AS01-Transformation

105072123 黃海茵

```
// [TODO] given a translation vector then output a Matrix4 (Translation Matrix)
Matrix4 translate(Vector3 vec)
{
    Matrix4 mat;

    mat = Matrix4(
        1, 0, 0, vec.x,
        0, 1, 0, vec.y,
        0, 0, 1, vec.z,
        0, 0, 0, 1
    );

    return mat;
}

// [TODO] given a scaling vector then output a Matrix4 (Scaling Matrix)
Matrix4 scaling(Vector3 vec)
{
    Matrix4 mat;

    mat = Matrix4(
        vec.x, 0, 0, 0,
        0, vec.y, 0, 0,
        0, 0, vec.z, 0,
        0, 0, 0, 1
    );

    return mat;
}
```

```
// [TODO] given a float value then ouput a rotation matrix alone axis-X (rotate alone axis-X)
Matrix4 rotateX(GLfloat val)
{
    Matrix4 mat;

    mat = Matrix4(
        1, 0, 0, 0,
        0, cos(val), -sin(val), 0,
        0, sin(val), cos(val), 0,
        0, 0, 0, 1
    );

    return mat;
}

// [TODO] given a float value then ouput a rotation matrix alone axis-Y (rotate alone axis-Y)
Matrix4 rotateY(GLfloat val)
{
    Matrix4 mat;

    mat = Matrix4(
        cos(val), 0, sin(val), 0,
        0, 1, 0, 0,
        -sin(val), 0, cos(val), 0,
        0, 0, 0, 1
    );

    return mat;
}

// [TODO] given a float value then ouput a rotation matrix alone axis-Z (rotate alone axis-Z)
Matrix4 rotateZ(GLfloat val)
{
    Matrix4 mat;

    mat = Matrix4(
        cos(val), -sin(val), 0, 0,
        sin(val), cos(val), 0, 0,
        0, 0, 1, 0,
        0, 0, 0, 1
    );

    return mat;
}
```

```
// [TODO] compute viewing matrix accroding to the setting of main_camera
void setViewingMatrix()
{

    Matrix4 T = Matrix4(
        1, 0, 0, -main_camera.position.x,
        0, 1, 0, -main_camera.position.y,
        0, 0, 1, -main_camera.position.z,
        0, 0, 0, 1
    );

    Vector3 Rz = (main_camera.position - main_camera.center).normalize();
    Vector3 Rx = -Rz.cross(main_camera.up_vector).normalize();
    Vector3 Ry = Rz.cross(Rx);

    Matrix4 R = Matrix4 (
        Rx.x, Rx.y, Rx.z, 0,
        Ry.x, Ry.y, Ry.z, 0,
        Rz.x, Rz.y, Rz.z, 0,
        0, 0, 0, 1
    );

    view_matrix = R * T;
}

// [TODO] compute orthogonal projection matrix
void setOrthogonal()
{
    cur_proj_mode = Orthogonal;

    GLfloat tx = -((proj.right + proj.left) / (proj.right - proj.left));
    GLfloat ty = -((proj.top + proj.bottom) / (proj.top - proj.bottom));
    GLfloat tz = -((proj.farClip + proj.nearClip) / (proj.farClip - proj.nearClip));

    project_matrix = Matrix4(
        2 / (proj.right - proj.left), 0, 0, tx,
        0, 2 / (proj.top - proj.bottom), 0, ty,
        0, 0, -2 / (proj.farClip - proj.nearClip), tz,
        0, 0, 0, 1
    );
}
```

以上 translative, scaling, rotate, setViewingMatrix, setOrthogonal 這幾個 fuction 都是依照老師講義中刻出來的。

```
// [TODO] compute persepective projection matrix
void setPerspective()
{
    cur_proj_mode = Perspective;

    GLfloat f = 1 / tan (proj.fovy / (360/3.14));

    project_matrix = Matrix4(
        f / proj.aspect, 0, 0, 0,
        0, f, 0, 0,
        0, 0, (proj.farClip + proj.nearClip) / (proj.nearClip - proj.farClip), (2 * proj.farClip * proj.nearClip) / (proj.nearClip - proj.farClip),
        0, 0, -1, 0
    );
}
```

setPerspective 我原本也是依照老師講義中的打，結果不知道為什麼顯示會上下左右顛倒。上網查發現有人說，把 f 的 fovy/2 改成 fovy/(360/3.14)就好了，結果還真的好了，但我還是不知道為什麼，請助教解惑，謝謝！（還是其實我是其他地方打錯了？？應該用 fovy/2 才對？？）

```
// Call back function for window reshape
void ChangeSize(GLFWwindow* window, int width, int height)
{
    glViewport(0, 0, width, height);
    // [TODO] change your aspect ratio

    if (cur_proj_mode == Perspective) {
        proj.aspect = (float)width / (float)height;
        setPerspective();
    }
    else {
        setOrthogonal();
    }
}
```

