BASIC：

动态效果见附视频文件 hw5.mp4。

1. 投影(Projection):
   - 把上次作业绘制的 cube 放置在(-1.5, 0.5, -1.5)位置，要求 6 个面颜色不一致
     见演示视频
   - 正交投影(orthographic projection): 实现正交投影，使用多组(left, right, bottom, top, near, far)参数， 比较结果差异
     演示视频中通过调节 ortho 函数的六个参数，可以看出调节 left/right 时使得图形在水平上位移和形变，top/bottom 则产生垂直方向上的位移和形变，调节 near/far 的深度信息时，物体的深度需要再 near-far 的范围内，否则物体会被截取或者不可见，而且可以发现，正交投影后的标准设备中显示的是左手坐标系。
   - 透视投影(perspective projection)：实现透视投影，使用多组参数，比较结果差异
     演示视频中通过调节 perspective 函数的 4 个参数，可以看出调节 foxy 时使得视野范围变化，aspect 调整的是投影平截头体的宽高比，调节 near/far 的深度信息时，物体的深度需要再 near-far 的范围内而且物体随着距离远近大小变化时，透视投影在 Opengl 中标准设备变换后显示的是右手坐标系。

2. 视角变换(View Changing):
   - 把 cube 放置在(0, 0, 0)处，做透视投影，使摄像机围绕 cube 旋转，并且时刻看着 cube 中心
     见演示视频，摄像机在 XOZ 平面上围绕 cube 旋转

3. 在 GUI 里添加菜单栏，可以选择各种功能。 Hint: 使摄像机一直处于一个圆的位置，可以参考以下公式：
   camPosX=sin(clock()/1000.0)*Radius;
   camPosZ=cos(clock()/1000.0)*Radius;
   原理很容易理解，由于圆的公式 a^2+b^2=1 ，以及有 sin(x)^2+cos(x)^2=1 ，所以能保证摄像机在 XoZ 平面的一个圆上。
   菜单栏能调节投影的各种参数。

4. 在现实生活中，我们一般将摄像机摆放的空间 View matrix 和被拍摄的物体摆设的空间 Model matrix 分开，但是在 OpenGL 中却将两个合二为一设为 ModelView matrix，通过上面的作业启发，你认为是为什么呢？在报告中写入。(Hints：你可能有不止一个摄像机)
   当多个摄像机要从不同位置拍摄物体，需要切换视角的时侯，每个摄像机的位置和角度不同，各个摄像机的 ModelView 变换矩阵整合一起针对同一个坐标系进行变换利于直观操作和理解。

Bonus

1. 实现一个 camera 类，当键盘输入 w,a,s,d ，能够前后左右移动；当移动鼠标，能够视角移动("look around")， 即类似 FPS(First Person Shooting)的游戏场景
   见演示视频

代码思路解析

附代码文件（本次作业的核心代码在 hw5.cpp、camera.h 文件内）

1. 6 个面颜色不同的立方体画法
   使用 24 个顶点坐标，每个面由四个相同颜色的顶点坐标组成，配合索引数组 EBO，绘制出立方体。
2. 摄像机类定义，参考 OpenGL 官网的摄像机类的定义实现代码
3. 变换矩阵主要分成模型矩阵 model,观察矩阵 view,投影矩阵 projection 三个，传入顶点着色器中。

```
const char *vertexShaderSource = "#version 330 core\n"
    "layout (location = 0) in vec3 aPos;\n"
    "layout (location = 1) in vec3 aColor;\n"
    "uniform mat4 model;\n"
    "uniform mat4 view;\n"
    "uniform mat4 projection;\n"
    "out vec3 ourColor;\n"
    "void main()\n"
    "{\n"
    "   gl_Position = projection * view * model * vec4(aPos, 1.0f);\n"
    "   ourColor = aColor;\n"
    "}\0";
```