Computer Graphics
- Interactive Program1

Sung Soo Hwang

Review: Callback functions in GLUT

- Keyboard callbacks
  - void cb_keyboard(unsigned char key, int x, int y)
    - key: ASCII character of the pressed key
    - x, y: mouse position
    - It is registered by glutKeyboardFunc(cb_keyboard).

*ASCII: American Standard Code for Information Interchange

Review: Callback functions in GLUT

- Keyboard callbacks
  - void cb_special(int key, int x, int y)
    - key: non-ASCII of the pressed key
    - x, y: mouse position
    - It is registered by glutSpecialFunc(cb_special).

GLUT_KEY_F1, GLUT_KEY_F2, …, GLUT_KEY_F12	F1 through F12 keys
GLUT_KEY_PAGE_UP, GLUT_KEY_PAGE_DOWN	Page Up and Page Down keys
GLUT_KEY_HOME, GLUT_KEY_END	Home and End keys
GLUT_KEY_LEFT, GLUT_KEY_RIGHT, GLUT_KEY_UP, GLUT_KEY_DOWN	Arrow keys
GLUT_KEY_INSERT	Insert key

Review: Callback functions in GLUT

- How to deal with shift, ctrl, and alt modifiers?

- int glutGetModifiers(void)
  - Returns the state of modifier keys (shift, ctrl, alt) at the time when the input event for a keyboard, special, or mouse callback is generated.
  - The return value is generated from the following constants:

GLUT_ACTIVE_SHIFT	Set if the Shift modifier is active
GLUT_ACTIVE_CTRL	Set if the Ctrl modifier is active
GLUT_ACTIVE_ALT	Set if the Alt modifier is active

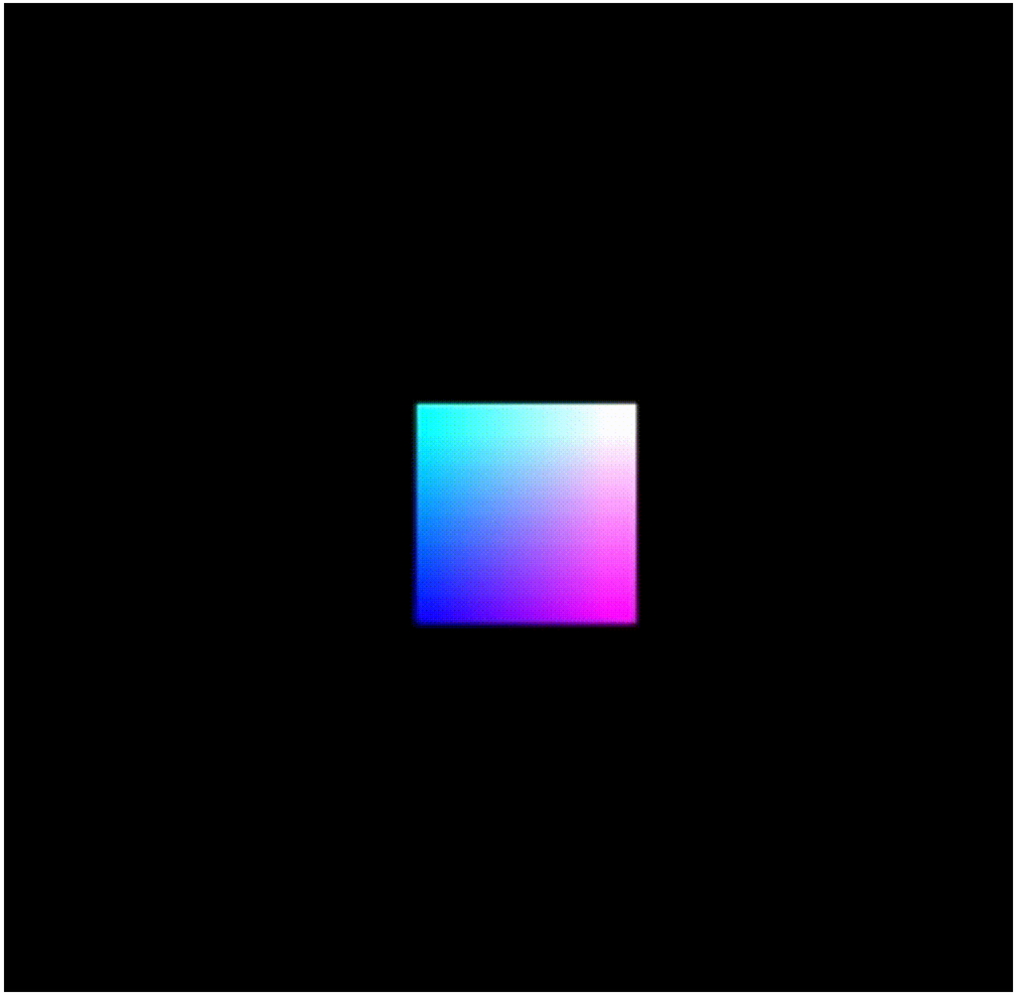
  - Note : there can be multiple active modifiers, which can be checked with the bitwise AND operator (&) as follows:

```
int modifiers = glutGetModifiers();
if (modifiers & GLUT_ACTIVE_CTRL) printf("ctrl pressed₩n");
if (modifiers & GLUT_ACTIVE_ALT) printf("alt pressed₩n");
if (modifiers & GLUT_ACTIVE_SHIFT) printf("shift pressed₩n");
```

