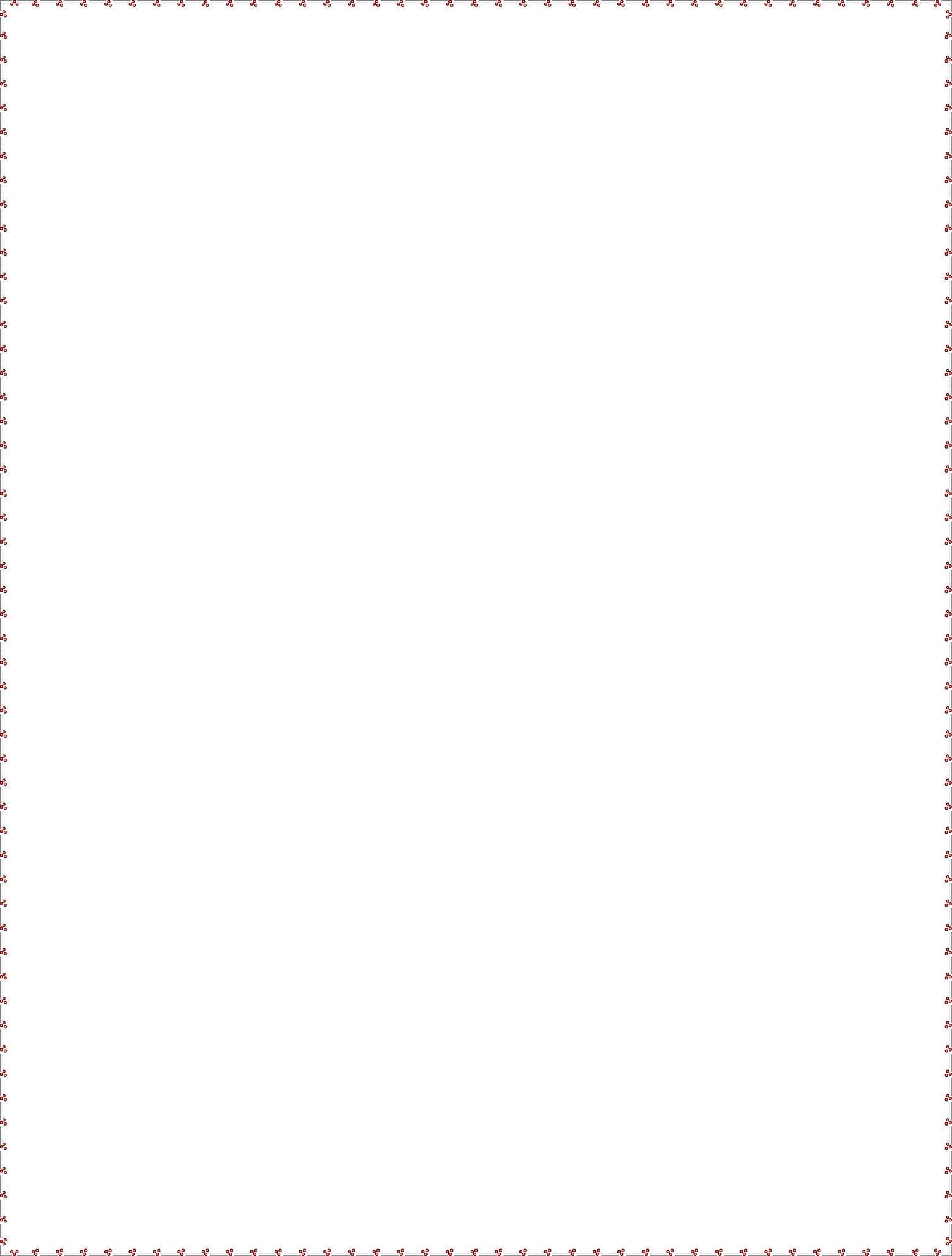
TRƯỜNG ĐẠI HỌC THỦY LỢI



**KHOA CÔNG NGHỆ THÔNG TIN**

**BÁO CÁO BÀI TẬP LỚN**

**ĐỀ TÀI**

**VẼ Ô TÔ LĂN TRÊN TRỤC NẰM NGHIÊNG**

*Giảng viên hướng dẫn :* ***Ngô Trường Giang***

*Nhóm sinh viên thực hiện :*

1. Trần Nhật Duật – 60TH4
2. Vũ Minh Thăng – 60TH4
3. Nguyễn Văn Luân – 60TH4

**LỜI NÓI ĐẦU**

1. **Giới thiệu.**

Trong những năm trở lại đây, với sự phát triển mạnh mẽ của công nghệ thông tin. Với việc hỗ trợ của công nghệ thông tin thì mọi công việc đều có thể giải quyết một cách dễ dạng và thuận tiện hơn.

Đồ họa máy tính là một trong các lĩnh vực mà ngành tin học đang rất quan tâm. Xuất phát từ nhu cầu đồ họa thực tế, nhóm chúng em đã học tập, phân tích, thực hành và làm hoàn thiện chương trình đề tài cô đưa ra: “Phát triển chương trình vẽ một cái ô tô lăn trên một trực nằm nghiêng, tô bóng và chiếu sáng sao cho gần giống với thế giới thực.

1. **Công nghệ sử dụng.**

Công cụ lập trình: Visual studio 2019.

Ngôn ngữ lập trình: C và thư viện đồ họa OpengGL (Open Graphics Library).

Hệ điều hành: Windown 10.

1. **Nội dung.**

Nội dung báo cáo gồm 4 chương:

Chương I: Tổng quan lý thuyết và ý tưởng chương trình

Chương này, trình bày tổng quan về kỹ thuật đồ họa, đối tượng đồ họa, một số kỹ thuật liên quan tới đối tượng đồ họa và ý tưởng của chương trình.

Chương II: Phân tích cấu trúc thuật toán

Chương này, trình bày về thuật toán các bước áp dụng để xây dựng đối tượng đồ họa giải quyết bài toán chương trình đề ra.

Chương III: Mã nguồn và giao diện chương trình

Chương này, hoàn thiện chương trình, đưa ra phân tích mã nguồn chương trình và giao diện kết quả hiện thị.

