Seung Min Choi

-JF Engine-

Technical Manual

내용

[서론 3](#_Toc469433700)

[본 문서의 목적 3](#_Toc469433701)

[제작 목적 3](#_Toc469433702)

[참고한 레퍼런스들 3](#_Toc469433703)

[1. MSDN ( MicroSoft Developer Network ) 3](#_Toc469433704)

[2. DirectX 11을 이용한 3D 게임 프로그래밍 입문 3](#_Toc469433705)

[3. 셰이더 프로그래밍 입문서 3](#_Toc469433706)

[4. REAL – TIME RENDERING 2 3](#_Toc469433707)

[엔진 세부 설명 4](#_Toc469433708)

[FrameWork 4](#_Toc469433709)

[SmartPointer 4](#_Toc469433710)

[App 4](#_Toc469433711)

[Exception 5](#_Toc469433712)

[Scene 5](#_Toc469433713)

[Scene 5](#_Toc469433714)

[Graphic API 6](#_Toc469433715)

[RenderState 6](#_Toc469433716)

[DirectX11Device 7](#_Toc469433717)

[Component 8](#_Toc469433718)

[GameObject 9](#_Toc469433719)

[Camera 9](#_Toc469433720)

[Tansform 10](#_Toc469433721)

[Light 11](#_Toc469433722)

[Colision 12](#_Toc469433723)

[Renderer 13](#_Toc469433724)

[Mesh & SkyBox 14](#_Toc469433725)

[Object Data 15](#_Toc469433726)

[Material 15](#_Toc469433727)

[Texture 15](#_Toc469433728)

[Rendering Pipeline 15](#_Toc469433729)

[1. G-Buffer ( GeometryBuffer ) 15](#_Toc469433730)

[2. SSAO And SSAO Blur 16](#_Toc469433731)

[3. Light Buffer 17](#_Toc469433732)

[4. Shadow Buffer 18](#_Toc469433733)

[5. Forward Render 19](#_Toc469433734)

[차후 개발 계획 20](#_Toc469433735)

[엔진 20](#_Toc469433736)

[랜더링 20](#_Toc469433737)

[IDE 20](#_Toc469433738)

# 서론

## 본 문서의 목적

* JF Engine을 기반으로 엔진을 제작 하려는 사람들의 이해를 돕기 위해 작성했습니다
* 또한 차후에 JF Engine을 협업 제작함에 있어서 참고자료로 쓰일 수 있습니다.

## 제작 목적

* 연구 개발 및 다양한 기술들을 적용해 보기 위해 제작.
* 차후 멀티 플랫폼 지원 및 특정 장르에 특화된 Template을 제공 할 수 있도록 제작 예정.

## 참고한 레퍼런스들

1. MSDN ( MicroSoft Developer Network )

* C++, DirectX 11에 대한 Document Reference들 및 예제들을 얻을 수 있다.

1. DirectX 11을 이용한 3D 게임 프로그래밍 입문

* DirectX 11 API를 통한 3D 프로그래밍의 기초적인 이론들에 대해 서술.

1. 셰이더 프로그래밍 입문서

* 기본 셰이더 기법들에 대한 이론 및 구현에 대해 서술.

1. REAL – TIME RENDERING 2

* 3D 프로그래밍에 대한 이론 및 기술 서술.

# 엔진 세부 설명

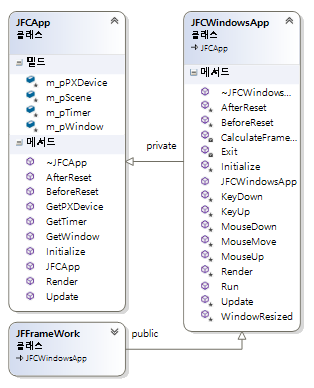
* 엔진에 대해서 구조 및 세부적 기술 설명을 하는 부분입니다.
* 프레임 워크부터 시작해서 각각의 구현 설명을 합니다.

## FrameWork

### SmartPointer

모든 동적 할당 받는 객체들은 Shared\_ptr을 통해서 참조하고 있는 객체가 없을 때 자동으로 소멸되고있습니다.

### App

-JFCApp-

해당 클래스에서는 플랫폼에 상관없이 공통적으로 지닐 수 있는 함수 및 객체들이 존재합니다.

( 아직 완벽하게 멀티 플랫폼을 지원하지 않습니다.)

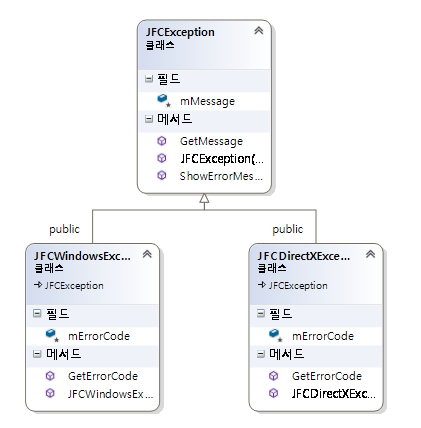
-JFCWindowsApp-

해당 클래스에서는 Input이나 Window 생성 Window Size 조절 등과 같이 Windows용으로 처리하도록 되어있습니다.

