


캡스톤 디자인 I

종합설계 프로젝트

프로젝트 명	<i>Lit</i>
팀 명	<i>Lit</i>
문서 제목	결과보고서

Version	1.0
Date	2021-JUN-09

팀원	곽 상열 (조장)
	양 교원

 국민대학교 소프트웨어학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Lit	
	팀 명	Lit	
	Confidential Restricted	Version 1.0	2021-JUN-09


CONFIDENTIALITY/SECURITY WARNING

이 문서에 포함되어 있는 정보는 국민대학교 소프트웨어융합대학 소프트웨어학부 개설 교과목 캡스톤 디자인 I 수강 학생 중 프로젝트 "Lit"를 수행하는 팀 "Lit"의 팀원들의 자산입니다. 국민대학교 소프트웨어학부 및 팀 "Lit"의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다.

문서 정보 / 수정 내역

Filename	Team6 Lit-수행결과보고서.docx
원안작성자	곽상열, 양교원
수정작업자	곽상열, 양교원

수정날짜	대표수정자	Revision	추가/수정 항목	내 용
2021-05-24		1.0	최초 작성	

 국민대학교 소프트웨어학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Lit	
	팀 명	Lit	
	Confidential Restricted	Version 1.0	2021-JUN-09

목 차

1	개요	4
1.1	프로젝트 개요	4
1.2	추진 배경 및 필요성	4
2	개발 내용 및 결과물	5
2.1	목표	5
2.2	연구/개발 내용 및 결과물	6
2.2.1	연구/개발 내용	6
2.2.2	시스템 기능 요구사항	6
2.2.3	시스템 구조 및 설계도	7
2.2.4	활용/개발된 기술	7
2.2.5	현실적 제한 요소 및 그 해결 방안	8
2.2.6	결과물 목록	8
	Final Result	10
2.3	기대효과 및 활용방안	10
3	자기평가	11
4	참고 문헌	13
5	부록	14
5.1	사용자 매뉴얼	14
5.2	운영자 매뉴얼	15
5.3	테스트 케이스	16

 국민대학교 소프트웨어학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Lit	
	팀 명	Lit	
	Confidential Restricted	Version 1.0	2021-JUN-09

1 개요

1.1 프로젝트 개요

자연의 아름다움을 리얼타임으로 컴퓨터에 구현하는 것은 힘든 일이다.
특히 Global Illumination효과의 경우 높은 퀄리티의 렌더링 결과를 내기 위해 중요한 요소이다. 따라서 리얼타임으로 실제와 같은 이미지를 생성하는 Renderer를 만들고자 한다.

1.2 추진 배경 및 필요성

두 학생 모두 Real-time Physically Based Rendering에 관심이 있었다. 따라서 그것과 관련된 프로젝트를 진행하고자 하였다. 그리고 두 학생 모두 대학원에 진학을 생각하고 있었으므로 대학원 입학 전 관심있는 분야에 대한 공부 겸 프로젝트를 진행하려고 했다. 따라서 둘의 관심분야에 맞는 렌더링 분야 중 Global Illumination효과를 내는 Renderer를 만들어보려고 한다.

최근 NVIDIA의 RTX플랫폼을 이용하여 Global Illumination을 구현하고 있는데 해당 하드웨어 유닛들을 갖추지 않은 사용자들의 경우 해당 기술을 사용할 때 퍼포먼스가 떨어지는 현상이 발생한다. 따라서 우리는 해당 하드웨어 유닛들이 없는 사용자에게도 비슷한 결과물을 내는 Renderer가 필요하다고 생각한다.