Chương IV: Kết luận

Chương này, trình bày nhưng kết quả đã được, chưa đạt được, định hướng phát triển trong tương lai và lời cảm ơn.

**CHƯƠNG I: TỔNG QUAN LÝ THUYẾT VÀ Ý TƯỞNG CHƯƠNG TRÌNH**

**1.1 Tổng quan lý thuyết.**

**1.1.1 Một số khái niệm.**

- Đồ họa máy tính là một lĩnh vực của [khoa học máy tính](https://vi.wikipedia.org/wiki/Khoa_h%E1%BB%8Dc_m%C3%A1y_t%C3%ADnh) nghiên cứu về cơ sở [toán học](https://vi.wikipedia.org/wiki/To%C3%A1n_h%E1%BB%8Dc), các [thuật toán](https://vi.wikipedia.org/wiki/Thu%E1%BA%ADt_to%C3%A1n) cũng như các kĩ thuật để cho phép tạo, hiển thị và điều khiển hình ảnh trên màn hình [máy tính](https://vi.wikipedia.org/wiki/M%C3%A1y_t%C3%ADnh)

- OpenGl (*Open Graphics Library*) là một thư viện lập trình đồ hoaj3D, tiêu chuẩn kỹ thuật [đồ họa](https://vi.wikipedia.org/wiki/%C4%90%E1%BB%93_h%E1%BB%8Da) có mục đích định ra một [giao diện lập trình ứng dụng](https://vi.wikipedia.org/wiki/Giao_di%E1%BB%87n_l%E1%BA%ADp_tr%C3%ACnh_%E1%BB%A9ng_d%E1%BB%A5ng) [đồ họa 3 chiều](https://vi.wikipedia.org/wiki/%C4%90%E1%BB%93_h%E1%BB%8Da_3_chi%E1%BB%81u). OpenGL cũng có thể được dùng trong các ứng dụng [đồ họa 2 chiều](https://vi.wikipedia.org/w/index.php?title=%C4%90%E1%BB%93_h%E1%BB%8Da_2_chi%E1%BB%81u&action=edit&redlink=1). Giao diện lập trình này chứa khoảng 250 [hàm](https://vi.wikipedia.org/wiki/H%C3%A0m) để vẽ các cảnh phức tạp từ những hàm đơn giản. Nó được dùng rộng rãi trong các [,](https://vi.wikipedia.org/wiki/Tr%C3%B2_ch%C6%A1i_%C4%91i%E1%BB%87n_t%E1%BB%AD) CAD, thực tế ảo. Đối thủ DirectX của Microsoft.