Practice

- According to the user's keyboard input, the position and rotation of the primitive are interactively changed as follows:
  - **Right arrow / Left arrow:**
    Add a positive / negative offset to the position of the model along the x-axis of the local coordinate system.
  - **Up arrow / Down arrow:**
    Add a positive / negative offset to the position of the model along the y-axis of the world coordinate system.
  - **F1 key / F2 key:**
    Rotate the model by a positive / negative offset angle around the y-axis of the world coordinate system, i.e., $[0\ 1\ 0]^T$.

Practice



*The provided code is only available for the geometric primitives (cube, sphere, cone, cylinder, torus).

Practice

- According to the user's keyboard input, the position and rotation of the primitive are interactively changed as follows:
  - **Right arrow / Left arrow:**
    Add a positive / negative offset to the position of the model along the x-axis of the local coordinate system.
  - **Up arrow / Down arrow:**
    Add a positive / negative offset to the position of the model along the y-axis of the world coordinate system.
  - **F1 key / F2 key:**
    Rotate the model by a positive / negative offset angle around the y-axis of the world coordinate system, i.e., $[0\ 1\ 0]^T$.

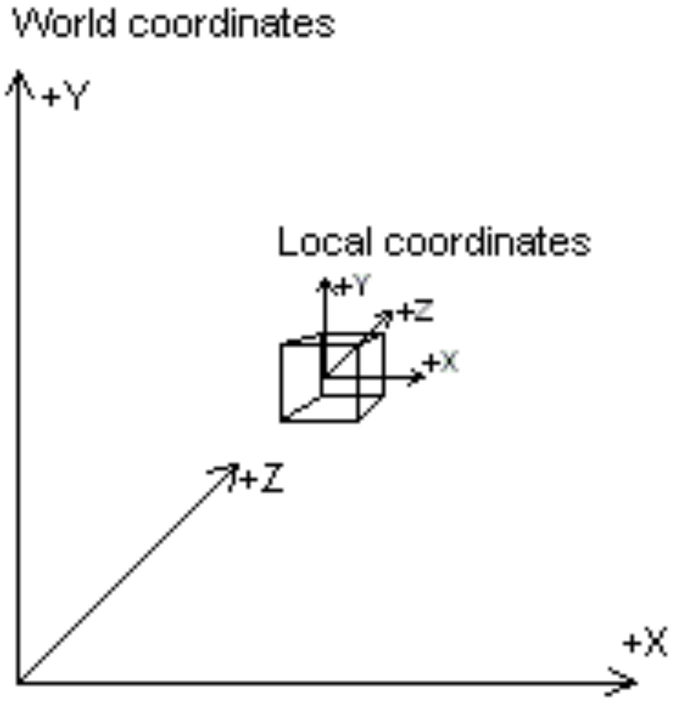
What is the difference between local and world coordinate?

Local vs World

- The coordinate system used for creating a model is named local coordinate system.

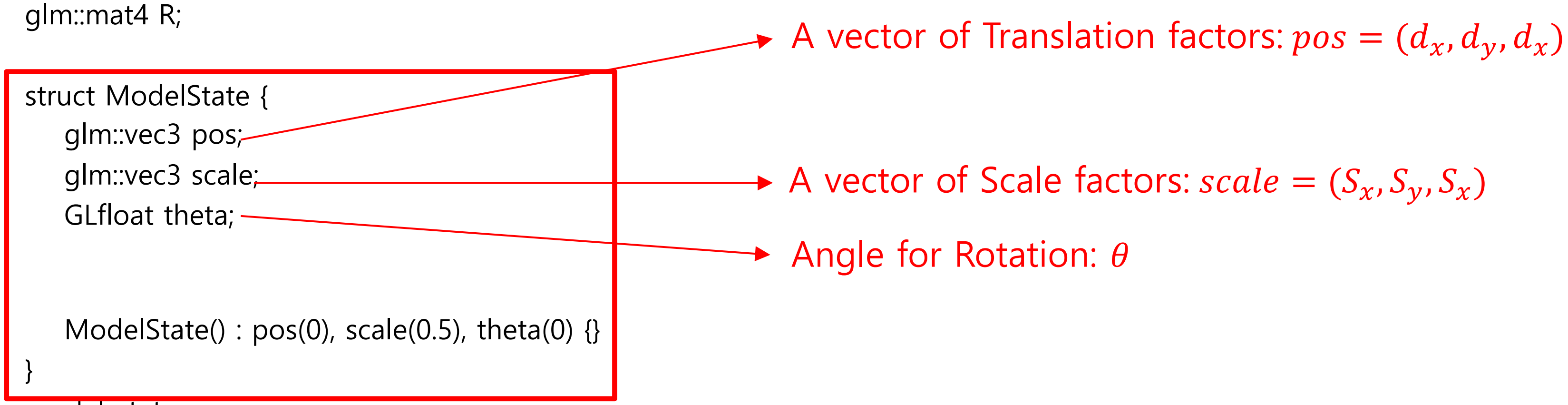
- The single coordinate system that “assembles” all models is named world coordinate system.

