

Computer Graphics Project

李欣 1252368 陈瑞年 1252371 陈莘莘 12523

1 简介

这是一个 3D 的拼图游戏，玩家可以喜欢的模型传入，游戏会自动分割模型并打散，当玩家操作两个碎片距离足够接近时，碎片将会自动合在一起。

2 编译方法

这份代码需要用到以下几个库，我已经解决了在 Linux 上运行的 bug，建议在 ubuntu 上安装，Windows 上比较麻烦。

- boost
- CGAL
- SPL
- GLUT
- Assets Import
- DevIL
- gmp
- MPFR

2.1 Ubuntu 下编译方法

如果在 Ubuntu 下进行编译，只需要先执行

```
sudo apt-get install g++ libboost-dev libboost-system-dev libboost-  
thread-dev libdevil-dev libgmp-dev libgmpxx4ldbl libmpfr-dev  
freeglut3-dev
```

接下来解压 *Z6.zip*, 进入其中 *src* 文件夹, 执行 **make**, 就会在 *Z6/bin* 中生成可执行文件 *main*, 进入 *bin* 文件夹, 使用这个命令 **LD_LIBRARY_PATH=./lib ./main** 可以执行程序.

3 使用方法

编译后会得到可执行文件 *main*，模型和难度通过命令行参数传入。

命令行参数的第一个为模型文件所在路径，第二个参数可选，为难度 (1-3, 1 为最简单)，如果不提供难度，程序将只能进行模型查看。

程序自带了几个模型，放在 *Z6/models* 中。例如在 *Z6/bin* 中执行

```
LD_LIBRARY_PATH=../lib ./main ../models/cow-nonnormals.obj
```

2

就可以使用难度 2 将牛模型进行拆分。

4 操作方法

4.1 视角移动

使用 W、S 可以让视角在垂直于屏幕的方向上进行前后移动，A、D 可以左右移动，按住右键可以上下移动。鼠标移动可以使视角旋转。

4.2 物体移动

抓取 当屏幕中心聚焦到一个碎片上面时 (会显示此碎片的 bounding box)，此时按 E 可以抓住物体

平移 在物体被抓取的时候，使用 WASD 移动视角的同时，物体会跟着运动。抓住物体的时候无法使用右键上下移动。

旋转 在物体被抓取的时候，使用左键可以旋转物体本身。如果需要旋转移动方向，需要先退出物体抓取，使用鼠标旋转视角后再进行物体抓取。

5 游戏流程

游戏程序启动后，会有五秒的观察时间，此时可以任意改变视角检查物体形状。接下来物体会变成若干个碎片散开，动画持续两秒。动画结束，游戏正式开始。

游戏开始后可以使用所有操作，如果两个碎片足够接近，他们会吸附在一起变成一个碎片，如果所有碎片都已经被合并，那么游戏结束，左上角显示你赢了以及所花的时间。

6 代码结构

这部分由李欣完成。

这份代码是三平台通用的 (Windows、Linux、Mac OS X)，使用宏来判断平台并包含相应的头文件，缺点是在 Windows 上编译起来比较困难。

主要设计了两个类，**Camera** 和 **Model**，**Camera** 类是进行视角控制的，**Model** 是用来表示一个碎片的类。

Model 中，使用了 **mesh** 来表示模型（模型文件的导入使用了 **assimp** 库），但用户操作碎片时，碎片本身的坐标不会发生变化，否则逻辑非常混乱（例如不同模型的坐标范围不同，模型中不同部分有可能使用了相对变换）。我选择在 **mesh** 基础上添加偏移量，旋转轴，旋转角和 **id** 这几个控制量，在 **Model::render** 函数中操作 **MODELVIEW** 矩阵来实现碎片的变换，这样也方便了碎片的吸附判定。

为了解决不同模型的坐标范围和变换方式不同，在读入模型后，会计算模型的 **Bounding Box** 的大小和中心，然后将整个模型的缩放比例调节到使得 **Bounding Box** 最长的一边缩放成 1，并计算出将中心移到原点所需的偏移量，这两个量（缩放比例和偏移量）会被所有碎片使用。

程序逻辑由 **main.cpp** 实现，主要是通过全局的时间和状态来控制当前的行为。

7 视角控制和碎片操作

这部分由陈瑞年完成。

视角控制 在程序中，视角被锁定在屏幕的中心，通过键盘上的 **wsad** 来实现相机位置的变动，每次鼠标移动时，通过移动的 **x, y** 值来转换成相应的角度，并以此来变换相机的方向。

碎片操作 碎片的操作主要采用了 **arcball** 的方式，通过鼠标在屏幕上的移动，转换成其对应 **arcball** 上的运动，然后通过球面上的两点与球心所形成的向量，便可以求得转轴和旋转的角度。

8 模型分割

这部分由李欣完成,用的是这篇论文<http://onlinelibrary.wiley.com/doi/10.1111/j.1467-8659.2007.01103.x/pdf>提到的 mesh segmentation 方法,主要思想就是使用形状直径函数 (Shape Diameter Function) 拟合混合高斯模型,然后加上不平滑度惩罚进行无监督聚类。但实现上,是使用了 CGAL (Computer Geometry Algorithms Library) 库中的函数。

9 模型选取和吸附判定

这部分由陈莘莘完成。

9.1 模型选取

模型选取使用了 OpenGL 提供的 Stencil Buffer,将模型的 id 写在 Stencil Buffer 中,需要选取时使用 `glReadPixel` 读取指定位置的 buffer 值即可知道选取的物体。id 的写入在 `Model` 类的绘制函数中。

9.2 吸附判定

由于碎片本身的坐标不会改变,所以吸附判定比较容易。

首先需要得到碎片之间的相邻关系,只有相邻的碎片才能吸附。由于使用了 CGAL 库进行 Mesh Segmentation,只能获得分割后的结果,不知道不同碎片间的相邻关系,所以必须额外计算。

大部分情况下,模型的点数和面数会特别多,相邻判定需要比较高效。这里采用了 $O(n \log n)$ 的算法, n 为点数。

对于两个碎片,我们认为如果它们的某些面共享了相同的顶点,那这两个碎片就是相邻的。因为模型是一个三角化过的整体,Mesh Segmentation 并不会将面进行分割,只会选择哪个面是属于哪个碎片的。所以只需要将碎片的顶点放在一起排序,那么相同的顶点一定会出现在排序后的相邻位置,如果相同的顶点来自两个不同的碎片,就说明这两个碎片是相邻的。

得到了碎片间的相邻关系之后,还需要判断两个相邻碎片是否在用户操作后被放在了合适的位置。我们采用的判定条件如下:

- 两个碎片的旋转轴张角不能超过一个阈值，张角判断可以转换为点乘判断，点乘的优势是计算方便，同时没有特殊情况，如果使用叉乘，就会在张角为 180 度附近时和 0 度重叠。
- 接下来计算每个碎片的 Bounding Box 的中心，两个碎片的初始中心相减可以得到初始位置关系。碎片中心加上用户操作偏移量就是当前中心，两个碎片的当前中心相减可以得到当前的位置关系。如果初始位置关系和当前位置关系的点乘特别接近初始位置关系的长度的平方，那么就可以认为两个相邻碎片可以吸附。

如果两个碎片可以被吸附在一起，那就把用户正在操作的碎片附在另一个碎片上，这样视觉效果比较好。