**1.1.2 Một số kỹ thuật liên quan tới đối tượng.**

**- Biến đổi điểm nhìn:** Là kỹ thuật chọn ví trí, hướng nhìn của mắt, camera, đồ vật quan sát tới điểm nhìn của đồ vật đươc cần được quan sát.

**+** Quy trình hiển thị là biến đổi đối tượng từ không gian đối tượng vào không gian chung gọi là không gian thực. OpenGL sử dụng lệnh gluLookAt để xác định vị trí và hướng của điểm nhìn.

• Cú pháp:

gluLookAt (GLdouble eyex, GLdouble eyey, GLdouble eyez, GLdouble centerx, GLdouble centery, GLdouble centerz, GLdouble upx, GLdouble upy, GLdouble upz);

• Ví trí trong chương trình: Được gọi trong hàm display hoặc hàm main

• Ví dụ:

gluLookAt (5.0, 5.0, 5.0, .0.0, 0.0, 0.0, 0.0, 3.0, .0);

- Giải thích: Ví trí của mắt tại tọa độ (5, 5, 5) trong không gian 3D, hướng nhìn tại tâm tọa độ (0, 0, 0), hướng đỉnh trên trục Oy có tọa độ (0, 3, 0).

**+ Biển đổi mô hình:** Là kỹ thuật xác định vị trí và hướng của mô hình của đối tượng trong không gian từ đó tạo nên mô hình đồ họa theo ý muốn.

Tiến trình đi từ điểm trong không gian thế giới thực đến pixel trên màn hình. Thực hiện chế độ biến đổi mô hình: glMatrixMode (GL\_MODELVIEW);

• Cú pháp các hàm biến đổi cơ bản:

+ Phép tịnh tiến: glTranslate {fd} (TYPE x, TYPE y, TYPE z).

+ Phép quay quanh trục nối gốc tọa độ với điển (x, y, z).: glRotate{fd}(TYPE x, TYPE y, TYPE z).

+ Phép co dãn: glScale {fd} (TYPE x, TYPE y, TYPE z).

• Ví dụ:

glTranslatef (0.0, 0.0, -0.6);

- Giải thích: Phép tịnh tiến đối tượng di chuyển về chiều âm của trục Oz tọa độ (0, 0, -0.6).

glRotatef (90, 0.0, 1.0, .0);

- Giải thích: Phép quay một góc 90 quang trục Ox có tọa độ (0, 1, 0).

glScalef (20, 0.0, 3.0);

- Giải thích: Phép co dãn tại trục x và trục z ví trí có tọa độ (20, 0, 30).

**- Tương tác hoạt cảnh:** Là kỹ thuật mô tả quá trình dịch chuyển của đối tượng đồ họa.

**+** Cho phép người dùng tường tác với đối tượng bằng cách sử dụng các thiết bị liên kết ngoài với bộ xử lý trong của chương trình như thao tác với chuột, bàn phím.

• Cú pháp:

+ Thao tác với hoạt cảnh bàn phím:

void keyboard(unsigned char key, int x, int y)

{ switch (key) {

case 'ten\_keyboard':

//Thao tác xử lý được gọi ở đây

glutPostRedisplay();

break;

default:

break;} }

-> Hàm được bắt xử lý sự kiện bàn phím được gọi trong hàm main, trong hàm display gọi sự kiện: glutSwapBuffers();

glutKeyboardFunc(keyboard);

• Ví dụ:

void keyboard(unsigned char key, int x, int y)

{ switch (key) {

case 'm':

dichchuyen = (dichchuyen + 1) / 1.1;

glutPostRedisplay();

break;

case 'n':

dichchuyen = (dichchuyen - 1)\*1.1;

glutPostRedisplay();

break;

default:

break;} }

**- Ngăn xếp ma trận:** Là kỹ thuật kết hợp với kỹ thuật biến đổi điểm nhìn và biến đổi mô hình tạo nên một trồng ngăn xếp ma trận, để xây dựng các mô hình phân cấp các đối tượng phức tập được xây dựng từ các mô hình đơn giản.