Let's practice with OpenGL code!



Practice: Whole code

```
// main.cpp
#include <stdio.h>
#include <GL/glew.h>
#include <GL/glut.h>
#include "LoadShaders.h"
#include <glm/gtc/matrix_transform.hpp>
#include <glm/gtc/type_ptr.hpp>
#include <time.h>
#include "Primi.h"
#include <vector>
#define _USE_MATH_DEFINES
#include <math.h>

GLuint program;
int idx_selected = 0;
int projection_mode = 0;
std::vector<Model*> models;

glm::mat4 R;
```



```
struct ModelState {
    glm::vec3 pos;
    glm::vec3 scale;
    GLfloat theta;


    ModelState() : pos(0), scale(0.5), theta(0) {}
}
model_state;
```

A vector of Translation factors: $pos = (d_x, d_y, d_x)$

A vector of Scale factors: $scale = (S_x, S_y, S_x)$

Angle for Rotation: θ

Practice: Whole code

```
void init()
{
    srand(clock());
    models.push_back(new CubePrimitive(1.0f, 1.0f, 1.0f));
    models.push_back(new SpherePrimitive(0.5f, 15, 15));
    models.push_back(new ConePrimitive(0.5f, 1.0f, 10));
    models.push_back(new CylinderPrimitive(0.5f, 1.0f, 15));
    models.push_back(new TorusPrimitive(0.3f, 0.3f, 30, 10));


    ShaderInfo shaders[] = {
        {GL_VERTEX_SHADER, "shader.vert"},
        {GL_FRAGMENT_SHADER, "shader.frag"},
        {GL_NONE, NULL}
    };

    program = LoadShaders(shaders);
    glUseProgram(program);

    int num_of_models = (int)models.size();
    for (int i = 0; i < num_of_models; ++i){
        models[i] -> init(program);
    }
    glEnable(GL_DEPTH_TEST);
    glDepthFunc(GL_LESS);

    glCullFace(GL_BACK);
    glEnable(GL_CULL_FACE);
}
```

*You should implement the car model for the assignment.

Practice: Whole code

```
void release_models()
{
    int n = (int) models.size();
    for (int i = 0; i < n; ++i){
        if (models[i]){
            delete models[i];
            models[i] = NULL;
        }
    }
    models.clear();
}

void display()
{
    using namespace glm;
    ... code ...

    mat4 Transf(1.0f);


    Transf = translate(Transf, model_state.pos);
    Transf = rotate(Transf, model_state.theta, vec3(0.0f, 1.0f, 0.0f));
    Transf = scale(Transf, model_state.scale);
    glUniformMatrix4fv(1, 1, GL_FALSE, value_ptr(Transf));

    ... code ...
    glFlush();
    glutSwapBuffers();
    //glutPostRedisplay
}
```

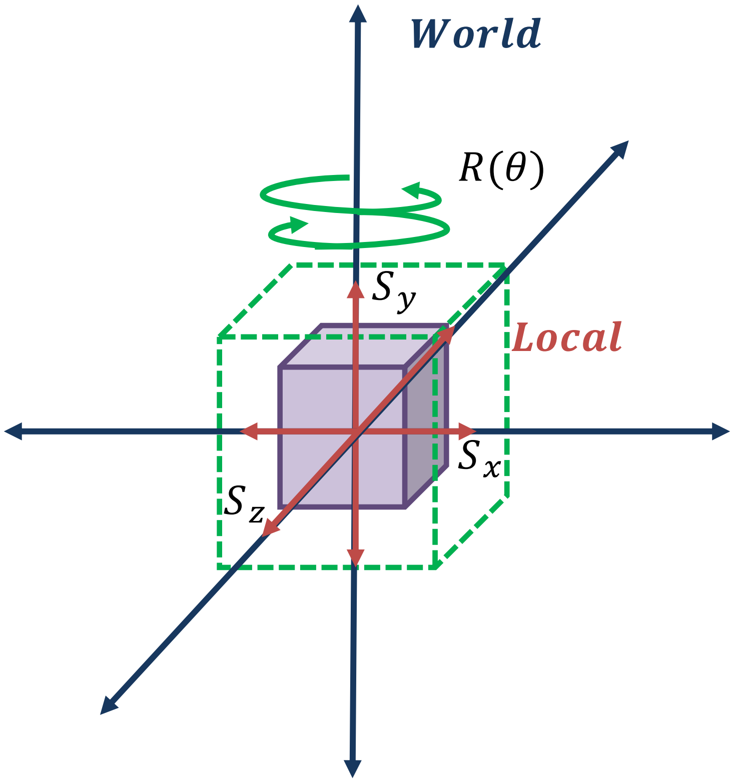
$$Transf = I \cdot T(d_x, d_y, d_z) \cdot R(\theta) \cdot S(S_x, S_y, S_z)$$

