ISSN: 2664-9640

Research and Implementation of 3D Model Visualization based on OptiX

Wei Huang, Haiyun Xiang

Network and Information Center, Southwest Petroleum University, Sichuan 610500, China.

Abstract

Image rendering plays an indispensable role in the field of 3D vision. To accelerate the development of computer graphics and improve the quality of image rendering, the basic situation of common image rendering engine in recent years is analysed, Nvidia's OptiX ray tracing engine is introduced, the process of building an integrated environment based on VS, CUDA and OptiX is described, a kind of 3D Model Visualization Algorithm based on OptiX is proposed, the rendering result of 3D Model is analyzed from the aspects of illumination, material and interaction. The result shows that the 3D Model Visualization Algorithm based on OptiX can restore the 3D scene clearly and truly, and achieve good results in color and sharpness.

Keywords

OptiX, CUDA, Render, 3D Model.

1. Introduction

In recent years, with the update and iteration of computer hardware and software, computer graphics has developed rapidly. The development of graphics marks that the computer has entered the age of 3D. 3D products are widely applied to education, game, entertainment and other industries with the rich color, the impressive image and the realistic operation experience.

Image rendering is a process of converting 3D light to the 2D image. At present, the development of rendering tools is more and more mature. Common image rendering APIs include DirectX, OpenGL, etc[1]. Common rendering engines include Ogre, Osg, OptiX, etc[2]. Most of early image rendering techniques only apply in CPU and are of long rendering time, poor imaging effect and other characteristics. OptiX is an emerging graphics rendering engine from Nvidia in recent years, its rendering process makes full use of the collaborative operation mechanism of CPU and GPU, and uses the ray tracing principle to reversely calculate the voxel information in three-dimensional space. On the GPU terminal, OptiX uses CUDA to realize multi-threaded cooperative rendering, which can greatly improve the rendering rate and enhance the rendering effect.

2. OpitX ray tracing engine

OptiX is a general purpose ray tracing engine designed for NVIDIA GPUs to achieve optimal ray tracing performance[3]. OptiX consists of the Host terminal and the Device terminal. The Host terminal is a host-based API, which defines the data structure based on ray tracing. The Device terminal is a programming system based on CUDA C, which can be used to generate new rays, calculate the ray's information and construct ray tracing pipelines[4]. Developers can program in C to create ray tracing applications based on GPU. OptiX is widely used in all kinds of fields, such as graphic fields, optical design fields and acoustic design fields. Table 1 shows that common elements in OptiX.

ISSN: 2664-9640

Table 1: OptiX element list

Element	Instruction
Context	An instance of a running OptiX engine
Program	A CUDA C function, compiled to NVIDIA's PTX virtual assembly language
Variable	The value used to pass C to OptiX
Buffer	A multidimensional array used to bind variables
TextureSampler	The interpolation mechanism used to bind buffers
Geometry	One or more primitives that a ray can be intersected with, such as triangles and rectangles, etc
Material	A set of programs executed when a ray intersects with the closest primitive or potentially closest primitive
GeometryInstance	Used to bind Geometry and Material
Group	A group of objects arranged in a hierarchy
GeometryGroup	A set of GeometryInstance objects
Transform	A hierarchical node
Selector	A programmable hierarchical node
Acceleration	An acceleration structure object

3. The construction process of OptiX integrated environment

The 3D visualization platform is built on the "Windows 10", which requires the integration of many compilation environments. The platform toolkit includes Visual Studio 2012, CUDA Toolkit 6.0, Nvidia OptiX 3.7.0 and Cmake 2.8.12.2.

1) Install the Visual Studio 2012.

2) Install the CUDA Toolkit 6.0.

(1) Install the CUDA package.

(2) Configure environment variables.

CUDA_SDK_PATH=C:\ProgramData\NVIDIA Corporation\CUDA Samples\v6.0

CUDA_LIB_PATH=%CUDA_PATH%\lib\x64

CUDA_BIN_PATH=%CUDA_PATH%\bin

CUDA_SDK_BIN_PATH=%CUDA_SDK_PATH%\bin\x64

CUDA_SDK_LIB_PATH=%CUDA_SDK_PATH%\common\lib\x64

(3) Configure the system variable "path".

