Mitsuba 系列 3-变换类与三角形的移植

Dezeming Family

2022 年 6 月 23 日

DezemingFamily 系列文章和电子书**全部都有免费公开的电子版**，可以很方便地进行修改和重新发布。如果您获得了 DezemingFamily 的系列电子书，可以从我们的网站 [https://dezeming.top/] 找到最新的版本。对文章的内容建议和出现的错误也欢迎在网站留言。

目录

一　基本介绍

本文我们移植变换类、Ray 类与三角形类（以及一些我们会用到的类），我们的目标是能够在屏幕上显示两个三角形。源码见 [2 - TransformAndTriangle]。

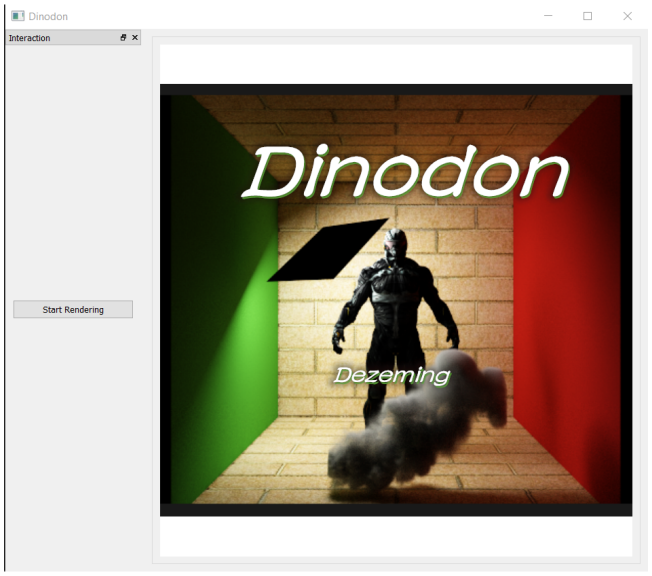
由于这些类都是属于基本功能，与渲染的流程都是相互解耦的，所以我们暂时不花费太多精力去搞明白每一个函数的作用和意义。在做光线求交与着色计算的时候，用到的函数我们会详细讲解。另外，一些需要用的到功能函数，比如 Util.h 里的函数，我们目前用到了哪个就移植哪个，而不是全都移植上来。

1 1　图形交互界面

为了能够初步显示一些图像，我们在 MainGUI 下添加了两个类：DisplayWidget 和 IMAGraphicsView，它们都是 Qt 控件。同时我们还添加了 InteractionDockWidget，这是一个 Dock 控件，用于做一些交互，目前唯一的交互就是启动渲染线程。

渲染线程类定义为 RenderThread，继承自 QThread 类，在该类 run() 函数里就是进行渲染的程序。

默认程序一启动显示背景：



我们也实现了一个简易的 FrameBuffer 类，放在了 Core 目录下。该类是一个 QObject 的派生类，用于定义相应的信号和槽。目前交互控件里没有使用复杂的功能，因为这样会使我们抓不住重点。

二　基本类别的移植

我们本节先移植 aabb.h、aabb.cpp、ray.h、bsphere.h。aabb.h 里定义了 AABB 包围盒类；bsphere.h 是包围球类，也可以看做是一种球形包围盒。我们目前只是移植这几个类的功能，在下一节我们结合三角形详细解释。

接下来移植 util.h 和 util.cpp 这两个文件，我们只移植其中的 solveQuadratic() 求解二次函数，如果二次方程有实根则返回 true，否则就返回 false。

然后移植 warp.h 和 warp.cpp 文件的 squareToUniformTriangle() 函数，该函数将 $[0,1)^2$ 二维均匀分布的变量转换到在三角形上均匀分布的变量，用来对三角形上的点采样。

以及 Math.h 的几个函数：castflt_down() 表示如果不能精确表示，则转换为单精度并向下舍入。castflt_up() 表示如果不能精确表示，则转换为单精度并向上舍入，该函数是借助 Union 来实现的。

注意，虽然我们目前的文件数量已经不少了，但很多文件我们只移植了其中几个函数，所以理解起来并不会很困难。下一节我们移植和讲解三角形类 Triangle，并讲解包围盒类的函数。