Practice: Whole code

```
void release_models()
{
    int n = (int) models.size();
    for (int i = 0; i < n; ++i){
        if (models[i]){
            delete models[i];
            models[i] = NULL;
        }
    }
    models.clear();
}

void display()
{
    using namespace glm;
    ... code ...

    mat4 Transf(1.0f);


    Transf = translate(Transf, model_state.pos);
    Transf = rotate(Transf, model_state.theta, vec3(0.0f, 1.0f, 0.0f));
    Transf = scale(Transf, model_state.scale);
    glUniformMatrix4fv(1, 1, GL_FALSE, value_ptr(Transf));

    ... code ...
    glFlush();
    glutSwapBuffers();
    //glutPostRedisplay
}
```

When Scaling and Rotation, the local and world coordinates coincide, so that the shape of the model and the axis of rotation are not "distorted".

$$Transf = I \cdot T(d_x, d_y, d_z) \cdot R(\theta) \cdot S(S_x, S_y, S_z)$$

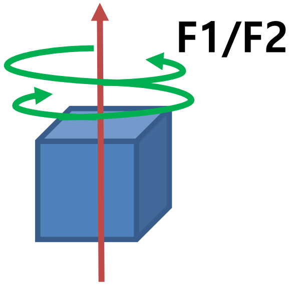


Practice: Whole code

```
void keyboard(unsigned char key, int x, int y)
{
    switch(key)
    {
        case '1':
        case '2':
        case '3':
        case '4':
        case '5':
            idx_selected = key - '1';
            glutPostRedisplay();
            break;
    }
}
```

'1': cube, '2': sphere, '3': cone, '4': cylinder, '5': torus

Let the GLUT redraw the screen by calling display().



```
void keyboardSpecial(int key, int x, int y)
{
    switch (key) {
    case GLUT_KEY_F1:
        model_state.theta -= 0.05f;
        glutPostRedisplay();
        break;
    case GLUT_KEY_F2:
        model_state.theta += 0.05f;
        glutPostRedisplay();
        break;
```

... Next Slide ...

```
void display()
{   ... code ...
    mat4 Transf(1.0f);


    Transf = translate(Transf, model_state.pos);
    Transf = rotate(Transf, model_state.theta, vec3(0.0f, 1.0f, 0.0f));
    Transf = scale(Transf, model_state.scale);
    glUniformMatrix4fv(1, 1, GL_FALSE, value_ptr(Transf));

    ... code ...
}
```

F1/F2:
Rotate the model by -0.05/+0.05 around the y-axis.

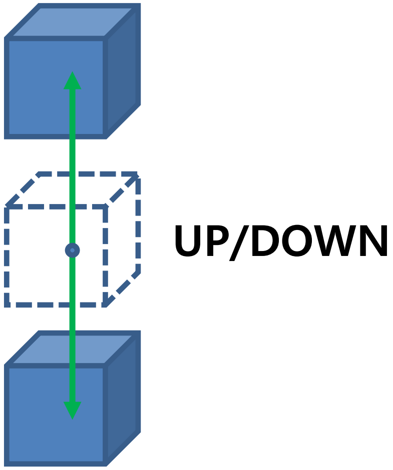
Practice: Whole code

```
case GLUT_KEY_UP:
    model_state.pos[1] += 0.05f;
    glutPostRedisplay();
    break;


case GLUT_KEY_DOWN:
    model_state.pos[1] -= 0.05f;
    glutPostRedisplay();
    break;
```

```
void display()
{   ... code ...
    mat4 Transf(1.0f);


    Transf = translate(Transf, model_state.pos);
    Transf = rotate(Transf, model_state.theta, vec3(0.0f, 1.0f, 0.0f));
    Transf = scale(Transf, model_state.scale);
    glUniformMatrix4fv(1, 1, GL_FALSE, value_ptr(Transf));


    ... code ...
}
```

UP/DOWN:
Add -0.05/+0.05 to the y coordinate of the y-axis of the world coordinate system

```
case GLUT_KEY_RIGHT:
    using namespace glm;
    R = rotate(mat4(1.0f), model_state.theta, vec3(0.0f, 1.0f, 0.0f));
    model_state.pos += 0.05f * vec3(R[0]);
    glutPostRedisplay();
    break;


case GLUT_KEY_LEFT:
    using namespace glm;
    R = rotate(mat4(1.0f), model_state.theta, vec3(0.0f, 1.0f, 0.0f));
    model_state.pos -= 0.05f * vec3(R[0]);
    glutPostRedisplay();
    break;
}
}
```



Practice: Whole code

```
    case GLUT_KEY_UP:
        model_state.pos[1] += 0.05f;
        glutPostRedisplay();
        break;


    case GLUT_KEY_DOWN:
        model_state.pos[1] -= 0.05f;
        glutPostRedisplay();
        break;


    case GLUT_KEY_RIGHT:
        using namespace glm;
        R = rotate(mat4(1.0f), model_state.theta, vec3(0.0f, 1.0f, 0.0f));
        model_state.pos += 0.05f * vec3(R[0]);
        glutPostRedisplay();
        break;


    case GLUT_KEY_LEFT:
        using namespace glm;
        R = rotate(mat4(1.0f), model_state.theta, vec3(0.0f, 1.0f, 0.0f));
        model_state.pos -= 0.05f * vec3(R[0]);
        glutPostRedisplay();
        break;
    }
}
```

Right/Left:
Add -0.05/+0.05 to the position of the model along the x-axis of the local coordinate system.

