CG大作业说明文档

景致 515030910301

Idea

- 最终定型：
  以萤火虫模拟为主的粒子系统。
  风格可由 low poly 风格切换为真实感风格。
- 特点：
  动态多光源，
  次级粒子系统，
  萤火虫群（swarm insects）行为仿真，
  基于帧缓冲（FBO）和泛光（Bloom）技术实现了萤火虫的荧光效果，
  3D 立体音效
- Version -1：
  最开始的构思是完全另一番景象。想要做一个扩展自 [纪念碑谷] 的更有游戏性的场景， [纪念碑谷] 的游戏模式加上可以自调的 camera ，来获得更多视觉错觉的可能性，然后通过玩家达成不同的错觉角度来触发不同的粒子系统。但后来觉得这个 idea 似乎并不是以粒子系统为主，有点跑题，于是就有了现在的 version。（ [纪念碑谷 plus] 的 idea 留在下学期的人机交互或者游戏设计课上实现）

操作方法

- 萤火虫自动逐渐产生，默认会被吸引到摄像机附近。
- 鼠标滑动控制摄像机观察方向，W、S、A、D 控制摄像机移动。
- 按下 R 键取消摄像机对萤火虫群的吸引，再按下 E 键恢复吸引。
- 按下 C 键清除场景中所有萤火虫。
- 按下 P 键让场景中所有萤火虫停止更新和运动（时间停滞），按下 O 键恢复运动。
- 按下 Q 键切换到真实感模型，在真实感的场景里按下 T 键触发萤火虫的消亡过程（次级粒子系统）。

代码架构

可编程管线（core-profile）实现。用到的库有：

```
GLEW
GLFW
glm：数学运算、线性代数等
SOIL：读取贴图
assimp：加载各种格式的模型
irrKlang：加载 2D 及 3D 声音
```

使用语言为 C++ 和 GLSL

- main.cpp
  负责创建窗口，播放 BGM，接收鼠标键盘的输入，实例化 Game 类创建 MyGame 对象，在窗口的主循环里调用 Game 类的 Update、Render等方法。
- game.h/game.cpp
  游戏的主体。负责实例化摄像机类，编译 shader 文件并存入资源管理池，初始化帧缓冲。其中的 Update 方法调用粒子系统的 Update 方法，Render 方法调用粒子系统和场景类的 Draw 方法将所有场景绘入帧缓冲。
- particle_generator.h/particle_generator.cpp
  粒子系统类。其中有粒子个体结构体（Particle）的定义。其中的 Update 方法更新所有粒子的状态（包括速度、加速度、位置、颜色、寿命等），Draw 方法绘制所有粒子并播放每个粒子的音效。
- environment.h/environment.cpp
  环境模型类，包括天空盒和环境模型。负责绘制天空盒和环境模型。
- 其他工具类
  Camera.h： 封装的摄像机类。
  shader.h：封装的 shader 处理器类。负责读取 shader 文件并编译，封装了一些给 shader 传变量的方法。
  texture.h：封装的纹理处理器类。负责读取纹理贴图，生成纹理对象。
  resource_manager.h：封装的资源池类。负责管理 shader 和 texture 的集合。其中方法都为静态方法，无需实例化直接调用。
  model.h/Mesh.h：封装的模型处理类。负责分析 assimp 库读取的模型数据并完成模型绘制。