;%CUDA_LIB_PATH%;%CUDA_BIN_PATH%;%CUDA_SDK_LIB_PATH%;%CUDA_SDK_BIN_PATH%;

(4) Create a CUDA project to test the CUDA compilation environment.

In "Project - Property", select "CUDA 6.0" for "Generating Customizations".

In "VC++ directory", add the following included directories:

C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v6.0\include

C:\ProgramData\NVIDIA Corporation\CUDA Samples\v6.0\common\inc

Add the following library directories:

C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v6.0\lib\x64

ISSN: 2664-9640

C:\ProgramData\NVIDIA Corporation\CUDA Samples\v6.0\commonlib\x64

In “Project - Property - Configuration Property – Linker - General - Additional Library Directory”, add the following directory:

$(CUDA_PATH_V6_0)\lib\$(Platform)

In “Project - Property - Configure Property - Linker - Input – Additional Dependencies”, add the following static libraries:

cublas.lib; cublas_device.lib; cuda.lib; cudadevrt.lib; cudart.lib; cudart_static.lib; cufft.lib; cufftw.lib; curand.lib; cusparse.lib; nppc.lib; nppi.lib; npps.lib; nvcuvenc.lib; nvcuvid.lib; OpenCL.lib;

At last, set the project type to "CUDA C/C++" and the VS platform to "X64".

3) Install the Nvidia OptiX 3.7.0.

4) Compile the OptiX toolkit with the Cmake.

(1) Run “cmake-gui.exe” as an administrator.

(2)Set the “source code” to compile OptiX, the path is “C:/ProgramData/NVIDIA Corporation/OptiX SDK 3.7.0/SDK”.

(3)Create a new folder named “build”, set the target path to compile OptiX. The path is “C:/ProgramData/NVIDIA Corporation/OptiX SDK 3.7.0/build”.

(4)In Cmake, check “Grouped” and “Advanced”, and click “Configure” and “Generate” to complete the compilation.

5) Build all samples in OptiX. Select the path to compile the OptiX solution. The path is “C:\ProgramData\NVIDIA Corporation\OptiX SDK 3.7.0\build”. After the compilation, the OptiX integration environment is configured.