Practice: Whole code

```
    case GLUT_KEY_UP:
        model_state.pos[1] += 0.05f;
        glutPostRedisplay();
        break;


    case GLUT_KEY_DOWN:
        model_state.pos[1] -= 0.05f;
        glutPostRedisplay();
        break;


    case GLUT_KEY_RIGHT:
        using namespace glm;
        R = rotate(mat4(1.0f), model_state.theta, vec3(0.0f, 1.0f, 0.0f));
        model_state.pos += 0.05f * vec3(R[0]);
        glutPostRedisplay();
        break;


    case GLUT_KEY_LEFT:
        using namespace glm;
        R = rotate(mat4(1.0f), model_state.theta, vec3(0.0f, 1.0f, 0.0f));
        model_state.pos -= 0.05f * vec3(R[0]);
        glutPostRedisplay();
        break;
    }
}
```

Right/Left:
Add -0.05/+0.05 to the position of the model along the x-axis of the local coordinate system.

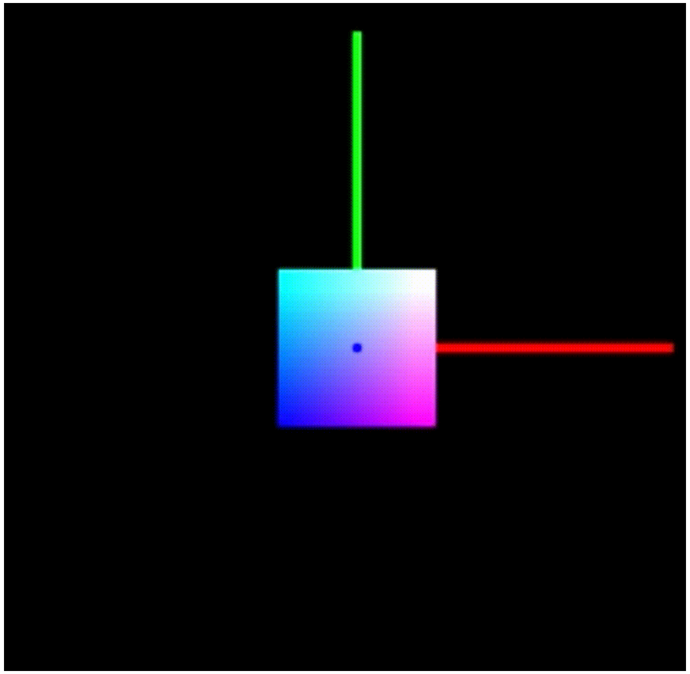
Why do this?

Practice: Whole code

```
    case GLUT_KEY_UP:
        model_state.pos[1] += 0.05f;
        glutPostRedisplay();
        break;


    case GLUT_KEY_DOWN:
        model_state.pos[1] -= 0.05f;
        glutPostRedisplay();
        break;


    case GLUT_KEY_RIGHT:
        using namespace glm;
        R = rotate(mat4(1.0f), model_state.theta, vec3(0.0f, 1.0f, 0.0f));
        model_state.pos += 0.05f * vec3(R[0]);
        glutPostRedisplay();
        break;


    case GLUT_KEY_LEFT:
        using namespace glm;
        R = rotate(mat4(1.0f), model_state.theta, vec3(0.0f, 1.0f, 0.0f));
        model_state.pos -= 0.05f * vec3(R[0]);
        glutPostRedisplay();
        break;
    }
}
```

Right/Left:
Add -0.05/+0.05 to the position of the model along the x-axis of the local coordinate system.

This is because when the model rotates about the y-axis, the x-axis and z-axis of the local coordinates also rotate.

