## 使用说明：

**功能介绍：**

1. 以红绿蓝三色的箭头绘制x y z坐标轴，以原点为中心随机绘制100个位置、大小、亮度各异的星星，颜色均为白色，且亮度随时间按正弦变化。每个星星均以随机速度在xOz平面内做匀速圆周运动
2. 每个实体都贴有纹理
3. 在原点处绘制一个太阳，然后绘制8个大小、轨迹随机的行星，土星有光环。然后绘制围绕地球公转的月球，一定比其他星球小，每个星球都在自转
4. 按下回车，从地球发射一个飞船。飞船中有两个宇航员Anon与Soyo
5. 太阳发出点光源照亮其他星球与飞船，飞船顶部有一个聚光灯照亮前方星球，飞船内部有一个点光源照亮内部，后两个灯光可自由开关
6. 使用欧拉角与子坐标两种方式进行视点控制
7. 共三种摄像机：上帝摄像机，飞船摄像机，Anon摄像机
8. 上帝摄像机下，可选中某个星球，使飞船自动向那个星球航行，或者视点绑定到所选星球，可方向键控制飞船飞行
9. 飞船摄像机下，通过鼠标控制飞船飞行，通过方向键控制Anon移动
10. 飞船摄像机下，可选择两种视角，常规视角下可观察飞船前方星球，内部视角下可观察飞船内部
11. Anon摄像机下，鼠标与WASD控制Anon移动，方向键控制Soyo移动
12. Anon摄像机下，可选择三种视角：第一人称、第三人称、第二人称（摄像机面对Anon）
13. 实时检测飞船与星球的碰撞、宇航员之间的碰撞以及宇航员与舱壁的碰撞，并将飞船与Anon的碰撞信息显示在屏幕上。宇航员碰撞时无法移动。

**操作说明：**

ESC退出

按下F1切换填充模式与线框模式

按下F2切换欧拉角视点控制与子坐标视点控制

按下F3显示碰撞箱

按下F4输出调试信息（相机当前位置，欧拉角，前方向量与上方向量等）

按下F5切换摄像机，切换顺序为：上帝摄像机->飞船摄像机->Anon摄像机，循环切换

按下F6切换视角

按下F12时间静止，所有星球停止移动但飞船不受影响，便于调试

鼠标移动控制视角移动

未发射飞船时，按下回车，从地球发射一个飞船

上帝摄像机下，WSAD控制相机移动，鼠标控制方向，方向键控制飞船移动（使用↑与↓控制飞船俯仰，使用←与→控制飞船偏航），R控制相机加速，左ctrl控制相机减速；J降低相机灵敏度，K增加相机灵敏度