实现过程中的点滴

- 虫群运动仿真
  参考了几篇论文但是都没有得到很 practical 的算法，最终决定照着 YouTube 上的几个视频效果来闭门造车。
  最终的模型是两个 factor 叠加：
  首先是**随机加速度**，用于模拟小虫无规则运动。这个加速度不是每一帧变化的，而是隔一段时间变化一次，期望能达到小虫一会儿向一个方向飞一会儿又向另一个方向飞的效果。
  第二个是**恒定大小指向吸引区的速度**，用于模拟小虫被某个点（以点为球心的球状区域）吸引的整体趋势。当小虫位于吸引区范围外，每一帧为它的速度加上一个恒大指向吸引区球心的速度；当小虫位于吸引区内部，只受到随机加速度的影响。
  两部分 factor 的参数进行了调节，达到更加拟真的效果。每个小虫个体有自己独特的**约束系数**（随机产生），当约束系数更大的时候，小虫会更快的被吸引区吸引，并紧紧围绕吸引区活动；当约束系数更小的时候，小虫会显得更“自由”，甚至可以暂时脱离吸引区的束缚飞向其他区域。
  另外还通过增大粒子系统的 dt 参数来达到减慢小虫运动速度的效果，因为考虑到萤火虫飞得太快视觉效果很缭乱不是很好。
- 动态多光源
  场景类（environment）持有粒子系统类（particle system）的指针，在每一帧的绘制中实时地取得当前存活地粒子地数量、位置并作为光源信息传给场景的片段着色器（environment.frag）进行多光源绘制。

在光照渲染渲染模型上，采用了 Phong 氏模型。
环境光分为固定的全局环境光和随着萤火虫数量而改变强度的局部环境光。
漫反射光和镜面反射光采用了不同的颜色值。在黄绿色（GreenYellow）的基础上，漫反射光更偏黄，镜面反射光更偏绿，来使得萤火虫的荧光效果更逼真。
所有的局部环境光、漫反射光和镜面反射光都考虑了随距离衰减，萤火虫由远及近的效果更好。

- 动态多光源的优化
  多光源做好后发现性能堪忧，当场景中的萤火虫数目（即光源数目）超过 100 个的时候，渲染效率低下导致摄像头移动的时候明显感觉帧数很低。
  于是重新设计了**优化算法**：因为最终大多数粒子都会围绕 camera 附近运动，所以考虑在 camera 附近采取等效的策略。当粒子个体距离 camera 较远时，还是按照原来的策略按照独立点光源计算效果并进行渲染； 而对于那些距离 camera 较近的个体，把它们等效成一个大的点光源，这个等效点光源的亮度由当前范围内的个体数目决定。
  在新的算法下，渲染效率明显提高。旧算法下场景里有多少只萤火虫就要计算多少个光源。新算法下如果 camera 静止，平均需要计算的光源数目不超过 50 个； 如果 camera 快速运动导致大量萤火虫离开等效范围重新成为“个体”，则会临时增大计算负荷。
  目前新算法还有一个缺点就是在萤火虫由远及近的过程中，在由个体光源转变成等效光源的瞬间会有一个跃变，当光源数目较少的时候会比较明显。
- 使用面剔除优化
  在添加动态多光源之后发现光源一多渲染效率低下，重新定义了模型顶点数据为逆时针顺序，并且开启面剔除，显著提高了渲染效率。
- 天空盒
  自建 Unity 项目，利用网上的素材搭建场景并截图用于天空盒的贴图。
  在绘制的时候需要关闭深度写入，

```
glDepthFunc(GL_LEQUAL);
```