 국민대학교 소프트웨어학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Lit	
	팀 명	Lit	
	Confidential Restricted	Version 1.0	2021-JUN-09


2 개발 내용 및 결과물

2.1 목표

컴퓨터 그래픽스에는 여러 가지 분야(렌더링, 애니메이션, 지오메트리 등)이 있지만 그 중 렌더링 분야는 사실적인 이미지(Photorealistic Image)나 비-사실적인 이미지(Non-Photorealistic Image)를 컴퓨터를 통하여 만들어 내는 것을 목표로 하고 있습니다. 여기서 사실적인 이미지란 현실 세계에서 빛이 어떻게 작용하는지 근사화(approximation)하고 계산을 통하여 유한한 시간 내에 실제와 가까운 이미지를 만들어내는 것이 중요합니다. 이를 위해 오프라인 렌더링(Offline Rendering) 알고리즘, 특히 경로 추적(Path tracing)과 라디오시티(Radiosity) 알고리즘을 이용하고, 충분한 계산 시간이 주어진다면 실제와 가까운 이미지를 생성할 수 있게 됩니다. 하지만 실시간 렌더링(Real-time Rendering)의 영역으로 넘어오게 된다면, 이러한 오프라인 렌더링 알고리즘들은 사용하기에 너무 계산 시간이 많이 들거나, 짧은 시간에 계산해낼 수 있더라도 많은 아티팩트(노이즈와 같은)들이 나타나게 됩니다. 그래서 보통 실시간 렌더링에서는 사전 계산(Precomputation)나 빛의 동작 또는 오프라인 렌더링 알고리즘을 근사화 하여 실시간 퍼포먼스(최소, 초당 30 프레임 이상)를 달성합니다.

보통 실시간 렌더링에서 사전 계산을 통한 효과를 제외한다면 광원(Light source)과 그 광원과 상호작용하는 물체만 고려하는 지역 조명(Local Illumination) 모델을 많이 사용하게 됩니다. 하지만 이 모델로는 실제 세계에서 빛이 동작하는 방식을 충분히 설명하지 못하여, 현실과는 많이 다른 형태의 이미지를 만들어 내게 됩니다(예를 들어, 지역 조명 모델에 그림자를 고려한 알고리즘을 사용하였을 때, 빛의 반대 방향은 완벽하게 어두운 상태로 나타납니다). 여기서 더 나아가, 광원과 물체의 직접적인 상호작용 뿐 아니라, 광원으로부터 발생한 빛이 물체에 부딪히고, 그 물체로부터 반사된 빛이 다시 또 다른 물체와 상호작용 하는 간접광(Indirect light)을 고려하는 모델을 전역 조명(Global Illumination) 모델이라고 합니다. 하지만 간접광으로 인해 나타나는 효과들은 사전 계산을 통해 어느정도 표현할 수 있지만, 이 경우 많은 제약(움직이지 않는 정적인 광원이나 물체에 대해서만)이 걸리게 되므로 특별한 알고리즘이 필요로 하게 됩니다.

최근의 연구들을 살펴보면, 하드웨어 적인 가속을 통하여 제한적으로 오프라인 렌더링을 진행하고, 신경망 학습 알고리즘을 이용한 업 스케일링(Up scaling)과 노이즈 필터링을 통하여 실시간으로 오프라인 렌더링과 가까운 결과물을 만들어 내기 위한 노력이 이어지고 있습니다. 하지만 여전히 모든 하드웨어에서 이런 기술들을 지원하는 것이 아니고, 아직 까진 실시간으로 모든 것을 처리하기에는 부족합니다. 그러므로 저희는 하드웨어에 독립적인(특히 NVIDIA의 RTX 하드웨어) 렌더링 알고리즘을 이용하여 전역 조명 효과를 근사화 하는 오픈소스 실시간 렌더러를 개발하는 것을 목표로 설정하게 되었습니다.

 국민대학교 소프트웨어학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Lit	
	팀 명	Lit	
	Confidential Restricted	Version 1.0	2021-JUN-09

2.2 연구/개발 내용 및 결과물

2.2.1 연구/개발 내용

본 프로젝트를 수행 할 때 중점적으로 생각한 NVIDIA RTX 플랫폼에 독립적으로 동작하는 Global Illumination 솔루션을 제공하는 것은 OpenGL을 이용하여 해결하였다.

