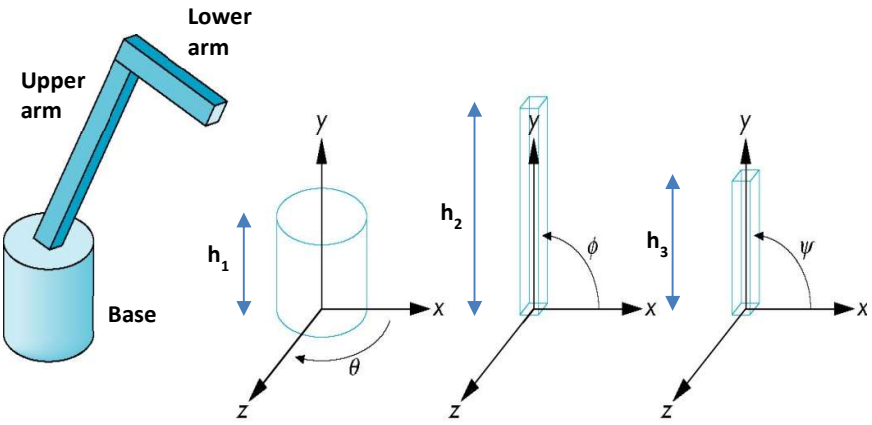
BÀI THỰC HÀNH SỐ 5

Thực hành các phép biến đổi, mô hình phân cấp.

I. Bài toán: Xây dựng điều khiển cánh tay robot bằng mô hình phân cấp.

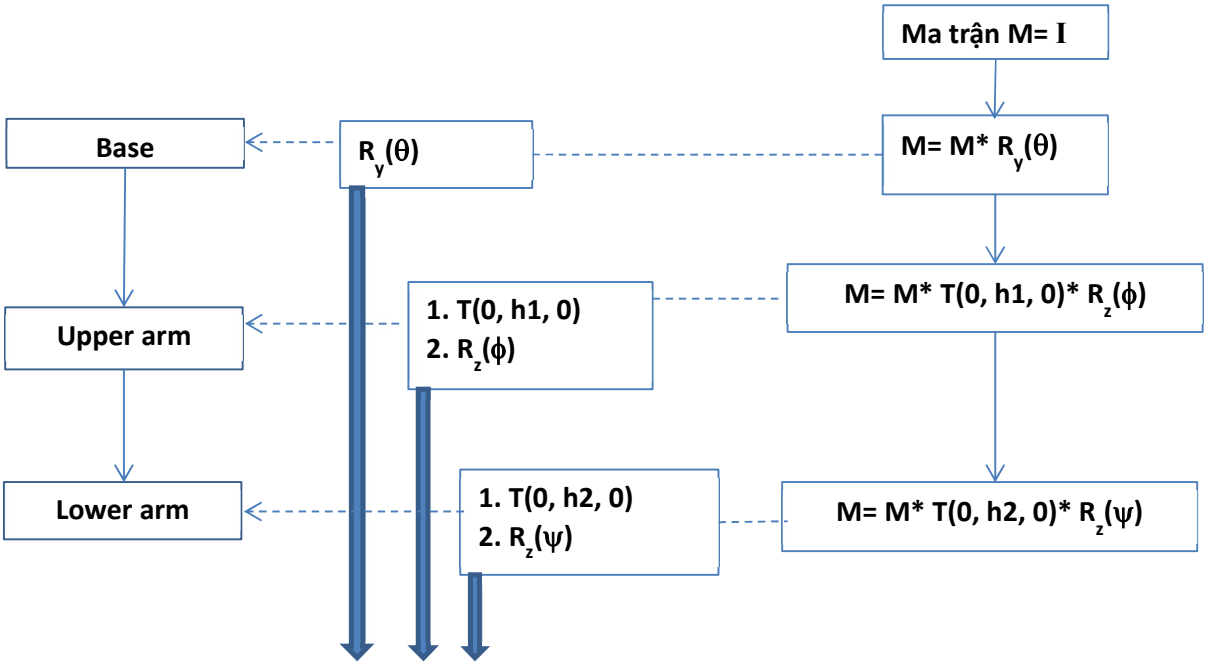


Cánh tay robot

Các bộ phận trong hệ tọa độ địa phương

- Base: Quay quanh trục thẳng đứng của nó góc quay θ
- Upper arm: Quay trong mặt phẳng xy của nó bởi góc quay ϕ
- Lower arm: Quay trong mặt phẳng xy của nó bởi góc quay ψ