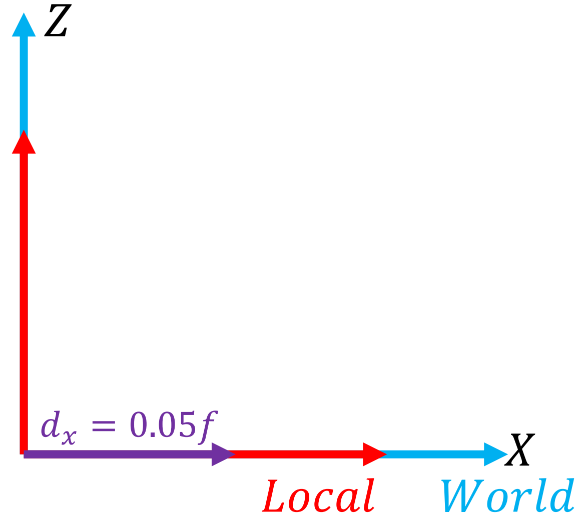


Practice: Whole code

```
    case GLUT_KEY_UP:
        model_state.pos[1] += 0.05f;
        glutPostRedisplay();
        break;


    case GLUT_KEY_DOWN:
        model_state.pos[1] -= 0.05f;
        glutPostRedisplay();
        break;


    case GLUT_KEY_RIGHT:
        using namespace glm;
        R = rotate(mat4(1.0f), model_state.theta, vec3(0.0f, 1.0f, 0.0f));
        model_state.pos += 0.05f * vec3(R[0]);
        glutPostRedisplay();
        break;


    case GLUT_KEY_LEFT:
        using namespace glm;
        R = rotate(mat4(1.0f), model_state.theta, vec3(0.0f, 1.0f, 0.0f));
        model_state.pos -= 0.05f * vec3(R[0]);
        glutPostRedisplay();
        break;
    }
}
```

Right/Left:
Add -0.05/+0.05 to the position of the model along the x-axis of the local coordinate system.

In Top view (**Before Rotation**)

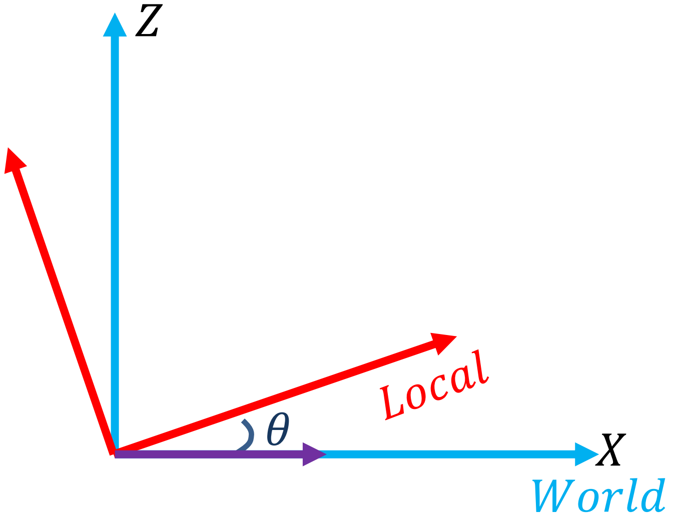


Practice: Whole code

```
    case GLUT_KEY_UP:
        model_state.pos[1] += 0.05f;
        glutPostRedisplay();
        break;


    case GLUT_KEY_DOWN:
        model_state.pos[1] -= 0.05f;
        glutPostRedisplay();
        break;


    case GLUT_KEY_RIGHT:
        using namespace glm;
        R = rotate(mat4(1.0f), model_state.theta, vec3(0.0f, 1.0f, 0.0f));
        model_state.pos += 0.05f * vec3(R[0]);
        glutPostRedisplay();
        break;


    case GLUT_KEY_LEFT:
        using namespace glm;
        R = rotate(mat4(1.0f), model_state.theta, vec3(0.0f, 1.0f, 0.0f));
        model_state.pos -= 0.05f * vec3(R[0]);
        glutPostRedisplay();
        break;
    }
}
```

Right/Left:
Add -0.05/+0.05 to the position of the model along the x-axis of the local coordinate system.

In Top view (**After Rotation**)

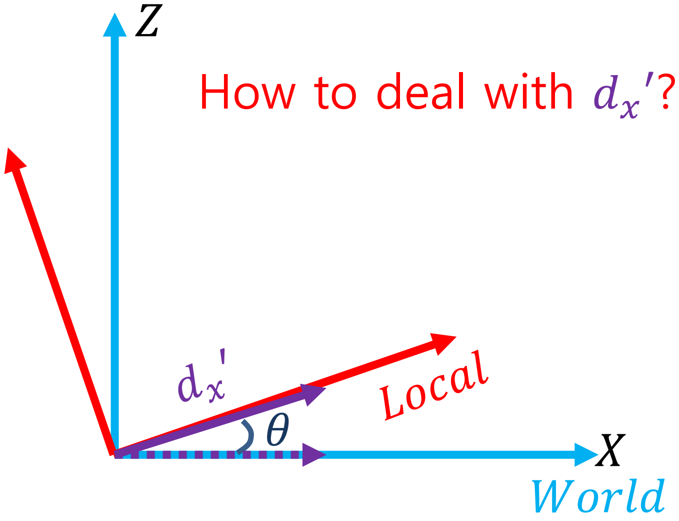


Practice: Whole code

```
    case GLUT_KEY_UP:
        model_state.pos[1] += 0.05f;
        glutPostRedisplay();
        break;


    case GLUT_KEY_DOWN:
        model_state.pos[1] -= 0.05f;
        glutPostRedisplay();
        break;


    case GLUT_KEY_RIGHT:
        using namespace glm;
        R = rotate(mat4(1.0f), model_state.theta, vec3(0.0f, 1.0f, 0.0f));
        model_state.pos += 0.05f * vec3(R[0]);
        glutPostRedisplay();
        break;


    case GLUT_KEY_LEFT:
        using namespace glm;
        R = rotate(mat4(1.0f), model_state.theta, vec3(0.0f, 1.0f, 0.0f));
        model_state.pos -= 0.05f * vec3(R[0]);
        glutPostRedisplay();
        break;
    }
}
```

Right/Left:
Add -0.05/+0.05 to the position of the model along the x-axis of the local coordinate system.