  并且在绘制完毕后开启深度写入

```
glDepthFunc(GL_LESS);
```

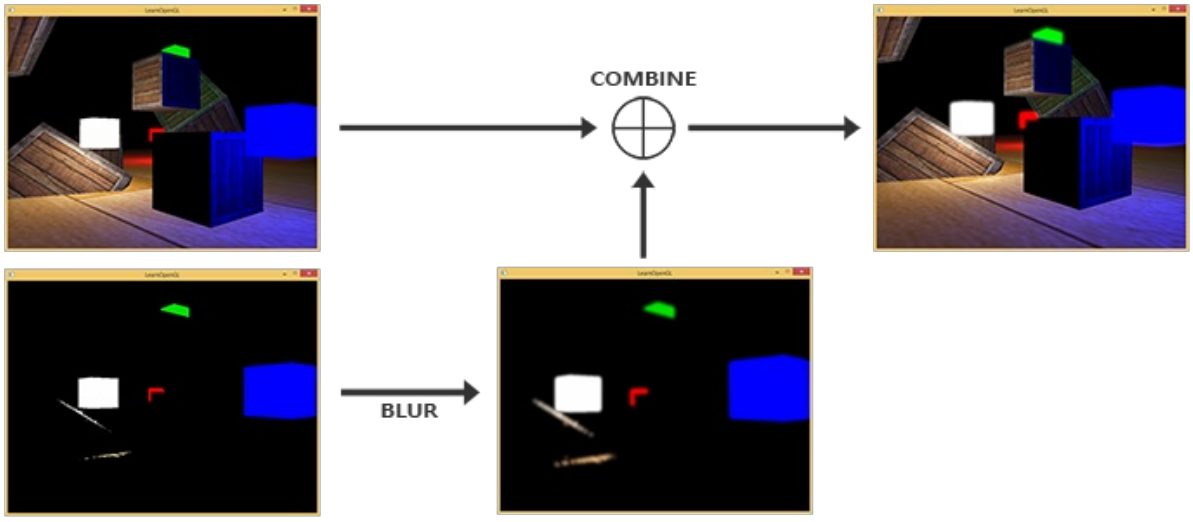
  才能让天空盒作为其他物体的背景。
- 3D 立体音效
  采用 irrklang 音效库，为每个粒子个体配备一个声音引擎，循环播放音效。在每一帧的更新中实时把摄像机位置（包括摄像机坐标和前向量）和当前粒子位置传给引擎，来达到立体音效。

技术细节

- 高光范围的调整
  使用了高动态范围（HDR）、曝光色调映射（Exposure Tone Mapping）、Gamma矫正的技术，使得等效光源强度过高处不会显得过于刺眼，而光线过暗处又不会失去过多细节。
- 萤火虫荧光效果
  使用了帧缓冲（FBO）和泛光（Bloom）技术实现萤火虫模型本身的**荧光**效果。原理介绍如下：
  1. 要将整个场景（包括环境模型和粒子系统）渲染到自定义的帧缓存对象中，目的是让我们可以实

时地访问每一帧地的像素值。

2. 在渲染发光体的时候通过界定阈值来提取发光体，将正常图像和提取出的发光部位分别装入两个颜色缓冲。
3. 再用高斯模糊处理提取出的发光体，让其变得模糊。
4. 接着将正常图像和模糊过的发光体融合在一起，得到最终场景贴图。
5. 最后将场景贴图渲染在整个屏幕上。



- 模型读取
  使用了 Assimp 模型加载库，但是从 Assimp 读取之后到绘制之前的衔接（包括 Mesh 类和 Model 类）全部由自己实现。
  Model 类负责**递归地**将 Assimp 加载出的每个面片进行分析，将其中的点坐标、法线、材质等数据抽离出来交给 Mesh 类进行处理。其中材质分为两种：颜色数据和纹理数据。如果是颜色数据，要将 Ambient、Diffuse、Specular 三种颜色数据分别提取并进行合成再传给 Mesh 类；如果是纹理数据，要做纹理映射后将纹理坐标传给 Mesh 类。
  为了实现不同风格的模型渲染分别实现了 environment_color 和 environment_texture 两套 shader。前者用于只有颜色数据的模型（如 low poly 风格模型）的加载，后者用于带纹理模型的加载。
- 次级粒子系统
  检测到 T 键被按下之后，每只萤火虫的个体开始受到一个向下的恒定加速度，同时颜色逐渐变暗直到全黑接着消失。整个消亡过程中的每一帧，每个萤火虫个体将会作为源头产生更小的粒子。次级粒子拥有更短的寿命，颜色取其产生时父亲粒子的颜色，以随机初速度从父亲粒子的位置发射，发射后受到向下的加速度。

Reference

- Learn OpenGL
- OpenGL Flocking System
- Inherent Noise-Aware Insect Swarm Simulation
- OpenGL Dynamic 2D Soft Shadows (Multiple Lights)