우선 프레임워크 제작에는 Renderer가 필요로 하는 정보들을 Shader에 전달하는 것을 목표로 객체지향 프로그래밍을 하였다. 렌더링 시 필요한 Model, Camera, Light 클래스로 구성되는 Scene이라는 클래스를 생성해서 기본적인 Scene Management를 하였다.

Renderer의 경우 Local Illumination을 구현할 때에는 Shader를 사용하여 Geometry Pass, Shadowing Pass, Lighting Pass와 같은 Render Pass들을 거쳐서 최종적인 각 픽셀의 색을 결정하였고, Global Illumination을 구현할 때에는 Shader를 사용하여 Shadowing Pass, Voxelization Pass, Mipmap Generation Pass, Cone Tracing Pass와 같은 Render Pass들을 거쳐서 최종적인 각 픽셀의 색을 결정한다.

Geometry Pass의 경우 화면의 Geometry Data(position, normal, albedo 등)를 텍스처화 해서 버퍼에 저장하는 과정이다.

Lighting Pass의 경우 Geometry Pass에서 생성된 텍스처를 이용하여 Screen Filled Quad를 생성하여 거기에 덮어 씌운 후 빛에 대한 계산을 하는 과정이다.

Shadowing Pass의 경우 빛의 관점으로 화면의 깊이를 나타내는 DepthMap을 만드는 과정이다.

Voxelization Pass의 경우 전체 화면의 Object의 Radiance를 3D texture로 voxelize하는 과정이다.

Mipmap Generation Pass의 경우 Voxel Cone Tracing에서 Cone을 sampling하기 위해서 존재하는 과정으로 이미 존재하는 Mipmap Generation방법 대신 Compute Shader로 Parallel Reduction을 사용하여 더 빠르게 생성할 수 있도록 하였다.

Cone Tracing Pass의 경우 Cone의 빛을 추적해서 최종적인 색을 결정하는 과정이다.

UI를 만드는 데에는 Dear ImGui를 사용하여 각 기능들을 켜고 끄는 등의 툴과 단축키를 알려주는 툴팁, 초당 프레임율을 알려주는 등의 기능을 넣었다.

2.2.2 시스템 기능 요구사항

NVIDIA RTX 플랫폼에 독립적으로 동작 - 완료


OpenGL 기반 구현 - 완료

Local Illumination (Deferred Rendering) - 완료

Voxelization - 완료

Global Illumination (Voxel Cone Tracing) - 완료

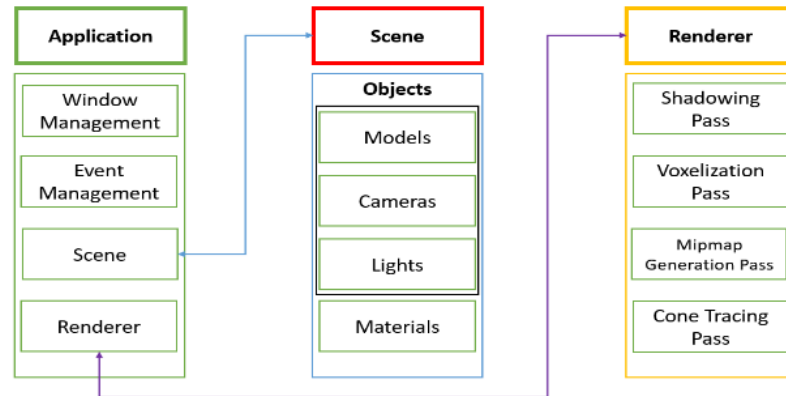
UI (Dear ImGui) - 완료

 국민대학교 소프트웨어학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Lit	
	팀 명	Lit	
	Confidential Restricted	Version 1.0	2021-JUN-09