Mô hình phân cấp và các ma trận biến đổi theo mô hình phân cấp cho mô hình cánh tay robot trên:



II. Hướng dẫn cài đặt không sử dụng stack

```
……….
GLdouble BASE_HEIGHT = 0.2, BASE_WIDTH = 0.2, UPPER_ARM_HEIGHT = 0.3,
UPPER_ARM_WIDTH = 0.1, LOWER_ARM_HEIGHT = 0.2, LOWER_ARM_WIDTH = 0.05;
mat4 instance;
mat4 model_view;
GLuint model_view_loc;
GLfloat theta[] = { 0, 0, 0 };

void base()
{
	instance = Translate(0.0, 0.5*BASE_HEIGHT, 0.0)
		*Scale(BASE_WIDTH, BASE_HEIGHT, BASE_WIDTH);
	glUniformMatrix4fv(model_view_loc, 1, GL_TRUE, model_view*instance);
	glDrawArrays(GL_TRIANGLES, 0, NumPoints);
}
void upper_arm()
{
	instance = Translate(0.0, 0.5*UPPER_ARM_HEIGHT, 0.0)
		*Scale(UPPER_ARM_WIDTH, UPPER_ARM_HEIGHT, UPPER_ARM_WIDTH);
	glUniformMatrix4fv(model_view_loc, 1, GL_TRUE, model_view*instance);
	glDrawArrays(GL_TRIANGLES, 0, NumPoints);
}
void lower_arm()
{
	instance = Translate(0.0, 0.5*LOWER_ARM_HEIGHT, 0.0)
		*Scale(LOWER_ARM_WIDTH, LOWER_ARM_HEIGHT, LOWER_ARM_WIDTH);
	glUniformMatrix4fv(model_view_loc, 1, GL_TRUE, model_view*instance);
	glDrawArrays(GL_TRIANGLES, 0, NumPoints);
}

……..
void display(void)
{

	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

	model_view = RotateY(theta[0]);
	base();
	model_view = model_view * Translate(0.0, BASE_HEIGHT, 0.0) * RotateZ(theta[1]);
	upper_arm();
	model_view = model_view * Translate(0.0, UPPER_ARM_HEIGHT, 0.0) * RotateZ(theta[2]);
	lower_arm();

	glutSwapBuffers();
}


void keyboard(unsigned char key, int x, int y)
{
	switch (key) {
	case 'b':
		theta[0] += 5;
		if (theta[0] > 360) theta[0] -= 360;
		glutPostRedisplay();
		break;
	case 'B':
		// một số lệnh
		theta[0] -= 5;
		if (theta[0] > 360) theta[0] -= 360;
		glutPostRedisplay();
		break;

	case 'u':
		// một số lệnh
		theta[1] += 5;
```

```
            if (theta[1] > 360) theta[1] -= 360;
            glutPostRedisplay();
            break;
        case 'U':
            // một số lệnh
            theta[1] -= 5;
            if (theta[1] > 360) theta[1] -= 360;
            glutPostRedisplay();
            break;

        case 'l':
            // một số lệnh
            theta[2] += 5;
            if (theta[2] > 360) theta[2] -= 360;
            glutPostRedisplay();
            break;
        case 'L':
            // một số lệnh
            theta[2] -= 5;
            if (theta[2] > 360) theta[2] = 0;
            glutPostRedisplay();
            break;

            break;

        default:
            break;
        }
}


int main( int argc, char **argv )
{
        // main function: program starts here

    glutInit( &argc, argv );
    glutInitDisplayMode( GLUT_DOUBLE|GLUT_RGBA);
    glutInitWindowSize( 640, 640 );
    glutInitWindowPosition(100,150);
    glutCreateWindow( "Drawing a Robot Arm without stack" );

    glewInit();

    generateGeometry( );
    initGPUBuffers( );
    shaderSetup( );

    glutDisplayFunc( display );
    glutKeyboardFunc( keyboard );

    glutMainLoop();
    return 0;
}
```

