

HƯỚNG DẪN THỰC HÀNH - OPENGL

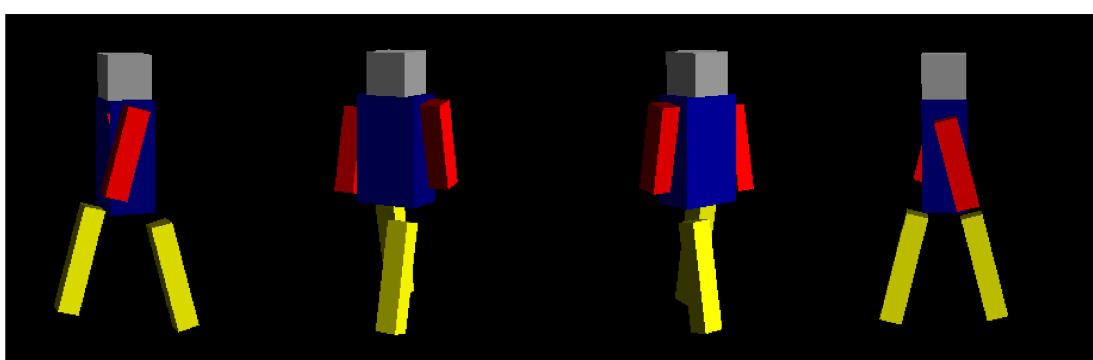
PHÉP BIẾN ĐỔI HÌNH HỌC

Mục tiêu:

- Sử dụng các hàm biến đổi hình học cơ bản của OpenGL: `glTranslatef`, `glScalef`, `glRotatef`.
- Thực hiện phép biến đổi hình học trên đối tượng phức (được lắp ghép từ các đối tượng cơ bản). Sử dụng các hàm thao tác với ma trận trạng thái biến đổi: `glPushMatrix`, `glPopMatrix`.
- Thực hiện phép biến đổi camera view, sử dụng hàm thư viện `gluLookAt`.
- Sử dụng mô hình chiếu sáng cơ bản.

1. Bài TH 01:

- Yêu cầu:
 - o Vẽ robot từ các hình lập phương: đầu, thân, hai tay, hai chân.
 - o Sử dụng các phép biến đổi để mô phỏng robot bước đi tại chỗ và tự xoay quanh (xem hình bên dưới).



- Hướng dẫn thực hiện:
 - o Các biến toàn cục cần khai báo:

```
// Hằng số ước tính trạng thái di chuyển của tay và chân
const char BACKWARD_STATE = 0;
const char FORWARD_STATE = 1;

// Index ước cho mảng (góc xoay của tay và chân)
const char LEFT = 0;
const char RIGHT = 1;
```

```

/* Trạng thái di chuyển hiện tại của tay và chân (BACKWARD_STATE/
FORWARD_STATE)*/
char legStates[2];
char armStates[2];

// Góc xoay hiện tại của tay và chân
float legAngles[2];
float armAngles[2];

// Góc xoay hiện tại của toàn bộ robot
float angle;

```

- Khai báo hàm:

```

void DrawCube(float xPos, float yPos, float zPos);
void DrawArm(float xPos, float yPos, float zPos);
void DrawHead(float xPos, float yPos, float zPos);
void DrawTorso(float xPos, float yPos, float zPos);
void DrawLeg(float xPos, float yPos, float zPos);
void DrawRobot(float xPos, float yPos, float zPos);
void Prepare();
void Display();
void Init();
void Reshape(int Width, int Height);
void Idle();

```

- Hàm **DrawCube** để vẽ hình lập phương tại vị trí chỉ định.

```

void DrawCube(float xPos, float yPos, float zPos)
{
    glPushMatrix(); /* Lưu trạng thái biến đổi hiện tại vào stack */
    glTranslatef(xPos, yPos, zPos);
    glBegin(GL_QUADS);
        // Vẽ mặt trên
        glNormal3d(0, 1, 0);
        glVertex3f(0.0f, 0.0f, 0.0f);
        glVertex3f(0.0f, 0.0f, -1.0f);
        glVertex3f(-1.0f, 0.0f, -1.0f);
        glVertex3f(-1.0f, 0.0f, 0.0f);
        // Vẽ mặt trước
        glNormal3d(0, 0, 1);
        glVertex3f(0.0f, 0.0f, 0.0f);
        glVertex3f(-1.0f, 0.0f, 0.0f);
        glVertex3f(-1.0f, -1.0f, 0.0f);
        glVertex3f(0.0f, -1.0f, 0.0f);
        // Vẽ mặt bên phải
        glNormal3d(1, 0, 0);
        glVertex3f(0.0f, 0.0f, 0.0f);
        glVertex3f(0.0f, -1.0f, 0.0f);