In top view (**After Rotation**)



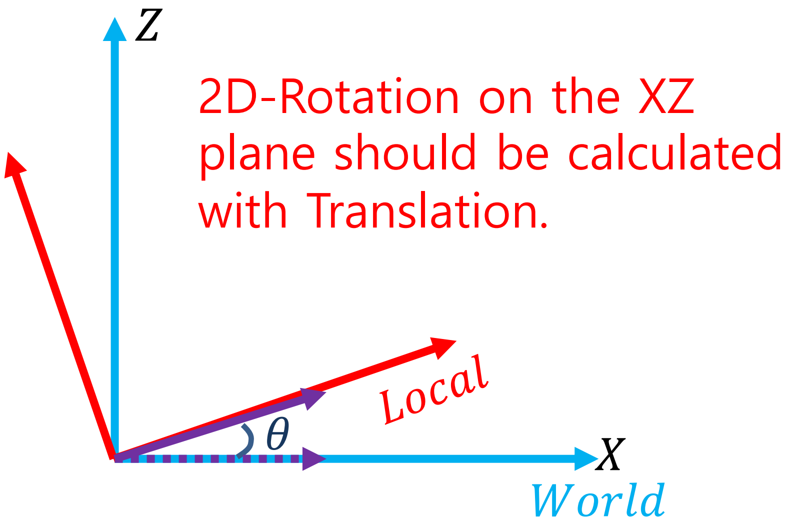
How to deal with d_x'?

Practice: Whole code

```
    case GLUT_KEY_UP:
        model_state.pos[1] += 0.05f;
        glutPostRedisplay();
        break;


    case GLUT_KEY_DOWN:
        model_state.pos[1] -= 0.05f;
        glutPostRedisplay();
        break;


    case GLUT_KEY_RIGHT:
        using namespace glm;
        R = rotate(mat4(1.0f), model_state.theta, vec3(0.0f, 1.0f, 0.0f));
        model_state.pos += 0.05f * vec3(R[0]);
        glutPostRedisplay();
        break;


    case GLUT_KEY_LEFT:
        using namespace glm;
        R = rotate(mat4(1.0f), model_state.theta, vec3(0.0f, 1.0f, 0.0f));
        model_state.pos -= 0.05f * vec3(R[0]);
        glutPostRedisplay();
        break;
    }
}
```

Right/Left:
Add -0.05/+0.05 to the position of the model along the x-axis of the local coordinate system.

In Top view (**After Rotation**)



2D-Rotation on the XZ plane should be calculated with Translation.

Practice: Whole code

```
    case GLUT_KEY_UP:
        model_state.pos[1] += 0.05f;
        glutPostRedisplay();
        break;


    case GLUT_KEY_DOWN:
        model_state.pos[1] -= 0.05f;
        glutPostRedisplay();
        break;


    case GLUT_KEY_RIGHT:
        using namespace glm;
        R = rotate(mat4(1.0f), model_state.theta, vec3(0.0f, 1.0f, 0.0f));
        model_state.pos += 0.05f * vec3(R[0]);
        glutPostRedisplay();
        break;


    case GLUT_KEY_LEFT:
        using namespace glm;
        R = rotate(mat4(1.0f), model_state.theta, vec3(0.0f, 1.0f, 0.0f));
        model_state.pos -= 0.05f * vec3(R[0]);
        glutPostRedisplay();
        break;
    }
}
```

Right/Left:
Add -0.05/+0.05 to the position of the model along the x-axis of the local coordinate system.

$$R_y(\theta) = \begin{vmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Practice: Whole code

```
    case GLUT_KEY_UP:
        model_state.pos[1] += 0.05f;
        glutPostRedisplay();
        break;


    case GLUT_KEY_DOWN:
        model_state.pos[1] -= 0.05f;
        glutPostRedisplay();
        break;


    case GLUT_KEY_RIGHT:
        using namespace glm;
        R = rotate(mat4(1.0f), model_state.theta, vec3(0.0f, 1.0f, 0.0f));
        model_state.pos += 0.05f * vec3(R[0]);
        glutPostRedisplay();
        break;


    case GLUT_KEY_LEFT:
        using namespace glm;
        R = rotate(mat4(1.0f), model_state.theta, vec3(0.0f, 1.0f, 0.0f));
        model_state.pos -= 0.05f * vec3(R[0]);
        glutPostRedisplay();
        break;
    }
}
```

Right/Left:
Add -0.05/+0.05 to the position of the model along the x-axis of the local coordinate system.