• Cú pháp:

glPushMatrix và glPopMatrix được sử dụng để sao chép thêm một ma trận hiện thời đưa lên đỉnh ngăn xếp và loại bỏ ma trận hiện thời ra khỏi ngăn xếp.

• Ví dụ:

glPushMatrix();

glRotatef(52, 0.0, 1.0, 1.0);

glScalef(20, 0.0, 3.0);

glutWireCube(1.0);

glPopMatrix();

**- Chiếu sáng đối tượng:** Là kỹ thuật thao tác với các màu của ánh sáng và các vật thể, loại bỏ các bề mặt ẩn, thiết lập các thông số, phân tích các nguồn ánh sáng ảnh hưởng để tạo nên các hiệu ứng màu của đối tượng.

**+** OpenGL có 4 loại nguồn sáng:

+ Môi trường: ánh sáng được chiếu sáng đầy đủ.

+ Khuếch tán: ánh sáng đến từ một phía, chiếu vào bề mặt đối tượng làm cho bề mặt này trở lên sáng chói hơn, sau đó ánh sáng được khuếch tán đi mọi hướng.

+ Phản chiếu: Ánh sáng tạo đốm phản chói, thường là màu trắng trên các bề mặt có tính phản chiếu cao.

+ Nguồn sáng: Nguồn sáng phát ra từ đối tượng như bóng đèn.

• Cú pháp:

+ Thiết lập các mảng giá trị chiếu sáng.

void setMaterial(GLfloat ambientR, GLfloat ambientG, GLfloat ambientB,

GLfloat diffuseR, GLfloat diffuseG, GLfloat diffuseB,

GLfloat specularR, GLfloat specularG, GLfloat specularB, GLfloat positionR, GLfloat positionG, GLfloat positionB)

{

GLfloat ambient[] = { ambientR, ambientG, ambientB }; //Ánh sáng môi trường.

GLfloat diffuse[] = { diffuseR, diffuseG, diffuseB }; //Ánh sáng khuếch tán.

GLfloat specular[] = { specularR, specularG, specularB }; //Ánh sáng phản chiếu.

GLfloat positionlight[] = { positionR, positionG, positionB }; //Ví trí chiếu sáng.

}

+ Định nghĩa nguồn sáng:

glMaterialfv(GL\_FRONT, GL\_AMBIENT, ambient);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, diffuse);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, specular);

 + Kích hoạt nguồn sáng:

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

**1.2 Ý tưởng thực hiện chương trình.**

Đề tài: “ Phát triển chương trình vẽ một cái ô tô lăn trên một trục nằm nghiêng, tô bóng và chiếu sáng đối tượng sao cho giống thế giới thực.”.

**⮚ Ý tưởng:** Từ những mô hình hình ảnh thực tế oto mà ta quan sát được từ thế giới thực và sự hộ trợ của OpenGL ta tiến hành:

Bước 1: Cần lựa chọn các mô hình mà đồ họa đã hỗ trợ sẵn để tiến hành xử lý tạo được đối tượng mô hình.

Bước 2: Thực hiện biến đổi đổi vị trí điểm nhìn trong không gian 3D sao cho nhìn thấy đối tượng hiển thị kích thước theo ý muốn.

Bước 3: Sử dụng các phép dịch chuyển xoay tịnh tiến sao cho mô hình động và thành hình khối nằm trên trục nằm nghiêng.

Bước 4: Sử dụng ngăn xếp ma trận với sự kết hợp các phép dịch chuyển xoay, tịnh tiến sao cho mô hình động và các khối liên kết được tạo thành đối tượng mong muốn

Bước 5: Thao tác sự kiện phía người dùng, sử dụng các kỹ thuật bắt sự kiện tương tác với chuột và bàn phím để thao tác động với đối tượng.