```

```

        glVertex3f(0.0f, -1.0f, -1.0f);
        glVertex3f(0.0f, 0.0f, -1.0f);
        // Vẽ mặt bên trái
        glNormal3d(-1, 0, 0);
        glVertex3f(-1.0f, 0.0f, 0.0f);
        glVertex3f(-1.0f, 0.0f, -1.0f);
        glVertex3f(-1.0f, -1.0f, -1.0f);
        glVertex3f(-1.0f, -1.0f, 0.0f);
        // Vẽ mặt dưới
        glNormal3d(0, -1, 0);
        glVertex3f(0.0f, 0.0f, 0.0f);
        glVertex3f(0.0f, -1.0f, -1.0f);
        glVertex3f(-1.0f, -1.0f, -1.0f);
        glVertex3f(-1.0f, -1.0f, 0.0f);
        // Vẽ mặt sau
        glNormal3d(0, 0, -1);
        glVertex3f(0.0f, 0.0f, 0.0f);
        glVertex3f(-1.0f, 0.0f, -1.0f);
        glVertex3f(-1.0f, -1.0f, -1.0f);
        glVertex3f(0.0f, -1.0f, -1.0f);
    glEnd();
    glPopMatrix(); // Quay lại trạng thái biến đổi đã lưu.
}

```

- Hàm **DrawArm** để vẽ cánh tay (hình hộp chữ nhật) tại vị trí chỉ định.

```

void DrawArm(float xPos, float yPos, float zPos)
{
    glPushMatrix();
        glColor3f(1.0f, 0.0f, 0.0f);      // Tô màu đỏ
        glTranslatef(xPos, yPos, zPos);
        glScalef(1.0f, 4.0f, 1.0f);      // Kích thước 1x4x1
        DrawCube(0.0f, 0.0f, 0.0f);
    glPopMatrix();
}

```

- Hàm **DrawHead** để vẽ đầu (hình hộp chữ nhật) tại vị trí chỉ định.

```

void DrawHead(float xPos, float yPos, float zPos)
{
    glPushMatrix();
        glColor3f(1.0f, 1.0f, 1.0f);      // Tô màu trắng
        glTranslatef(xPos, yPos, zPos);
        glScalef(2.0f, 2.0f, 2.0f);      // Kích thước 2x2x2
        DrawCube(0.0f, 0.0f, 0.0f);
    glPopMatrix();
}

```

- Hàm **DrawTorso** để vẽ thân (hình hộp chữ nhật) tại vị trí chỉ định.

```

void DrawTorso(float xPos, float yPos, float zPos)
{
    glPushMatrix();
        glColor3f(0.0f, 0.0f, 1.0f);      // Tô màu xanh
        glTranslatef(xPos, yPos, zPos);
        glScalef(3.0f, 5.0f, 2.0f);      // Kích thước 3x5x2
        DrawCube(0.0f, 0.0f, 0.0f);
    glPopMatrix();
}

```

- Hàm **DrawLeg** để vẽ chân (hình hộp chữ nhật) tại vị trí chỉ định.

```

void DrawLeg(float xPos, float yPos, float zPos)
{
    glPushMatrix();
        glColor3f(1.0f, 1.0f, 0.0f);      // Tô màu vàng
        glTranslatef(xPos, yPos, zPos);
        glScalef(1.0f, 5.0f, 1.0f);      // Kích thước 1x5x1
        DrawCube(0.0f, 0.0f, 0.0f);
    glPopMatrix();
}

