22355082-王宇和-个人报告

个人贡献

制作昼夜系统

思路如下：

1. 添加太阳立方用于指示太阳方位
2. 修改cube的裁切，使太阳立方可视
3. 给太阳立方增加随时间进行的移动
4. 修改天空盒的着色器，修改天空颜色
5. 调整太阳立方的位置、大小
6. 丰富太阳立方细节
7. 增加月亮立方用以指示月亮方位
8. 增加时间加快功能

具体实现

添加太阳立方用于指示太阳方位

复用项目中的`Cube`类，在`render.h`中添加太阳立方属性。

```
sunCube = new Cube(dummyTextures, main_light.GetLightboxShader());
```

修改cube的裁切，使太阳立方可视

在渲染后发现太阳立方看不见，通过调整太阳立方的位置发现是因为在`cube`的`Draw`方法中被裁切，调整投影矩阵如下（主要修改第四个`far`参数）：

```
glm::mat4 projection = glm::perspective(
    glm::radians(camera.Zoom),
    800.0f / 600.0f,
    0.1f,
    20000.0f
);
```

发现调整后太阳立方可视。

给太阳立方增加随时间进行的移动

与海浪的渲染共用同一个时间系统，即通过渲染帧数计算时间，将时间传给`renderer`。

```
float lastFrame = 0.0f;
while (!glfwWindowShouldClose(window))
{
    float currentFrame = static_cast<float>(glfwGetTime());
    deltaTime = currentFrame - lastFrame;
    lastFrame = currentFrame;
    ...
    renderer.RenderFrame(camera, static_cast<float>(screenWidth),
static_cast<float>(screenHeight), currentTime);
}
```

在`renderer`中实现太阳立方的移动：（见`UpdateDayNight`）

基本思路是通过固定角速度来计算并更新太阳位置、光线方向。

修改天空盒的着色器，修改天空颜色

有了太阳移动后，发现由于天空盒有贴图，并不受光线影响，故修改天空盒的着色器，通过将光照与天空盒`mix`的方式改变天空颜色，使昼夜系统看起来更真实。

基本思路是定义几个关键时刻的天空基色、根据时间指标所在区间做多段插值、叠加在原始天空盒上。（见`skybox.fs`）

例如：

```
// 将 skyTint 看作“环境色”，用 mix 控制其影响程度
// t 越接近白天，越多保留原始贴图；越接近夜晚，越强烈上色。
float tintStrengthNight = 0.8; // 夜晚对贴图影响更大
float tintStrengthDay   = 0.3; // 白天则弱一些
float tintStrength = mix(tintStrengthNight, tintStrengthDay, t);
vec3 tinted = mix(texCol, skyTint, tintStrength);
```

调整太阳立方的位置、大小

原本的太阳立方绕z轴做运动，使物体光影变化看起来效果不佳，通过调整`UpdateDayNight`方法调整太阳立方的旋转轴、距世界中心的位置、大小。

例如：

```
float angle = t * glm::radians(360.0f) - glm::radians(90.0f);
float radius = 7500.0f;
// 先在 XZ 平面
glm::vec3 basePos(
    radius * cos(angle),
    0.0f,
    radius * sin(angle)
);
// 绕 X 轴稍微抬一点（控制高度变化方向）
float tiltX = glm::radians(60.0f);
// 绕 Y 轴决定“从哪个方位升起”
float tiltY = glm::radians(30.0f);
```

丰富太阳立方细节

为太阳立方增加着色器，使其看起来更立体。

```
sunCube = new Cube(dummyTextures, main_light.GetLightboxShader());
```

增加月亮

通过在`UpdateDayNight`方法中增加月亮立方，并使其与太阳立方位置相反，同时调整颜色，修改太阳、月亮立方绘制逻辑。

例如：

```
if (sunAbove) // 画太阳
{
    ...
}
else // 画月亮（与太阳相对）
{
    glm::mat4 modelMoon(1.0f);
    modelMoon = glm::translate(modelMoon, moonPos);
    modelMoon = glm::scale(modelMoon, glm::vec3(150.0f)); // 略小一点
    lightShader.use();
    lightShader.setVec3("lightColor", moonColor);
    sunCube->Draw(const_cast<Camera&>(camera), noClip, modelMoon, false);
}
```

增加时间加快功能

增加按键F的监听，并给原本的时间系统增加一层`worldTime`的包装，从而实现时间的加速（原本的时间基于渲染帧数计算，需要增加系数来修正）

```
float worldTime = 0.0f;       // 昼夜系统用的“世界时间”
float timeScale = 0.2f;       // 默认时间倍率（<1 表示比真实时间慢）
bool fastTime = false;        // 是否处于加速状态
// 按 F 键切换时间快/慢
if (glfwGetKey(window, GLFW_KEY_F) == GLFW_PRESS && !fastTime)
{
    fastTime = true;
    timeScale = 5.0f;    // 快进倍率
}
if (glfwGetKey(window, GLFW_KEY_F) == GLFW_RELEASE && fastTime)
{
    fastTime = false;
    timeScale = 0.2f;    // 恢复到慢速
}
```

增加功能后。按下F键海浪的运动、天体的运动速度均会加快。