-JFFrameWork-

해당 클래스는 플랫폼에 해당하는(JFCWindowsApp 같은) 클래스를 상속받아서 Engine 설정을 할 수 있는 클래스입니다. Main 에서 해당 클래스 생성 후 Run을 호출하면 엔진이 시작됩니다.

### Exception

-JFCException-

기본적인 예외처리로 사용가능한 메시지 변수와 함수가 존재합니다.

-JFCWindowsException-

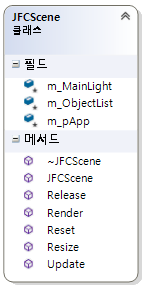
윈도우 MessageBox를 사용 가능한 Exception Class 입니다.

-JFCDirectXException-

DirectX Error 코드를 표현 가능한 Exception Class 입니다.

## Scene

### Scene

Unity / Unireal Engine 에 존재하는 Scene 과 같습니다.

물체들의 배치정보 설정 정보들을 담고있습니다.

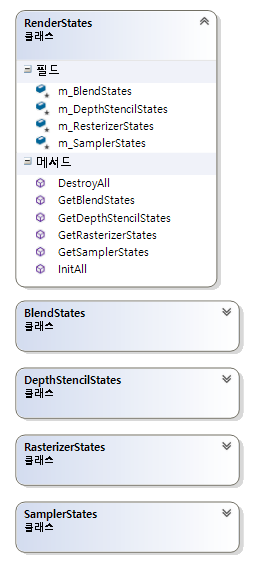
반투명 물체와 그림자에 영향을 주는MainLight ( GlobalLight )를

설정 가능합니다.

## Graphic API

* 그래픽 관련 Class들에 대한 설명입니다.

### RenderState

-RenderStates-

랜더링에 파이프라인에 적용 가능한 State들을 생성/삭제 합니다.

-BlendStates-

색상 혼합방식에 관련된 설정을 제공합니다.

(ex. AlphaBlend, NoColor … )

-DepthStencilStates-

Depth와 Stencil 설정을 제공합니다.

(ex. StencilEquals, DepthDisabled …)

-RasterizerStates-

Resterizer에서 사용되는 설정을 제공합니다.

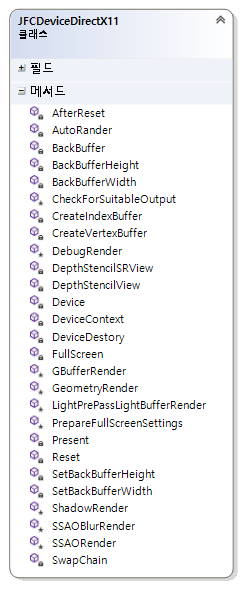
(ex. NoCull, FrontFaceCull …)

-SamplerStates-

텍스처를 읽을 때 사용할 설정을 제공합니다.

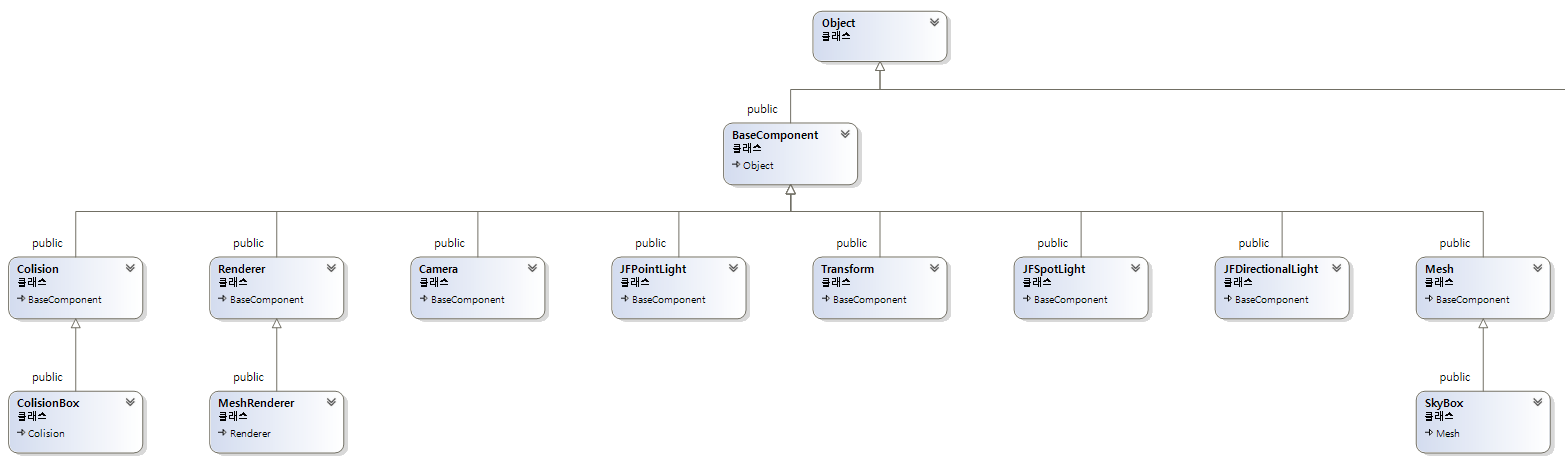
(ex. Linear, Point …)

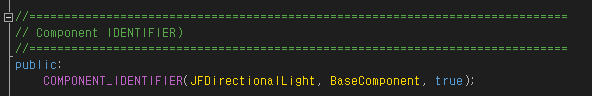
### DirectX11Device

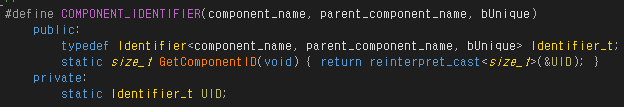
DirectX11 에 관련된 API 들이 정의되어 있는 클래스입니다.