III. Hướng dẫn cài đặt sử dụng stack

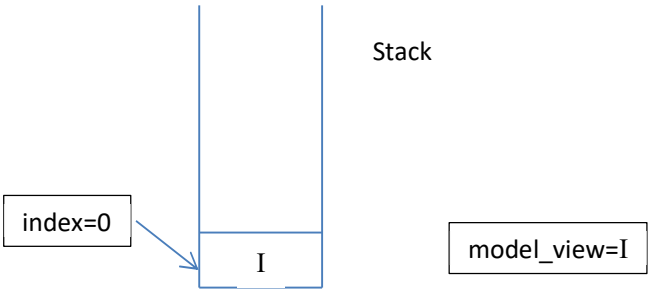
1. Tạo stack để lưu trữ các ma trận modeview

```
class matrix_stack      /*Khai báo lớp stack ma trận*/
{
public:
        static const int MAX = 50;    /*Số phần tử tối đa của stack*/
        matrix_stack() { index = 0; } /*Constructor: Khởi gán chỉ số phần tử đầu tiên là 0*/
        void push(const mat4& matrix);
        mat4 pop();
private:
```

```
        mat4 matrices[MAX];  /*Lưu các ma trận của stack*/
        int index; /*Chỉ số truy cập phần tử đầu tiên*/
};
void matrix_stack::push(const mat4& matrix)
{
        matrices[index] = matrix;
        index++;
}
mat4 matrix_stack::pop()
{
        index--;
        return matrices[index];
}
```

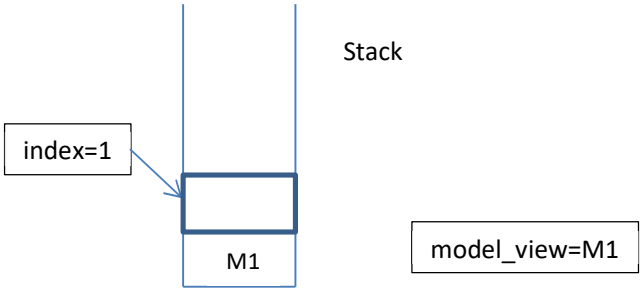
2. Sử dụng stack

```
mat4 model_view;
matrix_stack mvStack;
```




```
model_view=Translate(2, 0, 0); //=M1

mvStack.push(model_view);
```

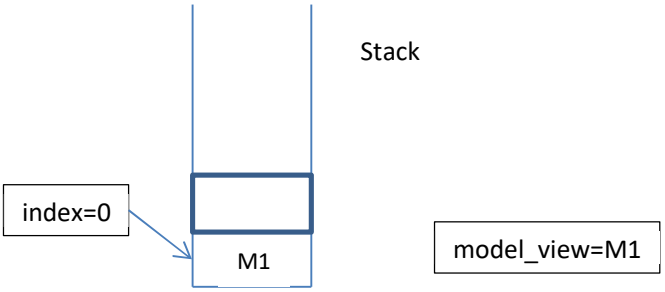



```
model_view *=RotateX(30); //=M2

//ve hình2

Model_view=mvStack.pop(); //M1

//ve hình 3
```