```

- Hàm **DrawRobot** để vẽ toàn bộ robot tại vị trí chỉ định.

```

void DrawRobot(float xPos, float yPos, float zPos)
{
    glPushMatrix();
        glTranslatef(xPos, yPos, zPos); // Tọa độ vẽ robot
        // Vẽ phần đầu và phần thân
        DrawHead(1.0f, 2.0f, 0.0f);
        DrawTorso(1.5f, 0.0f, 0.0f);
        /* Di chuyển cánh tay trái và xoay cánh tay để tạo hiệu ứng
        “đang bước đi”.*/
        glPushMatrix();
            glTranslatef(0.0f, -0.5f, 0.0f);
            glRotatef(armAngles[LEFT], 1.0f, 0.0f, 0.0f);
            DrawArm(2.5f, 0.0f, -0.5f);
        glPopMatrix();
        /* Di chuyển cánh tay phải và xoay cánh tay để tạo hiệu ứng
        “đang bước đi”.*/
        glPushMatrix();
            glTranslatef(0.0f, -0.5f, 0.0f);
            glRotatef(armAngles[RIGHT], 1.0f, 0.0f, 0.0f);
            DrawArm(-1.5f, 0.0f, -0.5f);
        glPopMatrix();
        /* Di chuyển chân trái và xoay chân để tạo hiệu ứng “đang bước
        đi”.*/
        glPushMatrix();
            glTranslatef(0.0f, -0.5f, 0.0f);

```

```

        glRotatef(legAngles[LEFT], 1.0f, 0.0f, 0.0f);
        DrawLeg(-0.5f, -5.0f, -0.5f);
    glPopMatrix();
    /* Di chuyển chân phải và xoay chân để tạo hiệu ứng “đang bước
     * đi”.*/
    glPushMatrix();
    glTranslatef(0.0f, -0.5f, 0.0f);
    glRotatef(legAngles[RIGHT], 1.0f, 0.0f, 0.0f);
    DrawLeg(1.5f, -5.0f, -0.5f);
    glPopMatrix();
    glPopMatrix();
}

```

- Hàm **Prepare** để tính toán các góc xoay của tay và chân.

```

void Prepare()
{
    /* Nếu tay/chân đang di chuyển về phía trước thì tăng góc xoay,
     ngược lại (di chuyển về phía sau) thì giảm góc xoay. */
    for (char side = 0; side < 2; side++) {
        // Góc xoay cho tay
        if (armStates[side] == FORWARD_STATE)
            armAngles[side] += 0.1f;
        else
            armAngles[side] -= 0.1f;

        // Thay đổi trạng thái nếu góc xoay vượt quá giá trị cho phép
        if (armAngles[side] >= 15.0f)
            armStates[side] = BACKWARD_STATE;
        else if (armAngles[side] <= -15.0f)
            armStates[side] = FORWARD_STATE;

        // Góc xoay cho chân
        if (legStates[side] == FORWARD_STATE)
            legAngles[side] += 0.1f;
        else
            legAngles[side] -= 0.1f;

        // Thay đổi trạng thái nếu góc xoay vượt quá giá trị cho phép
        if (legAngles[side] >= 15.0f)
            legStates[side] = BACKWARD_STATE;
        else if (legAngles[side] <= -15.0f)
            legStates[side] = FORWARD_STATE;
    }
}

```

- Hàm **Display** để dựng hình.

```

void Display()
{
    glEnable(GL_DEPTH_TEST);
    glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    angle = angle + 0.05f; // Tăng góc xoay của robot
    if (angle >= 360.0f) // Nếu xoay đủ vòng, reset góc xoay
        angle = 0.0f;

    glPushMatrix();
        // Xoay robot quanh trục y
        glRotatef(angle, 0.0f, 1.0f, 0.0f);
        Prepare();
        DrawRobot(0.0f, 0.0f, 0.0f); // Vẽ robot
    glPopMatrix();

    glFlush();
    glutSwapBuffers();
}

```

- Hàm **Init** để thiết lập ánh sáng môi trường và khởi tạo giá trị mặc định cho robot.

```

void Init()
{
    glEnable(GL_DEPTH_TEST);
    glShadeModel(GL_SMOOTH);

    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);

    // Màu ánh sáng ambient
    GLfloat ambientLight[] = { 0.2f, 0.2f, 0.2f, 1.0f };
    // Màu ánh sáng diffuse
    GLfloat diffuseLight[] = { 1.0f, 1.0f, 1.0f, 1.0f };
    // Vị trí nguồn sáng
    GLfloat lightPos[] = { 25.0f, 25.0f, 25.0f, 0.0 };
    // Hướng chiếu sáng
    GLfloat spotDir[] = { 0.0, 0.0, 0.0, 0.0 };

    glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambientLight);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuseLight);
    glLightfv(GL_LIGHT0, GL_POSITION, lightPos);
    glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, spotDir);

    glEnable(GL_COLOR_MATERIAL);
    glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);
}