Bước 6: To bóng, chiếu sáng các nguồn sáng phù hợp để đạt được đối tượng rất giống với đối tượng được đặt trong thế giới thực có điều kiện chiếu sáng nhiều màu sắc.

**CHƯƠNG II: PHÂN TÍCH CẤU TRÚC THUẬT TOÁN**

**2.1 Vẽ đường đi của xe.**

setMaterial(0.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 50);

glPushMatrix();

glRotatef(52, 0.0, 1.0, 1.0);

glScalef(20, 0.0, 3.0);

glutWireCube(1.0);

glPopMatrix**();**

**-** Giải thích: Đường đi của xe được chiếu sáng ánh sáng môi trường màu xanh, ánh sáng khuếch tán màu trắng và sáng phản chiếu màu vàng. Sử dụng kỹ thuật ngăn xếp ma trận, sự kết hợp biến đổi mô mình phép quay (một góc 52 quanh tại vị trí tọa độ (0, 1, 1)), phép co dãn tại vị trí tọa độ (20, 0, 3). Sử dụng mô hình opengl glWireCube kích thước (1.0).

**2.2 Vẽ thân trên của xe.**

glPushMatrix();

glTranslatef((GLfloat)dichchuyen / 6, (GLfloat)dichchuyen / 6, 0);

setMaterial(0.0, 0.0, 0.0, 0.6, 0.6, 0.0, 1.0, 1.0, 1.0, 50);

glTranslatef(0.55, 0.75, 0.5);

glScalef(1.0, 0.5, 1.5);

glutSolidCube(.35);

glPopMatrix();

- Giải thích: Sử dụng kỹ thuật ngăn xếp ma trận, phép tịnh tiến tạo sự di chuyển tại tọa độ (0.6, 0.6, 0). Kỹ thuật chiếu sáng ánh sáng môi trường màu đen, ánh sáng khuếch tán màu vàng và ánh sáng phản xạ màu trắng. Thao tác tinh tiến tại tọa độ (0.55, 0.75, 0.5). Phép co dãn tại vị trí (1, 0.5, 1.5). Sử dụng mô hình opengl có kích thước (0.35).

**2.3 Vẽ thân dưới của xe.**

glPushMatrix();

setMaterial(0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 50);

glTranslatef((GLfloat)dichchuyen / 5.3, (GLfloat)dichchuyen / 5.3, 0);

glScalef(1.0, 0.5, 2.0);

glutSolidCube(0.5);

glPopMatrix();

- Giải thích: Sử dụng kỹ thuật ngăn xếp ma trận, kỹ thuật chiếu sáng đối tượng với ánh sáng môi trường có màu đen, ánh sáng khuếch tán màu trắng và ánh snasg phản xạ màu vàng. Kỹ thuật biến đổi mô hình, phép tịnh tiến, phép co dẽn và mô hình hỗ trợ của opengl.

**2.4 Vẽ bốn bánh xe.**

glPushMatrix();

setMaterial(1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 50);

glTranslatef((GLfloat)dichchuyen / 5, (GLfloat)dichchuyen / 5, 0);

+ Bánh trước 1:

glRotatef(90, 0.0, 1.0, 0.0);

glTranslatef(0.1, -0.4, 0.0);

glRotatef(spin, 0.0, 0.0, 1.0);

glutSolidTorus(0.05, .1, 9.0, 9.0);

+ Bánh trước 2:

glRotatef(90, 0.0, 0.0, 1.0);

glTranslatef(0.0, 0.0, -0.3);

glRotatef(spin, 0.0, 0.0, 1.0);

glTranslatef(0.0, 0.0, -0.1);

glutSolidTorus(0.05, 0.1, 9.0, 9.0);

glPopMatrix();

- Giải thích: Sử dụng kỹ thuật ngăn xếp ma trận, kỹ thuật chiếu snasg, ánh sáng môi trường màu trắng, ánh sáng khuếch tán màu đen và ánh sáng phản xạ màu trắng. Đối với bánh trước 1. Phép quay một góc 90 độ, tại vị trí (0, 1, 0). Phép tịnh tiến có tọa độ ( 0.1, 0.4, 0.0). Phép quay một góc spin tại vị trí có tọa độ (0, 0, 1). Kỹ thuật đối với bánh trước 2 và bánh sau tương tự.