三　三角形类

我们把三角形类放在 Shape 目录下，而不是像 Mitsuba 那样放在 Core 目录下。

3 1　三角形类以及功能

Triangle 类里有下面几个成员函数：

```
getAABB
getClippedAABB
getBSphere
sample
surfaceArea
rayIntersect
```

其中，getClippedAABB() 函数用到了论文 [1] 中的技术，该算法是此函数是高效创建“完美分割”kdTree 的重要组成部分，此算法会应用到 Sutherland-Hodgman 算法，即 Triangle.cpp 中的 sutherlandHodgman() 函数。我们不详细讲解该函数的作用和意义，有兴趣的同学可以去阅读论文。

3 2　包围盒类以及功能

AABB 包围盒有一个 min 和一个 max，表示包围盒顶点的最大和最小值。

contains() 函数用来判断一个点或一个包围盒是否在包围盒内部。expandBy() 函数用来根据参数扩展包围盒大小。clip() 根据另外一个包围盒来求公共部分。getVolume() 求包围盒体积。getCenter() 求包围盒中心位置。

getCorner() 函数获得包围盒中的一个顶点，我们以 TPoint3<Float> 为例，dim=3，所以输入参数 index 取值为 0-7。

```
index==0: 返回Point(min[0],min[1],min[2])
index==1: 返回Point(max[0],min[1],min[2])
index==2: 返回Point(min[0],max[1],min[2])
index==3: 返回Point(max[0],max[1],min[2])
index==4: 返回Point(min[0],min[1],max[2])
index==5: 返回Point(max[0],min[1],max[2])
index==6: 返回Point(min[0],max[1],max[2])
index==7: 返回Point(max[0],max[1],max[2])
```

getChild() 函数找到中心点与其中一个顶点的子包围盒（可以想想八叉树结构）。overlaps() 函数判断包围盒与另一个包围盒是否有重叠。

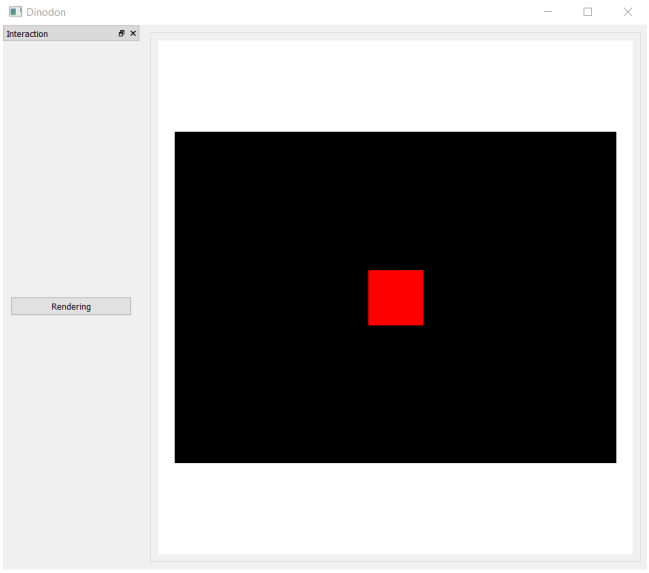
squaredDistanceTo() 函数判断包围盒顶点到一个 Point 点或者一个 AABB 包围盒的最小平方距离。distanceTo() 函数对最小平方距离开根号。

getLargestAxis() 和 getShortestAxis() 分别计算最长的和最短的轴的距离。

四　本文的最终测试

虽然我们已经移植了不少工具，但我们还不知道这些工具是否能正确工作，我们需要写一个小例子测试一下。

我们定义三维空间中的两个三角形，然后定义屏幕中的 Ray 去与这些三角形求交，如果相交，就显示红色。写成后的代码见 RenderThread::run() 函数，显示结果如下：



本文的文字部分较少，但我写了整整五个晚上，因为调试图形界面和移植很容易出错，尤其是多线程交互中的的内存访问报错。本文已经具备了渲染器的雏形，下一本小书我们开始实现模型的读取功能。

参考文献

[1] Wald I, Havran V. On building fast kd-trees for ray tracing, and on doing that in O (N log N)[C]//2006 IEEE Symposium on Interactive Ray Tracing. IEEE, 2006: 61-69.