LearnOptix-v7系列1-使用入门介绍

Dezeming Family

2023年8月5日

DezemingFamily系列书和小册子因为是电子书，所以可以很方便地进行修改和重新发布。如果您获得了DezemingFamily的系列书，可以从我们的网站[https://dezeming.top/]找到最新版。对书的内容建议和出现的错误欢迎在网站留言。

全局光照(GI)有很多种解决方案，比如VXGI、Lumen、DDGI、SSGI、IBL、PRT、SurfelsGI等，其中，越来越火的Nvidia的RTX技术也是一些软硬件结合的实时光追解决方案。

目录

一　多种Nvidia光追模块介绍

在《LearnOptix系列1-Optix使用入门介绍》中已经介绍了很多内容，所以这里只是简单介绍一些关键内容。

OptiX API已在多种应用中使用多年。它经过了多次实战测试，是将GPU光线追踪推向世界的关键工具。自NVIDIA的RT-Core硬件发布以来，人们越来越希望将GPU加速应用于更大、更复杂的光线跟踪工作负载，例如故事片的最终帧(final-frame)渲染和超大数据集的交互式预览。为此，Nvidia重新设计了OptiX API（也就是Optix 7），以在实现此类应用程序时提供更大的灵活性。这个新的API级别较低，类似于DXR或Vulkan光线追踪的抽象级别，但保留了OptiX中一直存在的许多关键概念。

在SIGGRAPH 2019上，NVIDIA发布了两个新版本的OptiX API：OptiX 7.0，其中包含新的低级API；以及OptiX 6.5，这是对经典API的更新。通过[3]中的表可以看到，对底层控制和硬件性能支持最好的就是Optix 7。

OptiX的核心是一个简单但强大的光追抽象模型，该光线跟踪器使用用户提供的程序来控制光线的起始、光线与曲面的相交、材质的着色以及新光线的生成。光线携带用户指定的有效数据，用于描述每条光线的属性，如颜色、递归深度、重要性或其他属性。开发人员以基于CUDA C的功能的形式向OptiX提供这些功能。由于光线跟踪是一种固有的递归算法，OptiX允许用户程序递归生成新光线，内部执行机制管理递归堆栈的所有细节。OptiX还提供了灵活的动态函数调度和复杂的变量继承机制，从而可以非常通用和紧凑地编写光线跟踪系统。

Optix的另一个很好的功能是，光线跟踪的执行通常是“次线性(sub-linear)”的，因为如果场景中对象数量提高两倍，运行时间不会增加两倍。这是通过将对象组织成一个加速结构来实现的，该加速结构可以在整个基元组不是与任何给定光线相交的候选者时快速拒绝它。对于场景的静态部分，此结构可以在应用程序的整个生命周期中重复使用。对于场景的动态部分，OptiX支持在需要时重建加速结构。该结构只查询其包含的任何几何对象的边界框，因此可以任意添加新类型的基元，并且只要新的基元可以提供边界框即可使用该加速结构。

我们下一节介绍Optix 7的编译和初步使用。

二　Optix 7的编译

在[6]中下载Optix 7.5，注意虽然标注的是win10，但是win11系统安装和使用也是没有什么问题的。

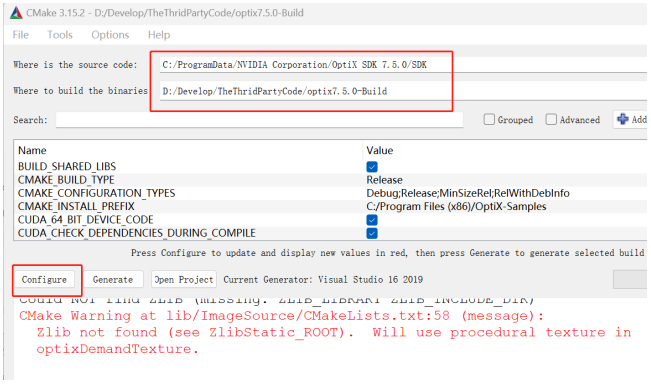
然后安装，一直点Next即可，不需要手动修改什么内容。默认的安装路径是：

```
C:\ProgramData\NVIDIA Corporation\OptiX SDK 7.5.0
```

注意ProgramData是一个隐藏路径，需要显示隐藏路径才能看到。在下面的路径里有CMakeLists：

```
C:\ProgramData\NVIDIA Corporation\OptiX SDK 7.5.0\SDK
```

我们安装Visual Studio 2019以及CUDA 11.7，以及下载CMake工具。之后新建一个Build目录，作为构建的Optix工程目录，然后配置，不需要其他多余的设置：



之后打开VS工程，编译即可，可以随便运行几个例子感受一下。注意Debug模式比Release模式要慢得多。

三　如何利用Optix开发

如果要自己配置一个工程，那么所需设置基本和Optix 6是一样的，在Optix的系列文章里我已经讲解过，所以目前不再赘述。

3 1　本系列目标

因为网上对于Optix 7已经有了一个不错的系列教程[7]，所以我认为我没有必要再自己写一个了，但是我会根据该教程写一些注意事项，包括环境搭建/代码构建/代码理解的一些重要问题。