**CHƯƠNG III: MÃ NGUỒN VÀ GIAO DIỆN CHƯƠNG TRÌNH**

**3.1 Mã nguồn chương trình**

#include <GL/glut.h>

static int dichchuyen = 0;

float spin = 0.0f;

//Hàm Xử lý tốc độ quay của bánh xe.

void spinDisplay(void) {

spin = spin + 1;

if (spin > 360.0)

spin = spin - 360.0;

glutPostRedisplay();

}

//Hàm khai bao thao tác xử lý chiếu sáng: ánh sáng bao quanh, ánh sáng khuếch tán, ánh sáng phản chiếu Red, Green, Blue.

void setMaterial(GLfloat ambientR, GLfloat ambientG, GLfloat ambientB,

GLfloat diffuseR, GLfloat diffuseG, GLfloat diffuseB,

GLfloat specularR, GLfloat specularG, GLfloat specularB,

GLfloat shininess)

{

GLfloat ambient[] = { ambientR, ambientG, ambientB }; //Ánh sáng môi trường.

GLfloat diffuse[] = { diffuseR, diffuseG, diffuseB }; //Ánh sáng khuếch tán.

GLfloat specular[] = { specularR, specularG, specularB }; //Ánh sáng phản chiếu.

glMaterialfv(GL\_FRONT, GL\_AMBIENT, ambient);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, diffuse);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, specular);

glMaterialf(GL\_FRONT, GL\_SHININESS, shininess);

}

void display(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glMatrixMode(GL\_MODELVIEW);

//Thao tác điều khiển ngăn xếp ma trận vẽ thanh nghiên giống đường đi.

glColor3f(1.0, 1.0, 0.0);

setMaterial(0.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 50);

glPushMatrix();

//Độ nghiêng

glRotatef(52, 0.0, 1.0, 1.0);

glScalef(20, 0.0, 3.0);

glutWireCube(1.0);

glPopMatrix();

//Thao tác ma trận vẽ ô tô biến đổi điểm nhìn và biến đổi mô hình.

glPushMatrix();

glPushMatrix();

setMaterial(0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 50);

//Vẽ ngăn xếp trên thân dưới hình

glTranslatef((GLfloat)dichchuyen / 5.3, (GLfloat)dichchuyen / 5.3, 0);

glScalef(1.0, 0.5, 2.0);

glutSolidCube(0.5);

glPopMatrix();

//Vẽ ngăn xếp thân trên hình.

glPushMatrix();

glTranslatef((GLfloat)dichchuyen / 6, (GLfloat)dichchuyen / 6, 0);

setMaterial(0.0, 0.0, 0.0, 0.6, 0.6, 0.0, 1.0, 1.0, 1.0, 50);

glTranslatef(0.55, 0.75, 0.5);

glScalef(1.0, 0.5, 1.5);

glutSolidCube(0.35);

glPopMatrix();

glTranslatef(0.0, 0.0, 0.25);

glPushMatrix();

// Thao tác ma trận thao tác vẽ bánh xe và chuyển động.

setMaterial(1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 50);

glTranslatef((GLfloat)dichchuyen / 5, (GLfloat)dichchuyen / 5, 0);

//Bánh trước 1.

glRotatef(90, 0.0, 1.0, 0.0);

glTranslatef(0.1, -0.4, 0.0);

glRotatef(spin, 0.0, 0.0, 1.0);

glutSolidTorus(0.05, .1, 9.0, 9.0);

//Bánh trước 2.