ChangeSize 的部分，我看到助教 demo 時，在 Orthogonal mode 並沒有處理形變問題，所以應該是處理 Perspective mode 就好了？？於是我就把它分開寫成只有 perspective mode 才要處理 proj.aspect。

```
void drawPlane()
{
    GLfloat vertices[18]{ 1.0, -0.9, -1.0,
        1.0, -0.9,  1.0,
        -1.0, -0.9, -1.0,
        1.0, -0.9,  1.0,
        -1.0, -0.9,  1.0,
        -1.0, -0.9, -1.0 };

    GLfloat colors[18]{ 0.0,1.0,0.0,
        0.0,0.5,0.8,
        0.0,1.0,0.0,
        0.0,0.5,0.8,
        0.0,0.5,0.8,
        0.0,1.0,0.0 };

    // [TODO] draw the plane with above vertices and color
    Matrix4 MVP;
    GLfloat mvp[16];
    MVP = project_matrix * view_matrix;
    mvp[0] = MVP[0];    mvp[4] = MVP[1];    mvp[8] = MVP[2];    mvp[12] = MVP[3];
    mvp[1] = MVP[4];    mvp[5] = MVP[5];    mvp[9] = MVP[6];    mvp[13] = MVP[7];
    mvp[2] = MVP[8];    mvp[6] = MVP[9];    mvp[10] = MVP[10];  mvp[14] = MVP[11];
    mvp[3] = MVP[12];   mvp[7] = MVP[13];   mvp[11] = MVP[14];  mvp[15] = MVP[15];


    glGenBuffers(1, &VBO);
    glGenBuffers(1, &quad.p_color);
    glGenVertexArrays(1, &VAO);
    glBindVertexArray(VAO);

    glBindBuffer(GL_ARRAY_BUFFER, VBO);
    glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices, GL_STATIC_DRAW);
    glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(float), (void*)0);
    glEnableVertexAttribArray(0);

    glBindBuffer(GL_ARRAY_BUFFER, quad.p_color);
    glBufferData(GL_ARRAY_BUFFER, sizeof(colors), colors, GL_STATIC_DRAW);
    glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(float), (void*)0);
    glEnableVertexAttribArray(1);

    glUniformMatrix4fv(iLocMVP, 1, GL_FALSE, mvp);
    glDrawArrays(GL_TRIANGLES, 0, 3*2);
}
```

drawPlane 我是參考討論區的問題才做出來的，因為助教說先觀察 RenderScene 裡面怎麼畫，然後先 bind VAO，然後再 call glDrawArrays，所以我就參考 RenderScene 的方式來實作，但不知道是不是打得有點太冗了，可能有更好的作法？？

```
static void cursor_pos_callback(GLFWwindow* window, double xpos, double ypos)
{
    // [TODO] cursor position callback function
    if (mouse_pressed) {
        switch (cur_trans_mode)
        {
        case GeoTranslation:
            models[cur_idx].position.x = (xpos - WINDOW_WIDTH / 2) / (WINDOW_WIDTH / 2);
            models[cur_idx].position.y = (WINDOW_HEIGHT / 2 - ypos) / (WINDOW_HEIGHT / 2);
            break;
        case GeoRotation:
            models[cur_idx].rotation.x = (WINDOW_HEIGHT / 2 - ypos) / (WINDOW_HEIGHT / 2);
            models[cur_idx].rotation.y = (xpos - WINDOW_WIDTH / 2) / (WINDOW_WIDTH / 2);
            break;
        case GeoScaling:
            models[cur_idx].scale.x = (xpos - WINDOW_WIDTH / 2) / (WINDOW_WIDTH / 2);
            models[cur_idx].scale.y = (WINDOW_HEIGHT / 2 - ypos) / (WINDOW_HEIGHT / 2);
            break;
        case ViewCenter:
            main_camera.center.x = (xpos - WINDOW_WIDTH / 2) / (WINDOW_WIDTH / 2);
            main_camera.center.y = (WINDOW_HEIGHT / 2 - ypos) / (WINDOW_HEIGHT / 2);
            setViewingMatrix();
            break;
        case ViewEye:
            main_camera.position.x = (xpos - WINDOW_WIDTH / 2) / (WINDOW_WIDTH / 2);
            main_camera.position.y = (WINDOW_HEIGHT / 2 - ypos) / (WINDOW_HEIGHT / 2);
            setViewingMatrix();
            break;
        case ViewUp:
            main_camera.up_vector.x = (xpos - WINDOW_WIDTH / 2) / (WINDOW_WIDTH / 2);
            main_camera.up_vector.y = (WINDOW_HEIGHT / 2 - ypos) / (WINDOW_HEIGHT / 2);
            setViewingMatrix();
            break;
        default:
            break;
        }
    }
}
```

處理 xpos, ypos，讓它會在-1 跟 1 之間。然後拖動圖形時，我會讓圖形在鼠標的位置。