```

```

glClearColor(0.0, 0.0, 0.0, 0.0);

// Gán giá trị mặc định ban đầu cho robot
angle = 0.0f;
armAngles[LEFT] = 0.0;
armAngles[RIGHT] = 0.0;
legAngles[LEFT] = 0.0;
legAngles[RIGHT] = 0.0;
armStates[LEFT] = FORWARD_STATE;
armStates[RIGHT] = BACKWARD_STATE;
legStates[LEFT] = FORWARD_STATE;
legStates[RIGHT] = BACKWARD_STATE;
}

```

- Hàm **Reshape** để thiết lập khung nhìn, phép chiếu, và camera.

```

void Reshape(int Width, int Height)
{
    glViewport (0, 0, (GLsizei) Width, (GLsizei) Height);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0, (GLfloat) Width / (GLfloat) Height, 1.0,
200.0);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(5.0, 5.0, 50.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
}

```

- Hàm **Idle** để yêu cầu vẽ lại màn hình liên tục.

```

void Idle()
{
    glutPostRedisplay();
}

```

- Hàm **Main**

```

int _tmain(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowPosition(80, 80);
    glutInitWindowSize(800, 600);
    glutCreateWindow("Robot_01");
    glutDisplayFunc(Display);
    glutReshapeFunc(Reshape);
    glutIdleFunc(Idle);
}

```

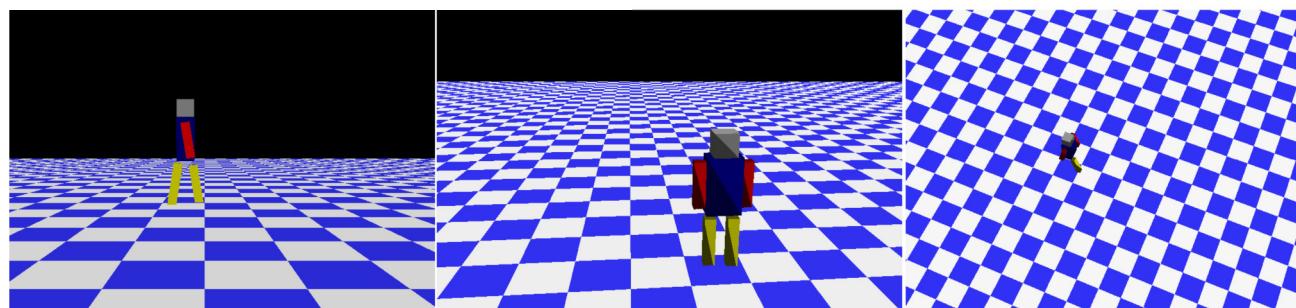
```

    Init();
    glutMainLoop();
    return 0;
}

```

2. Bài TH 02:

- *Yêu cầu:* (xem hình bên dưới).
 - o Vẽ robot di chuyển trên mặt phẳng Oxz (lưới các ô vuông).
 - o Sử dụng các phím \leftarrow , \uparrow , \rightarrow , \downarrow để điều khiển hướng di chuyển của robot.
 - o Sử dụng các phím ‘a’, ‘s’, ‘d’, ‘w’ để thay đổi vị trí camera.



- *Hướng dẫn thực hiện:*
 - o Khai báo thêm các biến toàn cục:

```

// Các hướng di chuyển có thể
const char MOVE_LEFT = 0;
const char MOVE_RIGHT = 1;
const char MOVE_UP = 3;
const char MOVE_DOWN = 4;

// Góc xoay robot tương ứng khi sử dụng các phím điều khiển
const int TURN_LEFT = -90;
const int TURN_RIGHT = 90;
const int TURN_UP = 180;
const int TURN_DOWN = 0;

const int GROUND_SIZE = 200; // Kích thước của mặt sàn
const float FLOOR_HEIGHT = -2; // Độ cao của mặt sàn
const int CHECK_SIZE = 5; // Kích thước của mỗi ô vuông

int moveDirection; // Hướng di chuyển hiện tại
float moveX; // Vị trí di chuyển đến theo trục x
float moveZ; // Vị trí di chuyển đến theo trục z

```

```

float theta;           // Góc xoay của camera (tính tọa độ x và z)
float y;               // Tọa độ y của camera
float dTheta;          // Mức tăng/giảm theta khi điều khiển
float dy;              // Mức tăng/giảm y khi điều khiển

```

- Khai báo thêm các hàm:

```

void DrawGround();
void SettingCamera(float theta, float y);
void Keyboard(unsigned char key, int, int);
void Special(int key, int, int);