4. A kind of 3D Model Visualization Algorithm based on OptiX

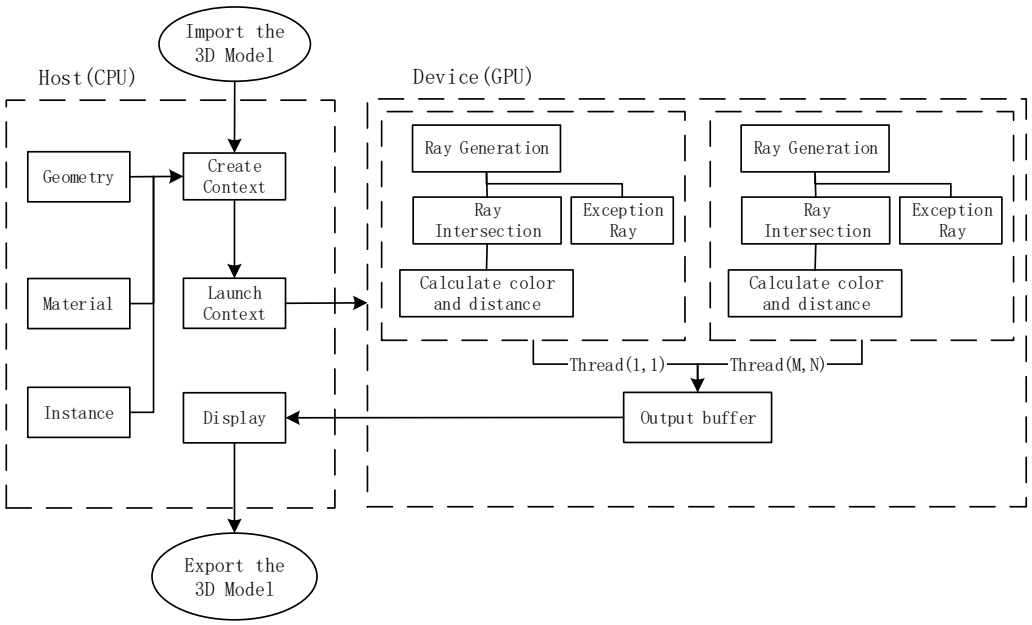


Figure 1: OptiX ray tracing process

The algorithm adopts the hybrid rendering mechanism of CPU and GPU, programs on the Host run on the CPU, and programs on the Device run on the GPU. On the Host terminal, the scene is arranged through OptiX. Its steps include importing the 3D Model, creating the “Context”, creating the “Geometry”, creating the “Material”, instantiating the scene, launching the “Context”, and exporting the 3D Model. On the Device terminal, OptiX uses CUDA to launch 3D scene rendering threads in parallel, uses the ray tracing algorithm to trace the voxel

ISSN: 2664-9640

information of the scene in the opposite direction, and creates the "pinhole_camera.cu, sphere.cu, normal_shader.cu, constantbg.cu". Figure 1 shows the ray tracing process.

4.1. Host terminal

1) Import the 3D Model. The algorithm uses the OBJ file as the model file. The "OpitXMesh" class is used to load the model, and the loading process is performed in the "Instantiate the scene". The steps are as follows:

OptixMesh loader(context, geometrygroup, material);

loader.loadBegin_Geometry(s);

loader.loadFinish_Materials();

2) Create the "Context". It contains the settings of camera's focal point and focal length in the scene, contains the settings of lighting parameters, contains the setting and invoking of the "pinhole_camera.cu" and the "constantbg.cu".

3) Create the "Geometry". It contains the setting and invoking of the "sphere.cu".

4) Create the "Material". It contains the setting and invoking of the "normal_shader.cu", contains the value setting of Kd, Ka, Ks, Phong lighting.

5) Instantiate the scene. It provides instantiation operations for the "Context", the "Geometry", and the "Material". It also includes importing the OBJ file.

6) Launch the "Context". This step involves compiling and running. OptiX compiles this context with the "rtContextCompile()" and launches it with the "rtContextLaunch2D()".

7) Export the 3D Model. This module mainly includes two parts: Pixel Data Conversion and Bitmap Saving. Pixel Data Conversion is performed by the function "SwizzleData()", which mainly involves converting pixels from RTbuffer to RGB format. Bitmap Saving is performed by the function "SaveBmp()", which saves the RGB array as the BMP.

4.2. Device terminal

1) Create the"pinhole_camera.cu". It includes the steps of the ray tracing algorithm, includes ray's generation, intersection and calculation. Rays that do not intersect with objects are considered "Exception Ray".

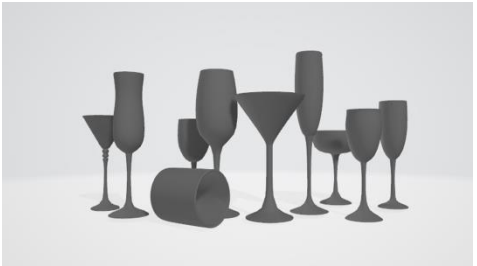
2) Create the"sphere.cu". The geometry of the algorithm adopts the geometry of "sphere" class.

3) Create the"normal_shader.cu". It includes calculations for Diffuse, Specular, Ambient, Kd, Ka, Ks, Phong Lighting.

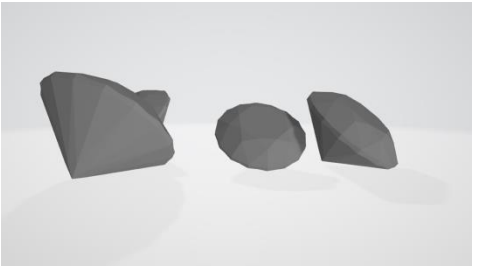
4) Create the"constantbg.cu". It shows the calculation of scene's background color.