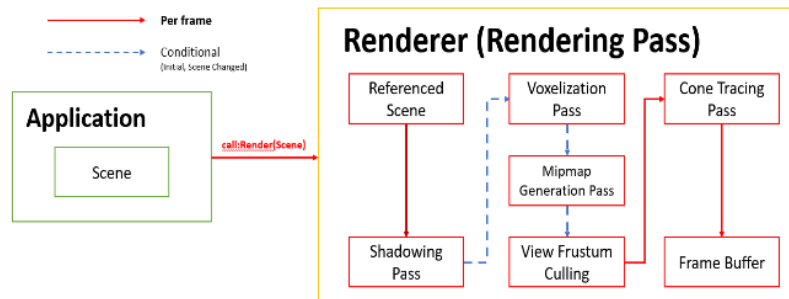
오픈소스 Renderer (MIT License) – 완료

2.2.3 시스템 구조 및 설계도

Framework



Renderer




2.2.4 활용/개발된 기술

Deferred Rendering : 물체의 position, normal, albedo 정보를 미리 하나의 Render Pass 를 거쳐서 텍스처화 하고 이후에 Light 에 대한 계산을 하는 Render Pass 를 거쳐 렌더링을 하는 기술

Depth Map : 빛의 관점에서 화면을 렌더링해서 깊이를 측정해서 텍스처화 하는 기술

Voxelization : 일반적인 화면은 2D 텍스처로 구성되는데 픽셀을 3D 텍스처인 Voxel 로 만드는 기술

 국민대학교 소프트웨어학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Lit	
	팀 명	Lit	
	Confidential Restricted	Version 1.0	2021-JUN-09

Voxel Cone Tracing : Voxelize 된 화면의 각 voxel 단위로 Cone 을 생성해서 Cone 의 빛을 추적하는 기술

View Frustum Culling : 카메라에 보이지 않는 물체들은 render 하지 않는 것으로 cone tracing 과 draw call 의 overhead 를 감소시켜서 프레임을 올려주는 기술

ImGui : 화면 위에 GUI 를 띄우는 기술

2.2.5 현실적 제한 요소 및 그 해결 방안

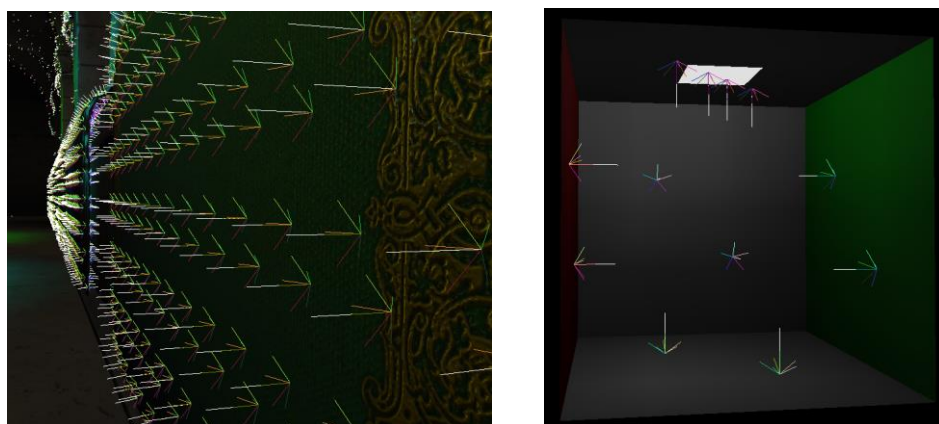
현재의 경우 Indirect Light Bounce가 1번뿐인데 Light Propagation Volume을 이용하거나 (N-1)번의 빛의 bounce를 simulate하는 compute shader를 통해서 Voxel Cone Tracing 에 두번째 bounce를 만들어볼 수 있을 것 같다.

또한 현재의 경우 Octree를 이용해서 Voxelization을 구현하였으나, Clip=map을 이용해서 memory footprint를 줄여볼 수 있을 것 같고, Normal, Albedo, Opacity와 같은 Geometry 데이터를 분리해서 빛이 물리적 quantity unit을 가지는 복잡한 장면을 다룰 수 있게 할 수 있을 것 같다.