$$R_y(\theta) = \begin{vmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

$$vec3(\cos\theta, 0, -\sin\theta)$$

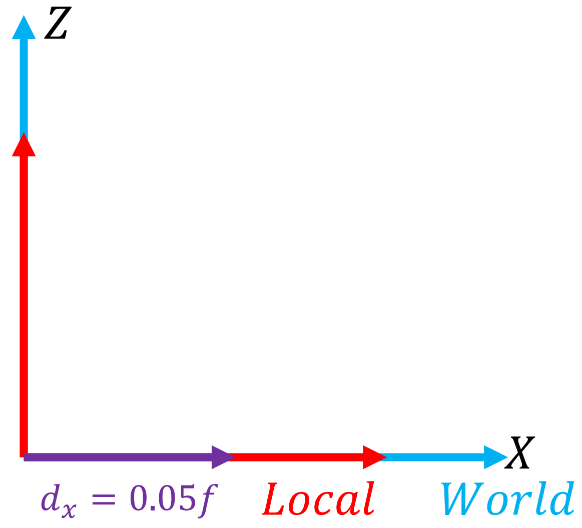
Extract the 2D Rotation component. (for d_x)

Practice: Whole code

```
    case GLUT_KEY_UP:
        model_state.pos[1] += 0.05f;
        glutPostRedisplay();
        break;


    case GLUT_KEY_DOWN:
        model_state.pos[1] -= 0.05f;
        glutPostRedisplay();
        break;


    case GLUT_KEY_RIGHT:
        using namespace glm;
        R = rotate(mat4(1.0f), model_state.theta, vec3(0.0f, 1.0f, 0.0f));
        model_state.pos += 0.05f * vec3(R[0]);
        glutPostRedisplay();
        break;


    case GLUT_KEY_LEFT:
        using namespace glm;
        R = rotate(mat4(1.0f), model_state.theta, vec3(0.0f, 1.0f, 0.0f));
        model_state.pos -= 0.05f * vec3(R[0]);
        glutPostRedisplay();
        break;
    }
}
```

Right/Left:
Add -0.05/+0.05 to the position of the model along the x-axis of the local coordinate system.

In Top view (**Before Rotation**)

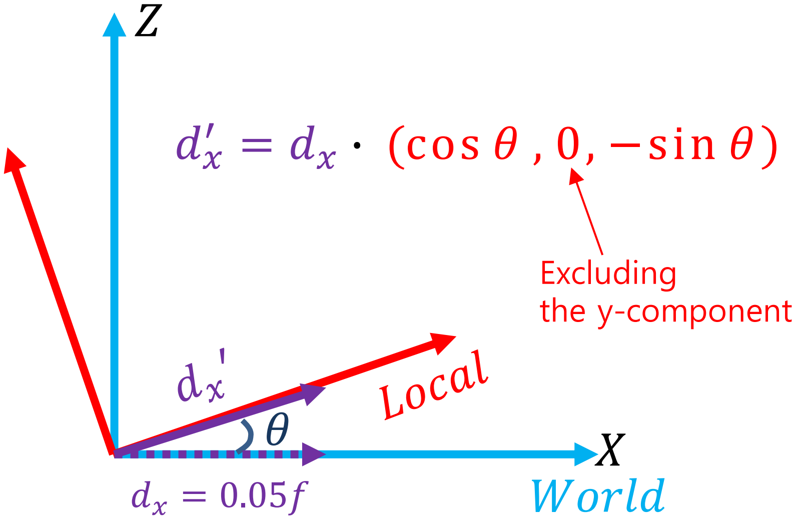


Practice: Whole code

```
    case GLUT_KEY_UP:
        model_state.pos[1] += 0.05f;
        glutPostRedisplay();
        break;


    case GLUT_KEY_DOWN:
        model_state.pos[1] -= 0.05f;
        glutPostRedisplay();
        break;


    case GLUT_KEY_RIGHT:
        using namespace glm;
        R = rotate(mat4(1.0f), model_state.theta, vec3(0.0f, 1.0f, 0.0f));
        model_state.pos += 0.05f * vec3(R[0]);
        glutPostRedisplay();
        break;


    case GLUT_KEY_LEFT:
        using namespace glm;
        R = rotate(mat4(1.0f), model_state.theta, vec3(0.0f, 1.0f, 0.0f));
        model_state.pos -= 0.05f * vec3(R[0]);
        glutPostRedisplay();
        break;
    }
}
```

Right/Left:
Add -0.05/+0.05 to the position of the model along the x-axis of the local coordinate system.

In Top view (**After Rotation**)



Z
$d'_x = d_x \cdot (\cos\theta, 0, -\sin\theta)$
Excluding the y-component
d_x'
θ
$Local$
X
$d_x = 0.05f$
$World$

Practice: Whole code

```
void main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGBA|GLUT_DOUBLE);
    glutInitWindowSize(512, 512);
    glutCreateWindow("Drawing Primitives");
    GLenum err = glewInit();
    if (err != GLEW_OK){
        fprintf(stderr, "Error: %s₩n", glewGetErrorString(err));
        exit(EXIT_FAILURE);
    }
    init();
    glutDisplayFunc(display);
    glutKeyboardFunc(keyboard);
    glutSpecialFunc(keyboardSpecial);
    glutMainLoop();
    release_models();
}
```