3. Một số cài đặt:

```
.........

GLdouble BASE_HEIGHT = 0.2, BASE_WIDTH=0.2, UPPER_ARM_HEIGHT = 0.3,
             UPPER_ARM_WIDTH=0.1, LOWER_ARM_HEIGHT = 0.2, LOWER_ARM_WIDTH=0.05;
/*mat4 instance;*/
mat4 model_view;
matrix_stack mvStack;
GLuint model_view_loc;
GLfloat theta[] = { 0, 0, 0 };

void base()
{
```

```
	/*instance = Translate(0.0, 0.5*BASE_HEIGHT, 0.0)*Scale(BASE_WIDTH, BASE_HEIGHT,
BASE_WIDTH);*/
	mvStack.push(model_view);
	model_view *= Translate(0.0, 0.5*BASE_HEIGHT, 0.0)*Scale(BASE_WIDTH, BASE_HEIGHT,
BASE_WIDTH);
	glUniformMatrix4fv(model_view_loc, 1, GL_TRUE, model_view);
	glDrawArrays(GL_TRIANGLES, 0, NumPoints);
	model_view = mvStack.pop();
}
void upper_arm()
{
	/*instance = Translate(0.0, 0.5*UPPER_ARM_HEIGHT, 0.0)
		*Scale(UPPER_ARM_WIDTH, UPPER_ARM_HEIGHT, UPPER_ARM_WIDTH);*/

	mvStack.push(model_view);
	model_view *= Translate(0.0, 0.5*UPPER_ARM_HEIGHT, 0.0)
			*Scale(UPPER_ARM_WIDTH, UPPER_ARM_HEIGHT, UPPER_ARM_WIDTH);
	glUniformMatrix4fv(model_view_loc, 1, GL_TRUE, model_view);
	glDrawArrays(GL_TRIANGLES, 0, NumPoints);
	model_view = mvStack.pop();
}
void lower_arm()
{
	/*instance = Translate(0.0, 0.5*LOWER_ARM_HEIGHT, 0.0)
		*Scale(LOWER_ARM_WIDTH, LOWER_ARM_HEIGHT, LOWER_ARM_WIDTH);*/
	mvStack.push(model_view);
	model_view *= Translate(0.0, 0.5*LOWER_ARM_HEIGHT, 0.0)
				*Scale(LOWER_ARM_WIDTH, LOWER_ARM_HEIGHT,
LOWER_ARM_WIDTH);
	glUniformMatrix4fv(model_view_loc, 1, GL_TRUE, model_view);
	glDrawArrays(GL_TRIANGLES, 0, NumPoints);
	model_view = mvStack.pop();
}

…….

void display( void )
{

    glClear( GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT );

	model_view = RotateY(theta[0]);
	base();
	model_view = model_view*Translate(0.0, BASE_HEIGHT, 0.0)*RotateZ(theta[1]);
	upper_arm();
	model_view = model_view*Translate(0.0, UPPER_ARM_HEIGHT, 0.0)*RotateZ(theta[2]);
	lower_arm();

	glutSwapBuffers();
}


void resharp(GLint w, GLint h)
{
	glViewport(0, 0, w, h);
}
void keyboard(unsigned char key, int x, int y)
{
	switch (key) {
	case 'b':
		theta[0] += 5;
		if (theta[0] > 360) theta[0] -= 360;
		glutPostRedisplay();
		break;
	case 'B':
		theta[0] -= 5;
```

```
                if (theta[0] > 360) theta[0] -= 360;
                glutPostRedisplay();
                break;

        case 'u':
                theta[1] += 5;
                if (theta[1] > 360) theta[1] -= 360;
                glutPostRedisplay();
                break;
        case 'U':
                theta[1] -= 5;
                if (theta[1] > 360) theta[1] -= 360;
                glutPostRedisplay();
                break;

        case 'l':
                theta[2] += 5;
                if (theta[2] > 360) theta[2] -= 360;
                glutPostRedisplay();
                break;
        case 'L':
                theta[2] -= 5;
                if (theta[2] > 360) theta[2] = 0;
                glutPostRedisplay();
                break;

                break;

        default:
                break;
        }
}


int main( int argc, char **argv )
{
        glutInit( &argc, argv );
        glutInitDisplayMode( GLUT_DOUBLE|GLUT_RGBA);
        glutInitWindowSize( 640, 640 );
        glutInitWindowPosition(100,150);
        glutCreateWindow( "Drawing a Robot Arm with stack" );

        glewInit();

        generateGeometry();
        initGPUBuffers();
        shaderSetup( );

        glutDisplayFunc(display);
        glutKeyboardFunc(keyboard);
        glutReshapeFunc(resharp);
        glutMainLoop();
    return 0;
}
```