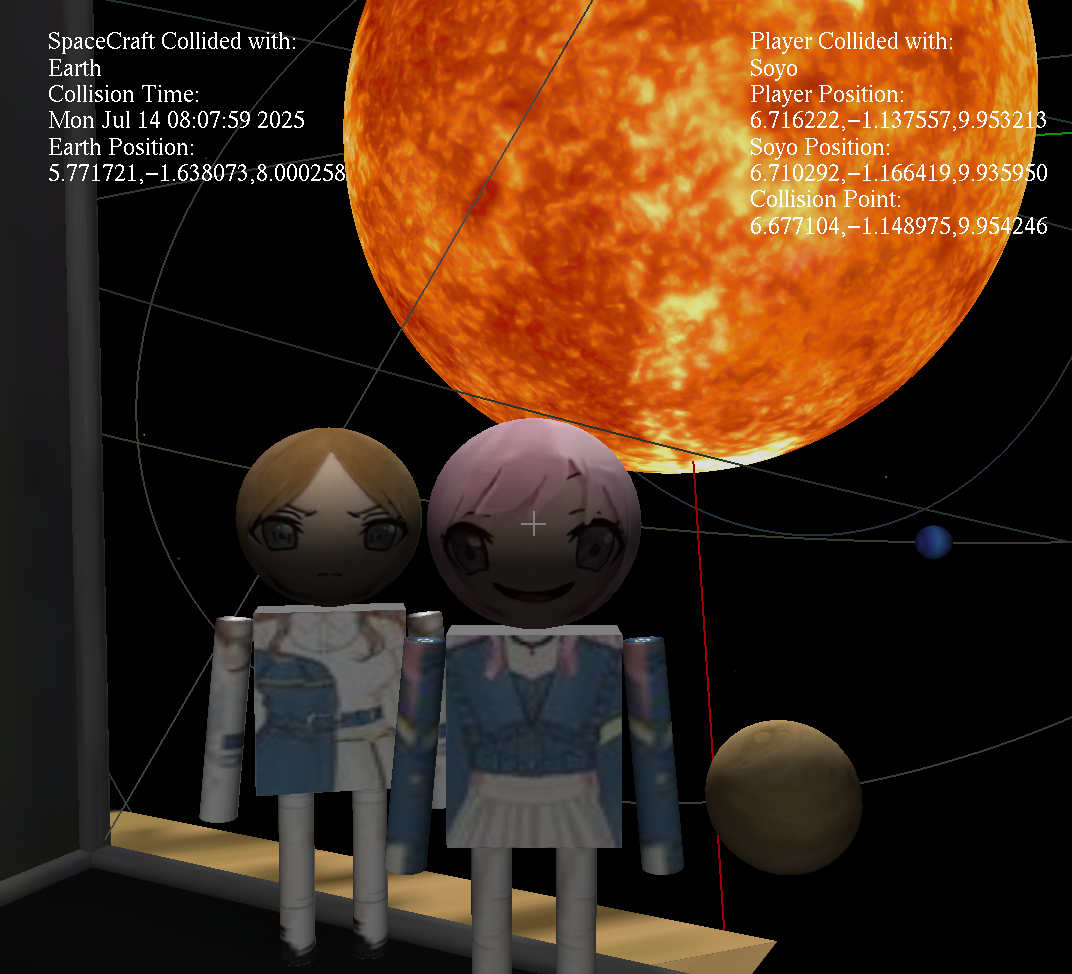
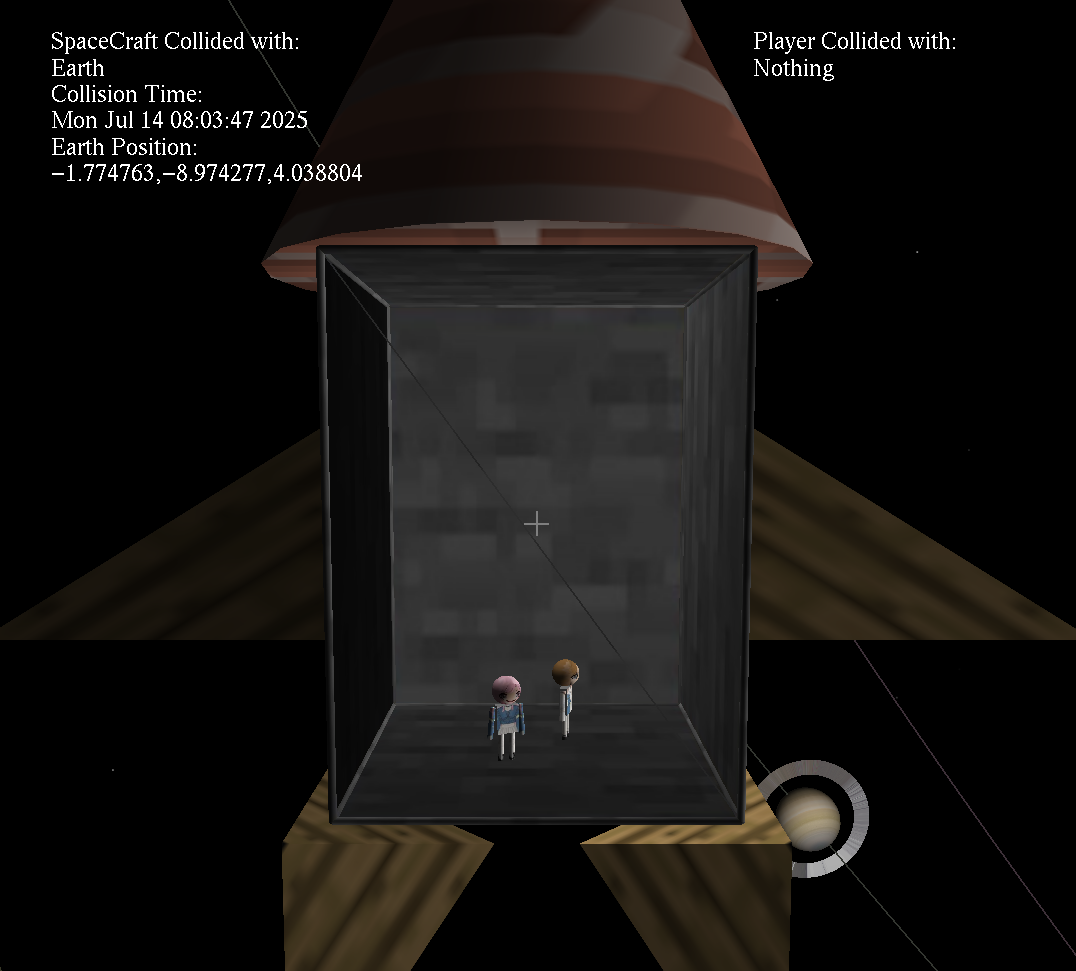