glRotatef(90, 0.0, 0.0, 1.0);

glTranslatef(0.0, 0.0, -0.3);

glRotatef(spin, 0.0, 0.0, 1.0);

glTranslatef(0.0, 0.0, -0.1);

glutSolidTorus(0.05, 0.1, 9.0, 9.0);

glPopMatrix();

glTranslatef(0.0,0.0, -0.6);

glPushMatrix();

glTranslatef((GLfloat)dichchuyen / 5, (GLfloat)dichchuyen / 5, 0);

setMaterial(1.0, 1.0, 1.0, 0.0, 0.0, .0, 1.0, 1.0, 1.0, 50);

//Bánh sau 1.

glRotatef(90, 0.0, 1.0, 0.0);

glTranslatef(0.2, -0.45, 0.0);

glRotatef(spin, 0.0, 0.0, 1.0);

glutSolidTorus(.05, .1, 9.0, 9.0);

//Bánh sau 2.

glRotatef(90, 0.0, 0.0, 1.0);

glTranslatef(0.0, 0.007, -0.2);

glRotatef(spin, 0.0, 0.0, 1.0);

glutSolidTorus(0.05, .1, 9.0, 9.0);

glPopMatrix();

glPopMatrix();

glFlush();

glutSwapBuffers();

}

void reshape(int width, int height)

{

glViewport(0, 0, width, height);

}

//Hàm sử lý sự kiện hoạt cảnh bàn phím

void keyboard(unsigned char key, int x, int y)

{

switch (key) {

case 'm': //Xe di chuyển xuống dốc.

dichchuyen = (dichchuyen + 1) / 1.1;

glutPostRedisplay();

break;

case 'n': //Xe di chuyển lên dốc.

dichchuyen = (dichchuyen - 1)\*1.1;

glutPostRedisplay();

break;

default:

break;

}

}

//Hàm xử lý sự kiện thao tác với chuột

void MouseButton(int type\_button, int state, GLint x, GLint y)

{

switch (type\_button)

{

case GLUT\_LEFT\_BUTTON: //click chuột trái bánh xe chuyển động quay.

if (state == GLUT\_DOWN)

glutIdleFunc(spinDisplay);

break;

default:

break;

}

}

void MouseMove(int x, int y) {

int a = 5;

}

void init(void)

{

glClearColor(0.0, 0.0, 1.0, 1.0);

glShadeModel(GL\_FLAT);

}

int main(int argc, char\*\* argv)

{

glutInit(&argc, argv);

init();

glutInitWindowSize(500, 500);

glutInitWindowPosition(0, 0);

glutInitDisplayMode(GLUT\_RGB | GLUT\_DEPTH);

glutCreateWindow("DHMT");

glutDisplayFunc(display);

glutReshapeFunc(reshape);

glutMouseFunc(MouseButton);

glEnable(GL\_DEPTH\_TEST);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(40, 1, 4, 20);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

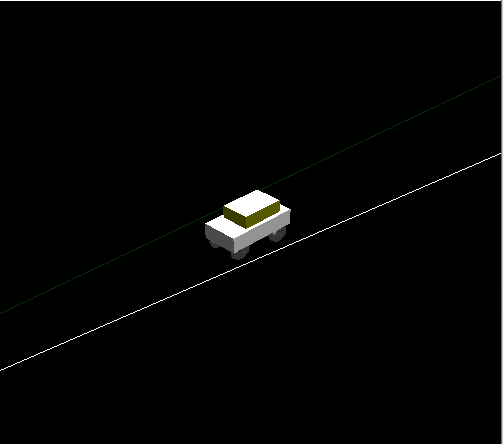
gluLookAt(5.0, 5.0, 5.0, 0.0, 0.0, 0.0, 0.0, 3.0, 0.0);

glutKeyboardFunc(keyboard);

glutMainLoop();

}

**3.2 Giao diện chương trình**



TÀI LIỆU THAM KHẢO

<https://www.slideshare.net/luongnv89/open-gl-introduction-27036246>

<https://www.coursehero.com/file/p3nssff/3-Ph%C3%A9p-bi%E1%BA%BFn-%C4%91%E1%BB%95i-%C4%91i%E1%BB%83m-nh%C3%ACn-v%C3%A0-m%C3%B4-h%C3%ACnh-Viewing-v%C3%A0-Modeling-Transformations-OpenGl/>