bSpe) {

point4 light\_position(0.0, 36.0, 0.0, 0.0);

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

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

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

color4 material\_ambient(r, g, b, 1.0);

color4 material\_diffuse(rDif, gDif, bDif, 1.0);

color4 material\_specular(rSpe, gSpe, bSpe, 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);

}

# KẾT QUẢ

## 3.1 Mô hình ô tô







## 3.2 Mô hình tên lửa



## 3.3 Mô hình trống



## 3.4 Mô hình robot



## 3.5 Mô hình dao



## 3.6 Mô hình bàn thanh toán, máy tính



## 3.7 Mô hình nhân viên bán hàng



## 3.8 Mô hình cửa ra vào

