310605009 吳公耀 HW1

做了 TODO 的部分

1.

```
for (auto& cube : cubes)
{
  glPushMatrix();
  cube->setupModel();

  cube->draw();
  glPopMatrix();
}
```

先將 27 個前面塞在 cubes 裡面的物件上色以及建立。需使用 glPushMatrix 記住自己現在的位置以及 glPopMatrix 回到之前記住的位置，第一次用不知道怎麼用以及放的位置，正個物體會不成立方體。

2.

```
glm::mat4 rotation_matrix = glm::mat4_cast(this->rotation);
glm::mat4 trans_matrix = glm::translate(this->translation, this->position);
const float* ptr1 = glm::value_ptr(rotation_matrix);
const float* ptr2 = glm::value_ptr(trans_matrix);
glMultMatrixf(ptr1);
glMultMatrixf(ptr2);
```

之後是 Update model matrix 的部分，這邊在 update 的時候遇到很多問題，一放面試不太了解 API 用法，另一方面是對空間矩陣以及 vec3 跟 mat4 的用法不太熟悉，如果再 ptr1 ptr2 放反的話在之後做整個魔術方塊的旋轉的話無法正確地對軸做旋轉。

3.

```
glBegin(GL_TRIANGLE_STRIP);
// Green, top
glColor3f(0.0f, 1.0f, 0.0f);
glNormal3f(0.0f, 1.0f, 0.0f);

glVertex3f(-1.0f, 1.0f, -1.0f);
glVertex3f(-1.0f, 1.0f, 1.0f);
glVertex3f(1.0f, 1.0f, -1.0f);
glVertex3f(1.0f, 1.0f, 1.0f);


glEnd();

glBegin(GL_TRIANGLE_STRIP);
// Blue, bottom
glColor3f(0.0f, 0.0f, 1.0f);
glNormal3f(0.0f, -1.0f, 0.0f);
glVertex3f(-1.0f, -1.0f, -1.0f);
glVertex3f(-1.0f, -1.0f, 1.0f);
glVertex3f(1.0f, -1.0f, -1.0f);
glVertex3f(1.0f, -1.0f, 1.0f);

glEnd();

glBegin(GL_TRIANGLE_STRIP);
// Red, right
glColor3f(1.0f, 0.0f, 0.0f);
glNormal3f(1.0f, 0.0f, 0.0f);
glVertex3f(1.0f, 1.0f, 1.0f);
glVertex3f(1.0f, -1.0f, 1.0f);
glVertex3f(1.0f, 1.0f, -1.0f);
glVertex3f(1.0f, -1.0f, -1.0f);

glEnd();
```

```
glBegin(GL_TRIANGLE_STRIP);

// Orange, left
glColor3f(1.0f, 0.5f, 0.0f);
glNormal3f(-1.0f, 0.0f, 0.0f);

glVertex3f(-1.0f, 1.0f, -1.0f);
glVertex3f(-1.0f, -1.0f, -1.0f);
glVertex3f(-1.0f, 1.0f, 1.0f);
glVertex3f(-1.0f, -1.0f, 1.0f);
glEnd();

glBegin(GL_TRIANGLE_STRIP);

// Yellow, front
glColor3f(1.0f, 1.0f, 0.0f);
glNormal3f(0.0f, 0.0f, 1.0f);  //

glVertex3f(-1.0f, 1.0f, 1.0f);
glVertex3f(-1.0f, -1.0f, 1.0f);
glVertex3f(1.0f, 1.0f, 1.0f);
glVertex3f(1.0f, -1.0f, 1.0f);  /
glEnd();

glBegin(GL_TRIANGLE_STRIP);


//White, back
glColor3f(1.0f, 1.0f, 1.0f);
glNormal3f(0.0f, 0.0f, -1.0f);


glVertex3f(1.0f, 1.0f,- 1.0f);
glVertex3f(1.0f, -1.0f, -1.0f);
glVertex3f(-1.0f, 1.0f, -1.0f);
glVertex3f(-1.0f, -1.0f, -1.0f);
```