더 나은 결과를 내기 위해서 DOF, Bloom, Exposure, Bokeh등과 같은 post-process효과 들을 구현해볼 수 있을 것 같다.

마지막으로 BSDF와 같은 더 유연하고 물리적으로 더 사실적인 Function들을 찾아볼 수 있을 것 같다.

2.2.6 결과물 목록



Visualized diffuse Cone's Directions


 <div> 국민대학교 소프트웨어학부 캡스톤 디자인 I </div>	결과보고서		
	프로젝트 명	Lit	
	팀 명	Lit	
	Confidential Restricted	Version 1.0	2021-JUN-09

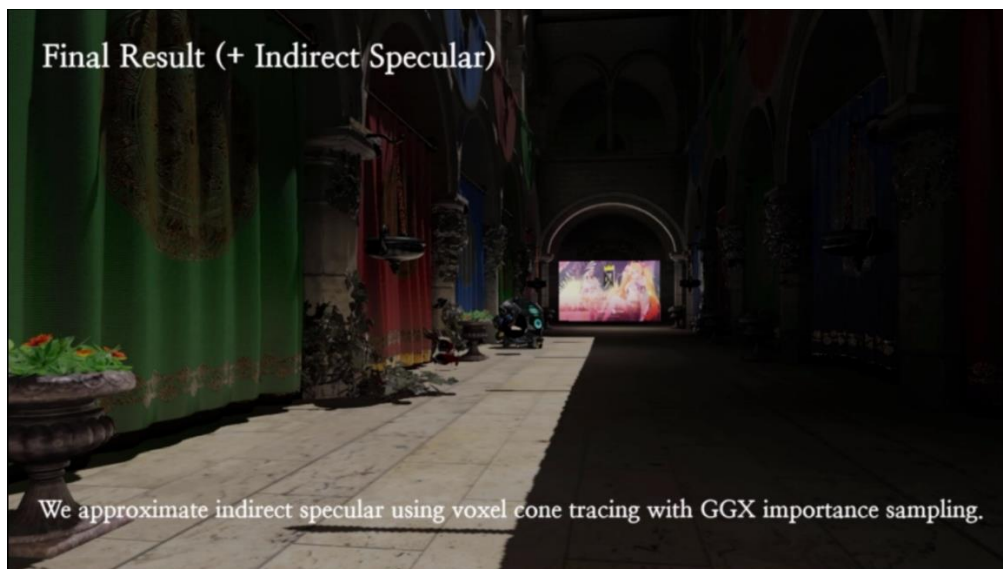


Axis Aligned Bounding Box (AABBs)



Emissive + Direct Diffuse Reflection


 국민대학교 소프트웨어학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Lit	
	팀 명	Lit	
	Confidential Restricted	Version 1.0	2021-JUN-09



Final Result

2.3 기대효과 및 활용방안


NVIDIA RTX 하드웨어 유닛이 없어도 비슷한 결과물과 더 나은 퍼포먼스를 보여주는 Renderer 를 개발함으로써 해당 하드웨어 유닛이 없어도 Global Illumination 효과를 낼 수 있다는 것을 보여줄 수 있을 것이다.
이를 활용하여 미래의 또 다른 프로젝트의 기초적인 Framework 로 활용할 수 있다.