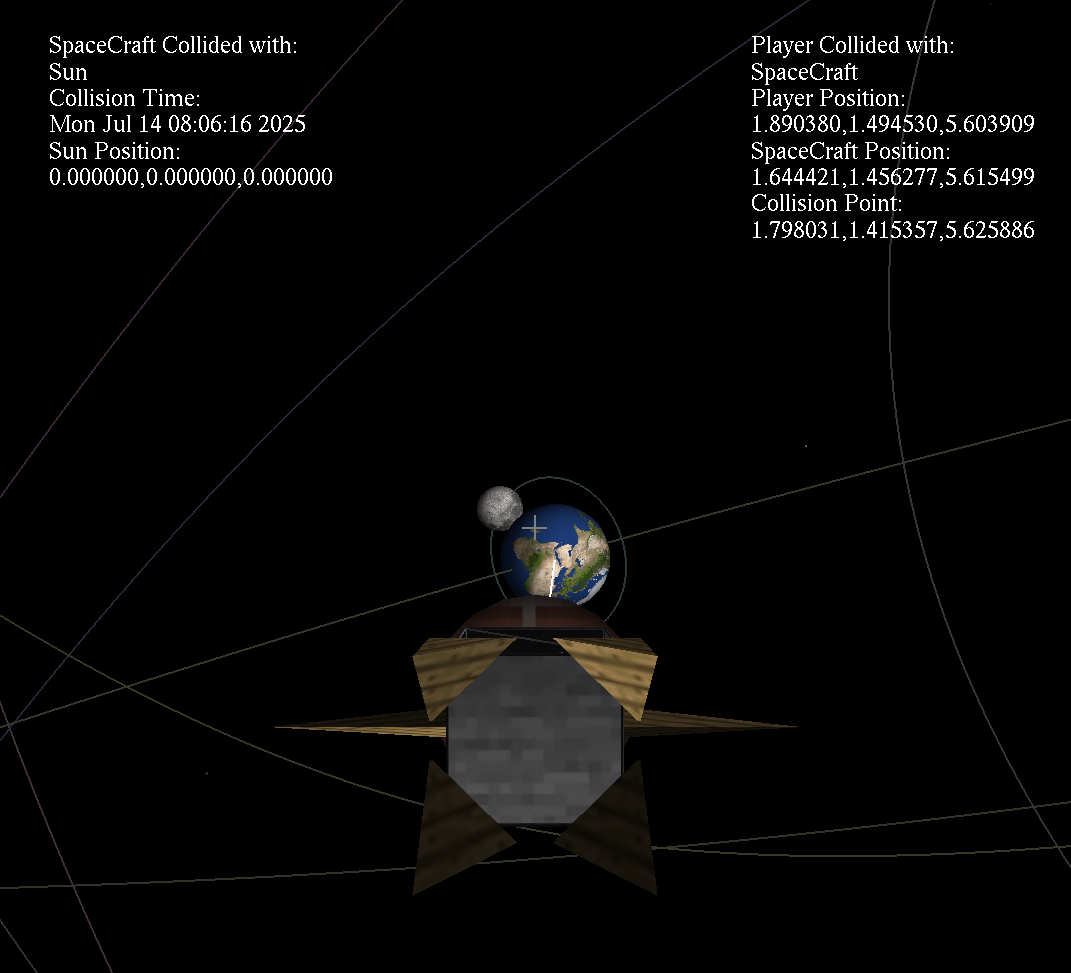
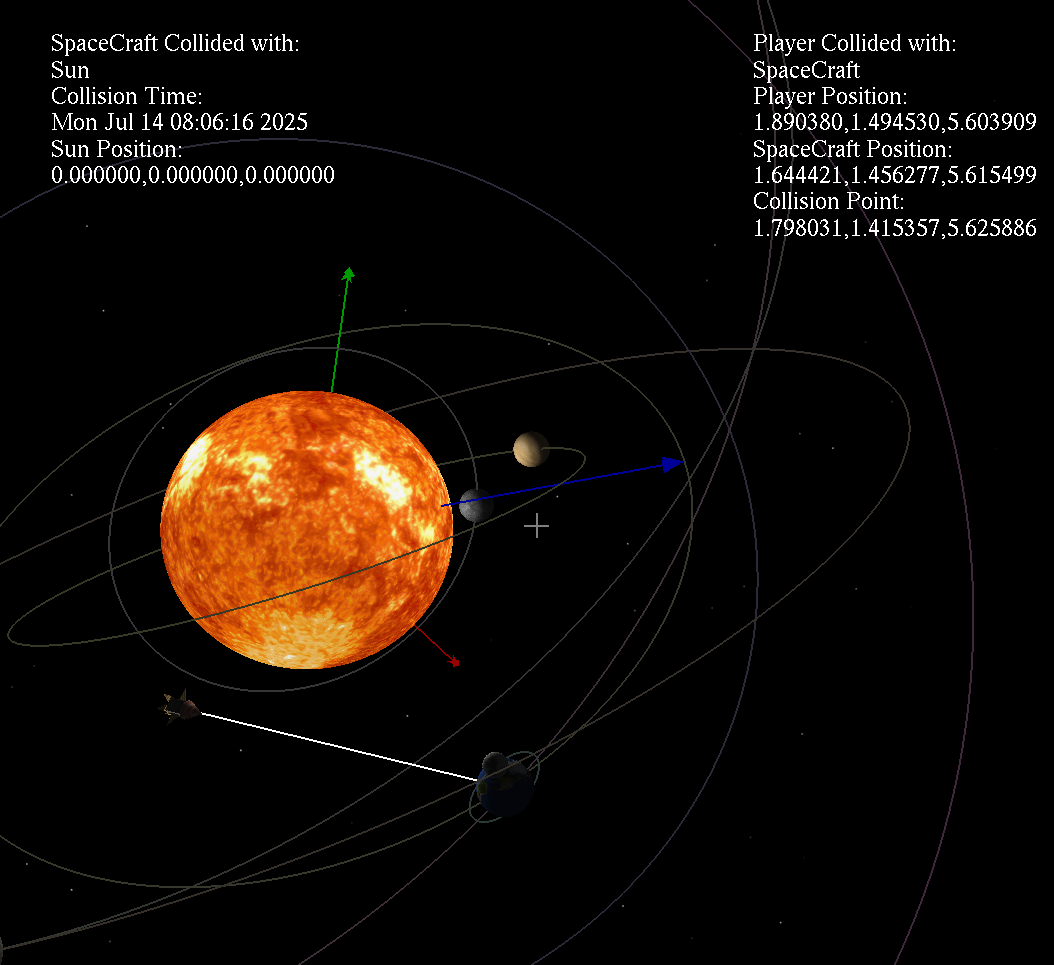
上帝摄像机下，鼠标左键选中某个星球，按下回车可使飞船自动飞向那个星球，使用方向键改变航向后取消自动飞行。按下空格可视点绑定到所选星球上方，取消选择后按下空格取消绑定。