5. Analysis of 3D Model rendering results

In the experiment, the OBJ file is selected as the 3D original file imported into OptiX engine for rendering. The render result is saved as BMP, the image size is 3840*2160px. Figure 2 shows a 3D Instance of OptiX engine rendering.

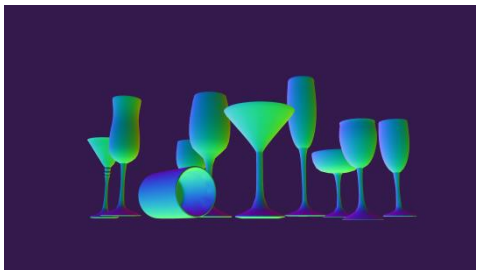


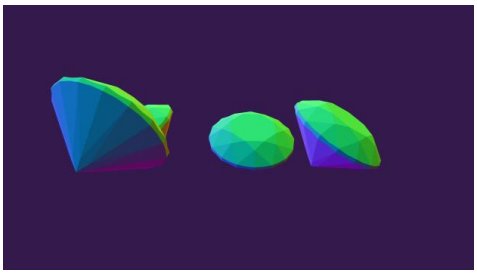
(a) Glass

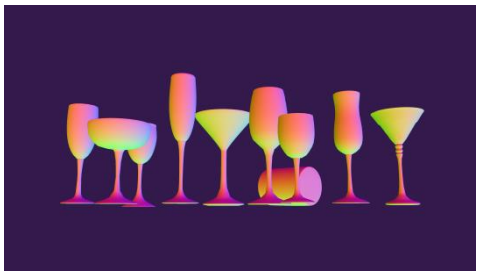


(b) Diamond

ISSN: 2664-9640


(c) Glass-Material 1


(d) Diamond-Material 1


(e) Glass-Material 2

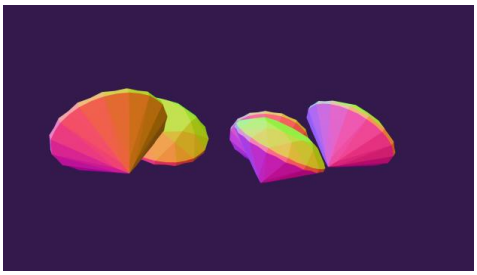

(f) Diamond-Material 2

Figure 2: 3D Instance of OptiX engine rendering

5.1. Light

Three different lighting parameters are set. Figure 2 shows the lighting effect produced by light1's rendering. OptiX lighting provides a good basis for shading, reflection and refraction.

Light1：

```
BasicLight lights[] = {
{ make_float3( -60.0f, 30.0f, -120.0f ), make_float3( 0.2f, 0.2f, 0.25f )*1.0f, 0 },
{ make_float3( -60.0f, 0.0f, 120.0f ), make_float3( 0.1f, 0.1f, 0.10f )*1.0f, 0 },
{ make_float3( 60.0f, 60.0f, 60.0f ), make_float3( 0.7f, 0.7f, 0.65f )*1.0f, 1 }
};
```

Light2：

```
BasicLight lights[] = {
{ make_float3( -5.0f, 60.0f, -16.0f ), make_float3( 1.0f, 1.0f, 1.0f ), 1 }
};
```

Light3：

```
BasicLight lights[] = {
{ make_float3(-60.0f, 60.0f, 60.0f), make_float3(1.0f, 1.0f, 1.0f), 1 }
};
```