```

- Hàm **drawGround** để vẽ mặt sàn.

```

void DrawGround()
{
    int x, z;
    int counter = 0;

    for (x = -GROUND_SIZE; x < GROUND_SIZE; x += CHECK_SIZE)
    {
        for (z = -GROUND_SIZE; z < GROUND_SIZE; z += CHECK_SIZE)
        {
            if (counter % 2 == 0)
                glColor3f(0.2, 0.2, 1.0); // Tô màu xanh
            else
                glColor3f(1.0, 1.0, 1.0); // Tô màu trắng

            glBegin(GL_QUADS);
                glNormal3d(0, 1, 0);
                glVertex3f(x, FLOOR_HEIGHT, z);
                glVertex3f(x, FLOOR_HEIGHT, z + CHECK_SIZE);
                glVertex3f(x + CHECK_SIZE, FLOOR_HEIGHT, z +
                           CHECK_SIZE);
                glVertex3f(x + CHECK_SIZE, FLOOR_HEIGHT, z);
            glEnd();
            counter++;
        }
        counter++;
    }
}

```

- Hàm **SettingCamera** để thiết lập camera.

```

void SettingCamera(float theta, float y)
{
    gluLookAt(50 * sin(theta), y, 50 * cos(theta),

```

```

    0.0, 0.0, 0.0,
    0.0, 1.0, 0.0);
}

```

- Hàm **Keyboard** xử lý sự kiện từ bàn phím (thay đổi vị trí camera).

```

void Keyboard(unsigned char key, int, int)
{
    switch(key)
    {
        case 'a': theta -= dTheta; break;
        case 'd': theta += dTheta; break;
        case 'w': y += dy; break;
        case 's': if (y > dy) y -= dy; break;
    }
    glutPostRedisplay();
}

```

- Hàm **Special** xử lý các phím ←, ↑, →, ↓

```

void Special(int key, int, int)
{
    switch (key)
    {
        case GLUT_KEY_LEFT:
            moveDirection = MOVE_LEFT;
            angle = TURN_LEFT;
            break;
        case GLUT_KEY_RIGHT:
            moveDirection = MOVE_RIGHT;
            angle = TURN_RIGHT;
            break;
        case GLUT_KEY_UP:
            moveDirection = MOVE_UP;
            angle = TURN_UP;
            break;
        case GLUT_KEY_DOWN:
            moveDirection = MOVE_DOWN;
            angle = TURN_DOWN;
            break;
    }
    glutPostRedisplay();
}

```

- Bổ sung thêm cho **Display** để thay đổi camera view và di chuyển robot

```

void Display()
{
    glEnable(GL_DEPTH_TEST);
}

```

```

glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glLoadIdentity(); // reset modelview matrix
SettingCamera(theta, y);

glPushMatrix();
    glTranslatef(moveX, 8.0f, moveZ);
    glRotatef(angle, 0.0f, 1.0f, 0.0f);
    Prepare();
    DrawRobot(0.0f, 0.0f, 0.0f);
glPopMatrix();

glPushMatrix();
    DrawGround();
glPopMatrix();

glFlush();
glutSwapBuffers();
}

```

- o Bổ sung thêm cho hàm **Prepare** để tính toán vị trí di chuyển của robot.

```

void Prepare()
{
    /* Nếu tay/chân đang di chuyển về phía trước thì tăng góc xoay,
    ngược lại (di chuyển về phía sau) thì giảm góc xoay. */
    for (char side = 0; side < 2; side++) {
        // Góc xoay cho tay
        if (armStates[side] == FORWARD_STATE)
            armAngles[side] += 0.1f;
        else
            armAngles[side] -= 0.1f;

        // Thay đổi trạng thái nếu góc xoay vượt quá giá trị cho phép
        if (armAngles[side] >= 15.0f)
            armStates[side] = BACKWARD_STATE;
        else if (armAngles[side] <= -15.0f)
            armStates[side] = FORWARD_STATE;

        // Góc xoay cho chân
        if (legStates[side] == FORWARD_STATE)
            legAngles[side] += 0.1f;
        else
            legAngles[side] -= 0.1f;

        // Thay đổi trạng thái nếu góc xoay vượt quá giá trị cho phép
        if (legAngles[side] >= 15.0f)
            legStates[side] = BACKWARD_STATE;
        else if (legAngles[side] <= -15.0f)
            legStates[side] = FORWARD_STATE;
    }
}