飞船摄像机下，鼠标控制飞船前进方向，方向键控制Anon移动（使用↑与↓控制行走，使用←与→控制转向），可切换常规视角与内部视角

Anon摄像机下，WASD控制Anon移动，鼠标控制方向，方向键控制Soyo移动，可切换第一人称、第三人称与第二人称视角

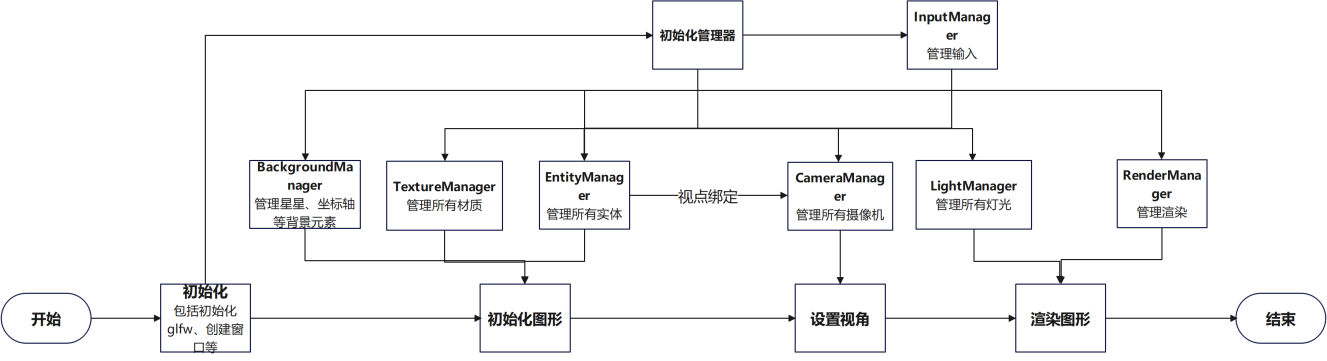
任意摄像机下，F使所有宇航员原地踏步，PgUp使飞船加速，PgDown使飞船减速，按下1关闭飞船内部灯光，按下2关闭飞船顶部聚光灯

**运行截图：**



## 设计说明：

**程序流程图：**



**关键函数说明：**

bool TextureManager::loadTexture(const char\* ID)

用于加载纹理

void LightManager::initSunLight()

初始化太阳光

void ViewPoint::handleMouseOffset(double dx, double dy, AbstractEntity\* entity)

绑定视点后，视点接受鼠标位移，同时控制摄像机与实体

bool Cuboid::checkCollision(const Cuboid& other)

检测长方体之间是否发生碰撞

static inline void drawCollisionInfo()

使用glut库在屏幕上绘制2D文字，输出碰撞信息

**设计思路说明：**

使用的库为GLFW与GLUT

各图形的顶点与颜色等数据采用CVector类表示，矩阵采用CMatrix。

所有实体用到的图元均继承自Geometry类，Geometry类中现在添加了法向量数组与纹理坐标数组，用来接受光照与绘制纹理

实体Entity现在继承自AbstractEntity，抽象实体如摄像机与视点直接继承自AbstractEntity

带有视点的实体（事实上目前所有实体都有视点）同时继承自Entity与ViewPoint两个类

抽象实体Camera未绑定视点时，使用Camera类中的方法控制视角移动；绑定视点后，实时与视点中的front与up等数据同步

在TextureManager中加载与管理所有纹理

在LightManager中初始化与更新所有光照

在EntityManager中存储与管理所有实体，包括星体、飞船、宇航员

在BackgroundManager中管理星星等背景元素。

在InputManager中管理键鼠状态与读取用户输入。

在CameraManager中管理视角变化并存储视图矩阵。

在RenderManager中管理所有渲染逻辑