5.2. Material

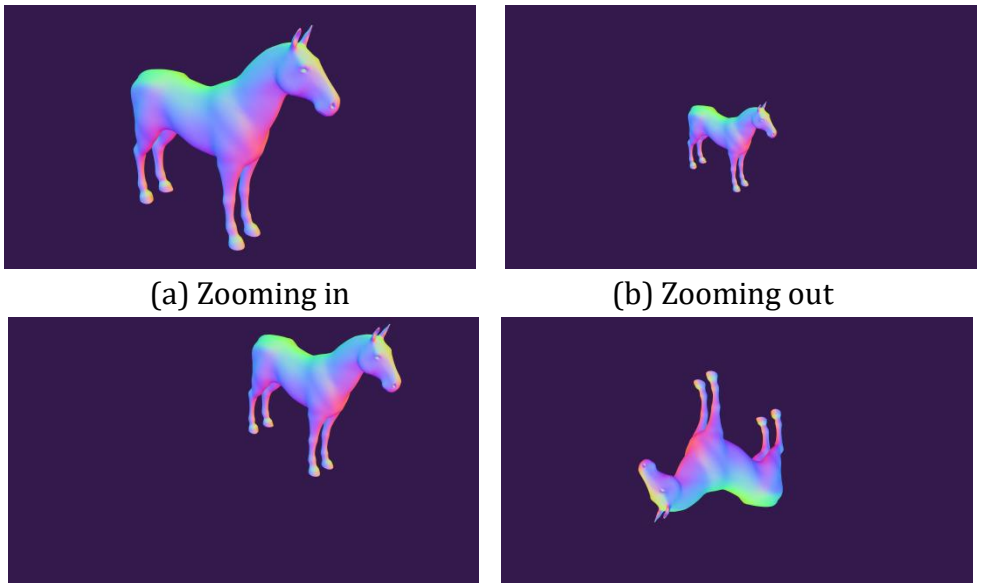
In “normal_shader.cu”, two different material settings for the model are implemented in the functions closest_hit_radiance1() and closest_ hit _radiance2(), respectively. The material parameters are instantiated. The specific parameters are set to Kd=1.0f, Ka=0.6f, Ks=0.0f, phong_exp=0.0f.

Figure 2 (a) and (b) show the original OBJ files of Glass and Diamond in the virtual scene, respectively. Figure 2 (c) and (d) show the rendering of Material 1, and perform green and purple. In another view, Figure 2 (e) and (f) show the rendering of Material 2, and perform red and yellow. By the OptiX, the 3D Model presents a high resolution visual effect, which enhances the visual experience of virtual reality.

ISSN: 2664-9640

5.3. Interaction

In OptiX, the observation point of observer is controlled by cam_eye[3]={x,y,z}. The variable "x", variable "y" and variable "z" represent the three directions of the observation point from x, y and z axis, respectively. The mouse event function is bound to variable "z" to adjust the distance of viewpoint, is bound to variable "x", variable "y" and variable "z" to control the rotation of the model. The focus of camera is controlled by lookat[3]={a,b,c}. Variable "a" controls the left and right adjustment of the focus, and variable "b" controls the up and down. Figure 3 shows the interactive effects of OptiX.



(a) Zooming in (b) Zooming out

(c) Translating (d) Rotating

Figure 3: OptiX interactive operations

6. Conclusion

The OptiX ray tracing engine runs entirely on the GPU. It breaks the traditional image rendering pipeline based on CPU and improves rendering quality and efficiency. OptiX integrated environment construction provides a clear idea of building 3D Model rendering environment. The 3D Model Visualization Algorithm based on OptiX provides a new way of 3D Model rendering. The practice shows that using OptiX engine to render 3D Model has good rendering experience in light, material and interaction, and also obtains a strong visual sense.

References

[1] Xiangming Zhou and Zhou Xiangming. Application of OpenGL Function in Computer Graphics[J]. Journal of Physics: Conference Series, 2020, 1648(3) : 032096-.

[2] Balvert Marleen et al. OGRE: Overlap Graph-based metagenomic Read clustEring[J]. Bioinformatics, 2021, 37(7) : 905-912.

[3] Xu Gaogui et al. Acceleration of shooting and bouncing ray method based on OptiX and normal vectors correction.[J]. PloS one, 2021, 16(6) : e0253743-e0253743.

[4] Simon Blyth. Opticks : GPU Optical Photon Simulation for Particle Physics using NVIDIA® OptiXTM[J]. EPJ Web of Conferences, 2019, 214.