這邊是將 27 個立方體上色的部分，遇到的問題包括沒有對每個面的上色做 glBegin/glEnd 而產生漸層，以及店的位置，需想像成透過 4 格點化兩個三角形，若順序有錯就會沒辦法化成正確的正方形，若 glBegin 使用不同的 enum 點的順序也會有不同規則。

4.

```
constexpr glm::vec3 original_front(0, 0, -1);
constexpr glm::vec3 original_up(0, 1, 0);
glm::vec3 rotated_front = this->rotation * original_front;
glm::vec3 rotated_up = this->rotation * original_up;
glm::vec3 cameraRight = glm::cross(rotated_up, rotated_front);

viewMatrix = glm::identity<glm::mat4>();
viewMatrix = glm::lookAt(this->position, this->position + rotated_front, rotated_up);
```

再來是 updateView 的部分，使用:lookAt 分別需要 camera 座標、camera 瞄準的座標、相機頭頂的方向向量，因此用一個 quaternion 紀錄轉的情況(this.rotation)算了一個 rotated_front 就是轉了之後瞄準的方向，剛開始不太懂以為需要算 cross 出來的解果但如果用 lookAt 就用不到，但如果要純手算 view matrix 的話可能會有問題。

5.

```
void QuaternionCamera::updateProjection(float aspectRatio) {
    constexpr float FOV = glm::radians(45.0f);
    constexpr float zNear = 0.1f;
    constexpr float zFar = 100.0f;
    projectionMatrix = glm::identity<glm::mat4>();
    projectionMatrix= glm::perspective(FOV, aspectRatio, zNear, zFar);
```

用此透視投影方法表示这个投影需要用到的變數包括垂直視野，以弧度為單位、縱橫比(取決於你的窗戶的大小給的比例)、靠近剪切平面、遠剪裁平面

6.

```
for (auto& cube : cubes) {
  switch (key) {
    case 'U':
       cube->rotate(Axis::X);
     break;
    case 'R':
      if (cube->getPosition(Axis::X) == 1) {

        cube->rotate(Axis::X);
      }
      break;
    case 'F':
      if (cube->getPosition(Axis::X) == 0) {
        cube->rotate(Axis::X);
      }
      break;
    case 'V':
      if (cube->getPosition(Axis::X) == -1) {
        cube->rotate(Axis::X);
      }
      break;
```

```
    case 'J':
      cube->rotate(Axis::Y);
      break;
    case 'T':
      if (cube->getPosition(Axis::Y) == 1) {
        cube->rotate(Axis::Y);
      }
      break;
    case 'G':
      if (cube->getPosition(Axis::Y) == 0) {
        cube->rotate(Axis::Y);
      }
      break;
    case 'B':
      if (cube->getPosition(Axis::Y) == -1) {
        cube->rotate(Axis::Y);
      }
      break;
```

```
    case 'M':
      cube->rotate(Axis::Z);
      break;
    case 'Y':
      if (cube->getPosition(Axis::Z) == 1) {
        cube->rotate(Axis::Z);
      }
      break;
    case 'H':
      if (cube->getPosition(Axis::Z) == 0) {
        cube->rotate(Axis::Z);
      }
      break;
    case 'N':
      if (cube->getPosition(Axis::Z) == -1) {
        cube->rotate(Axis::Z);
      }
      break;
  }
```

URFV 為對 X 做旋轉，第一個為整魔術方塊做旋轉，剩下三個分別為 1.-1.0
JTGB 為對 Y 做旋轉，第一個為整魔術方塊做旋轉，剩下三個分別為 1.-1.0
MYHN 為對 Z 做旋轉，第一個為整魔術方塊做旋轉，剩下三個分別為 1.-1.0

最後就是實作轉轉的方法，剛開始沒有使用 cube->getPosition(Axis::X)取的虛選轉的目標需要大量的程式碼因此看了一下給的 hint 改了選取該選轉的方塊