这个教程[7]是从最初的原理出发，逐步教授OptiX 7，并具有最小的外部依赖性。这使得这个特定的代码库非常适合学习基础知识；然而，它不一定是最终用来构建自己的高级项目的最大资源。如果真的想跟进并构建你的owl OptiX项目，建议也看看另一个OptiX 7资源，即OWL项目[8]，特别旨在通过将在本课程中学习的所有或大部分技术包装在OptiX 7之上更易于使用的“便利层”API中，使编写OptiX程序变得更容易。事实上，OWL现在也包含了一个例子，它在OWL框架内准确地再现了本课程的最终样本。

该教程旨在介绍全面的光线追踪Optix引入的内容，即OptiX Context, Module, Programs, Pipeline, Shader Binding Table (SBT), Accel Struct (AS), Build Inputs, Texture Samplers。为了做到这一点，这个教程有意不提供一个包含最终代码的示例，而是分成12个较小的示例，每个示例都修改和扩展了前一个示例，希望以一种相对容易发现差异的方式（即，发现从“A”到“B”到底需要添加什么）。

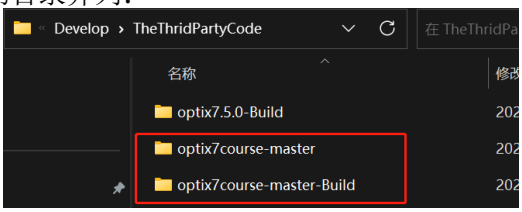
原教程可以在[7]页面上找到下载链接。

该教程的代码是故意以最小的依赖性编写的，只需要CMake（作为一个构建系统）、在Windows下使用Visual Studio 2017和2019进行测试，在Linux下使用GCC进行测试和OptiX 7 SDK（包括CUDA 10.1和NVIDIA驱动程序，这些驱动程序足够新，可以支持OptiX）。

3 2　编译方法

开发环境我用的是VS2019；Optix版本是7.5；CUDA版本是v11.7。

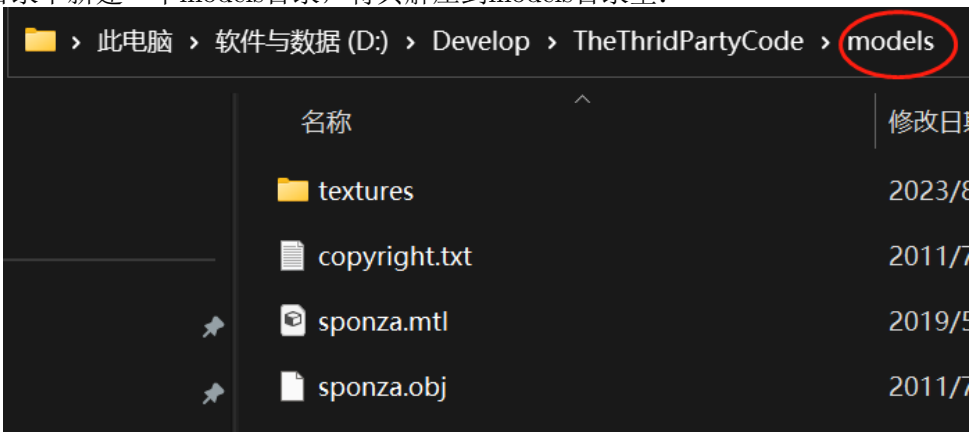
我把构建目录设置为和代码目录并列：



先使用CMake生成，第一次configure时会报错，此时需要修改一下下面的两个路径的位置：

OptiX_INCLUDE	C:/ProgramData/NVIDIA Corporation/OptiX SDK 7.5.0/include
OptiX_INSTALL_DIR	C:\ProgramData\NVIDIA Corporation\OptiX SDK 7.5.0\SDK

编译好以后，还需要下载一些数据集，从[9]中下载Crytek Sponza model ，然后在与构建目录和源码目录所在的目录下新建一个models目录，将其解压到models目录里：



目前工程已经都设置好，然后就可以运行任意一个工程代码了。

四　小结

下一篇文章开始我会根据官方教程来丰富和完善Optix 7的教程。

参考文献

[1] https://developer.nvidia.com/rtx/ray-tracing

[2] https://developer.nvidia.com/rtx/ray-tracing/optix

[3] https://developer.nvidia.com/blog/how-to-get-started-with-optix-7/

[4] https://raytracing-docs.nvidia.com/optix7/index.html

[5] https://raytracing-docs.nvidia.com/optix7/guide/index.html#preface#

[6] https://developer.nvidia.com/designworks/optix/downloads/legacy

[7] https://github.com/ingowald/optix7course

[8] https://owl-project.github.io/

[9] https://casual-effects.com/data/