```

```

    }

    switch(moveDirection) {
        case MOVE_LEFT:
            moveX = moveX - 0.015f; break;
        case MOVE_RIGHT:
            moveX = moveX + 0.015f; break;
        case MOVE_UP:
            moveZ = moveZ - 0.015f; break;
        case MOVE_DOWN:
            moveZ = moveZ + 0.015f; break;
    }

}

```

- Bổ sung thêm cho hàm **Init** để thiết lập khởi tạo giá trị mặc định cho robot và camera.

```

void Init()
{
    glEnable(GL_DEPTH_TEST);
    glShadeModel(GL_SMOOTH);

    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);

    // Màu ánh sáng ambient
    GLfloat ambientLight[] = { 0.2f, 0.2f, 0.2f, 1.0f };
    // Màu ánh sáng diffuse
    GLfloat diffuseLight[] = { 1.0f, 1.0f, 1.0f, 1.0f };
    // Vị trí nguồn sáng
    GLfloat lightPos[] = { 25.0f, 25.0f, 25.0f, 0.0 };
    // Hướng chiếu sáng
    GLfloat spotDir[] = { 0.0, 0.0, 0.0, 0.0 };

    glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambientLight);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuseLight);
    glLightfv(GL_LIGHT0, GL_POSITION, lightPos);
    glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, spotDir);

    glEnable(GL_COLOR_MATERIAL);
    glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);
    glClearColor(0.0, 0.0, 0.0, 0.0);

    // Gán giá trị mặc định ban đầu cho robot
    armAngles[LEFT] = 0.0;
    armAngles[RIGHT] = 0.0;
    legAngles[LEFT] = 0.0;
}

```

```

legAngles[RIGHT] = 0.0;
armStates[LEFT] = FORWARD_STATE;
armStates[RIGHT] = BACKWARD_STATE;
legStates[LEFT] = FORWARD_STATE;
legStates[RIGHT] = BACKWARD_STATE;

moveDirection = MOVE_LEFT;
angle = TURN_LEFT;
moveX = 0.0f;
moveZ = 0.0f;

// Gán giá trị mặc định ban đầu cho camera
theta = 0.0f;
y = 5.0f;
dTheta = 0.04f;
dy = 1.0f;
}

```

- **Hàm Main**

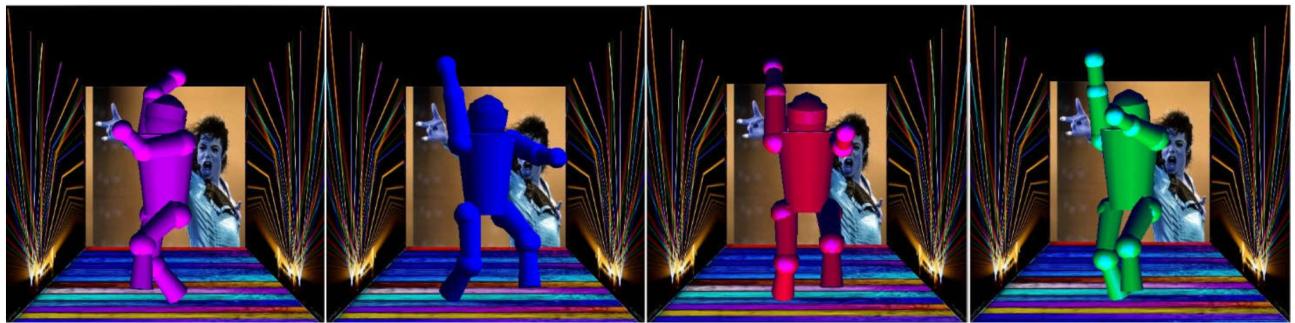
```

int _tmain(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowPosition(80, 80);
    glutInitWindowSize(800, 600);
    glutCreateWindow("Robot_02");
    glutDisplayFunc(Display);
    glutReshapeFunc(Reshape);
    glutIdleFunc(Idle);
    glutSpecialFunc(Special);
    glutKeyboardFunc(Keyboard);
    Init();
    glutMainLoop();
    return 0;
}

```

3. Bài TH 03:

- *Yêu cầu:* (xem hình bên dưới).
 - Vẽ robot nhảy múa biểu diễn ca nhạc.



- *Hướng dẫn thực hiện:*

- Tham khảo source code:
 - https://github.com/chiragragarwal/Dancing_Robot
 - https://github.com/chiragragarwal/Dancing_Robot/blob/master/bot_final.c
- Sinh viên biên dịch và đọc hiểu source code.