 국민대학교 소프트웨어학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Lit	
	팀 명	Lit	
	Confidential Restricted	Version 1.0	2021-JUN-09

3 자기평가

자 체 평 가 서

팀명	Lit	성명	곽상열
프로젝트제목	Lit		
자체 평가	<p>프로젝트 수행 소감: 프로젝트를 수행하면서 GPU를 이용한 프로그램이에 대한 더 깊은 이해가 가능했고, Renderer의 기초가 되는 Framework를 구성하는 방법에 대해서 알아볼 수 있었다. 또한 프로젝트를 진행하는 것이 코드를 구현하는 것 뿐 아니라 문서적으로도 할 일이 많다는 것도 느끼게 되었다. 그리고 팀원과의 협업을 할 때에 어떻게 해야 하고 의견 교류할 일이 있으면 어떻게 해야 하는지에 대해서도 배울 수 있었다. 또한 협업 시 서로 진행상황에 대한 이야기가 얼마나 중요한지도 알 수 있었다. 그래서 나중에 취업을 해서도 팀 프로젝트를 하게 되었을 때 팀원들과 지켜야 할 규칙 같은 것들을 어떻게 설정 하는지와 다른 사람이 코드를 보았을 때 이해할 수 있도록 주석을 제대로 다는 방법 등을 따로 공부하지 않아도 써먹을 수 있을 것 같다.</p> <p>개선요구사항: 조금 더 기간이 길었다면 더 좋은 결과물을 만들 수 있지 않았을까 하는 아쉬움이 있다.</p> <p>발전적 제언: 다음에는 조금 더 열심히 공부해서 조금 더 많은 기능을 구현해보았으면 좋겠다. 이번에도 기본적인 Framework와 Deferred Rendering을 구현한다는 측면에서 많이 배우긴 했지만 핵심적인 기술이었던 Voxel Cone Tracing에는 팀원의 노력이 더 많이 들어갔다는 생각을 가지고 있다. 따라서 조금 더 프로그래밍 실력을 길러서 Voxel Cone Tracing을 직접 구현해보는 등의 공부도 해야 할 것 같다.</p>		


 국민대학교 소프트웨어학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Lit	
	팀 명	Lit	
	Confidential Restricted	Version 1.0	2021-JUN-09

자 체 평 가 서

팀명	Lit	성명	양교원
프로젝트제목	Lit		
자체 평가	<p>프로젝트 수행 소감: 학기중에 직접 자신이 원하는 주제를 직접 정하고, 프로젝트를 팀원을 꾸려서 진행 해볼 수 있었다는 점이 정말 좋았다. 다만 다른 모든 일정을 겸하면서 프로젝트를 진행해야 했다는 점. 그리고 이러한 프로젝트들의 수행 여부나 기타 부가적인 시간 투자가 필요로 하는 일종의 과제들이 강제되어서 조금 힘들었던 것 같다. 또한 중간에 프로젝트의 의의를 다시 정하고 진행하여서 조금 불안은 했지만. 최종적으로 만족스러운 결과물을 만들어 냈다는 점은 매우 좋았다고 생각한다.</p>		
	<p>개선요구사항: 일단 기본적으로 작성하여야 하는 문서의 수를 간소화 하거나, 오히려 기본적인 논문 형태로 개선하면 좋을 것 같다. 현재 작성하여야 하는 문서에는 일종의 서비스를 개발하고 설명하는 것을 목적으로 작성되어 있는 느낌이 강한데. 추가적으로 자신들이 무엇을 공부했고 어떻게 개선하였고 앞으로 개선해내야 할 것들이 무엇이 있는지에 대해 정리하는 논문 형식의 템플릿을 추가해주면 좋을 것 같다. 그리고 발표자료에 대한 리뷰가 더 자주 이루어지면 좋겠다.</p>		
	<p>발전적 제언: 발표할 때 사용할 프레젠테이션을 만들 때 정보를 더 잘 정리하는 법이나, 어느정도의 선까지 설명을 해야 하는지에 대해서 좀 더 알 수 있었으면 좋았을 것 같다.</p>		

4 참고 문헌

번호	종류	제목	출처	발행년도	저자	기타
1	논문	Interactive Indirect Illumination Using Voxel Cone Tracing	Computer Graphics Forum	2011	Crassin, C.	
2	서적	OpenGL Insights – Sparse Voxelization	CRC Press	2012	Patrick Cozzi and Christophe Riccio	
3	기술문서	Physically Based Rendering in Filament	Google		Romain Guy, Mathias Agopian	https://google.github.io/filament/Filament.html#about/authors
4	기술문서	Fast Extraction of Viewing Frustum Planes from the World-View-Projection Matrix			Gill Gribb, Klaus Hartmann	
5	영상	TU Wien Rendering	TU Wien	2015	TU Wien	https://youtube.be/pjc1QAI6zS0
6	영상	SIGGRAPH University – Introduction to “Physically Based Shading in Theory and Practice”	SIGGRAPH	2016	Naty Hoffman	https://www.youtube.com/embed/j-A0mwsJRmk?start=69
7	웹페이지	Fixing frustum culling	Inigo Quilez	2013	Inigo Quilez	https://iquilezles.org/www/articles/frustumcorrect/frustumcorrect.htm

 국민대학교 소프트웨어학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Lit	
	팀 명	Lit	
	Confidential Restricted	Version 1.0	2021-JUN-09

8	웹 페이지	Voxel Cone Traced Global Illumination	Leif Erkenbrach	2015	Leif Erkenbrach	https://leifnode.com/2015/05/voxel-cone-traced-global-illumination/
9	웹 페이지	Learn OpenGL	Learn OpenGL	2016		https://learnopengl.com/
10	웹 페이지	Dear ImGui	Github			https://github.com/ocornut/imgui

5 부록

5.1 사용자 매뉴얼

실행 가이드

1. Projects 폴더 내 opengl_pbr.sln 파일을 Visual Studio 로 실행
2. 빌드를 진행
3. Build 폴더 내 Render.exe 파일 실행

단축키 가이드

UI 켜기 끄기 : U

카메라 움직이기 : W A S D

빛 움직이기 : 상하좌우 화살표

미리 정의된 카메라움직임 실행 : C

Light Rotation 실행 : L

Render Mode 바꾸기 : Left Bracket Right Bracket

Sponza 로 화면 전환 : F5

CornellBox 로 화면 전환 : F6

 국민대학교 소프트웨어학부 캡스톤 디자인 I	결과보고서		
	프로젝트 명	Lit	
	팀 명	Lit	
	Confidential Restricted	Version 1.0	2021-JUN-09

5.2 운영자 매뉴얼

새로운 Scene 구성 시 Scene 클래스를 상속하는 클래스를 생성하고 클래스의 멤버변수로 model, light, camera 클래스를 넣어주고 position, scale, rotation 등 기본 정보들을 지정해주는 클래스를 만들어주면 된다.

5.3 테스트 케이스

2.2.6 의 결과물 목록의 기능 일람표를 기준으로 작성된 테스트 케이스와 해당 테스트 케이스로 실제 테스트한 결과를 성공/실패로 작성한다. 필요에 따라 분류 체계(대분류/소분류)는 보다 상세히 할 수 있으며, 테스트 방법 및 기대 결과는 최대한 상세히 기술한다.

대분류	소분류	기능	테스트 방법	기대 결과	테스트 결과
디버깅	Diffuse Cone Direction	Cone 의 방향을 시각화 하여 디버깅 진행	UI 에서 Debug Cone Direction 을 활성화한다.	Diffuse Cone Direction 이 시각화 된다.	성공
디버깅	Axis Aligned Bounding Box	Axis Aligned Bounding Box 를 시각화 하여 디버깅	UI 에서 Debug BoundingBox 를 활성화 한다.	Axis Aligned Bounding Box 가 시각화 된다.	성공
렌더링	Local Illumination	Indirect Light 에 대한 고려 없이 Direct Light 만 고려하여 픽셀의 색 계산	UI 에서 Direct Diffuse 만 활성화한다.	Direct Light 만 고려하여 렌더링 된다.	성공
렌더링	Global Illumination	Direct Light 와 Indirect Light 를 모두 고려하여 픽셀의 색 계산	UI 에서 모든 reflection 을 활성화한다.	Direct Light 와 Indirect Light 를 고려하여 렌더링 된다.	성공