다중 그래픽 Device를 지원하기 위해서 모든 랜더링은 Scene에 배치된 GameObject들을 전달 받아서 해당 Class 안에서 이루어지고 있습니다.

## Component







* 기본적으로 모든 Component는 BaseComponent를 상속 받습니다.
* Component들을 식별하는 방식에는 TemplateMeta를 적용했으며 Component를 제작 할 때는

해당 매크로를 써 주어야합니다.

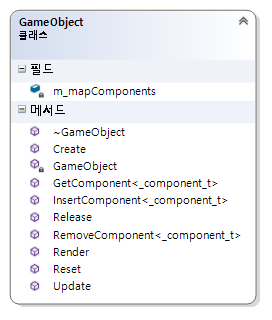
(std::\_if 등 TemplateMeta 를 사용함으로써Component 들을 찾을 때 드는 시간을 컴파일 시간으로 옮겼습니다.)

* Component 식별ID 는 Static 멤버 변수의 메모리 주소를 사용했습니다.
* 여기서 첫번째 파라미터는 제작할 Component의 Class 이름

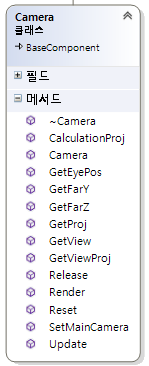
두번째는 부모 Componen의 Class 이름

세번째는 Component를 찾을 때 자신의 클래스에 자식이 존재하지 않는다면 True로 설정하면 됩니다.

### GameObject

Component들을 GameObject에 Insert/Remove 하는 식으로 붙여 나가면서 원하는 객체를 만들 수 있습니다.

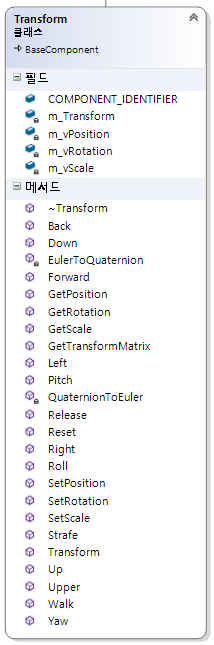
### Camera

Scene에서 보이는 화면을 설정 할 수 있는 카메라 Component 입니다.

MainCamera로 설정된 카메라는 조작을 통해서 움직일 수 있습니다.

이후 카메라의 영역지정을 가능하게 해서 다중 카메라 화면을 한 Scene에 표시 가능하게 할 예정입니다.

### Tansform



해당 클래스를 포함시킨 객체는 이동, 회전, 크기 조절이 가능해집니다.

또한 위치, 회전, 크기 값들을 얻어서 사용할 수 도 있습니다.

회전 수치는 Quaternion을 사용하며 Euler와의 상호 교환이 가능합니다.

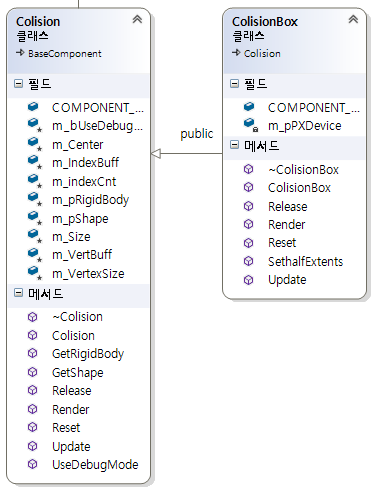
### Light

가장 많이 사용하는 3가지 조명을 Component화 했습니다.

Transform 과 조합해서 사용 할 수 있습니다.

이후 방향까지도 Transform 을 통해서 가능하도록 수정 예정.

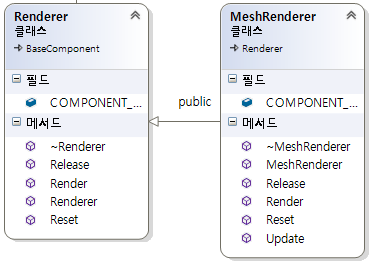
### Colision

해당 Component를 포함한 객체는 충돌체크 및 물리 효과를 사용 가능하게됩니다.  
충돌 체크를 할 때 사용할 도형의 크기를 설정할수 있습니다.

기본적으로 Transform의 위치, 회전, 크기 의 영향을 받습니다.

차후 Box이외에 다양한 도형 및 기능을 지원할 예정입니다.

### Renderer

-Renderer-

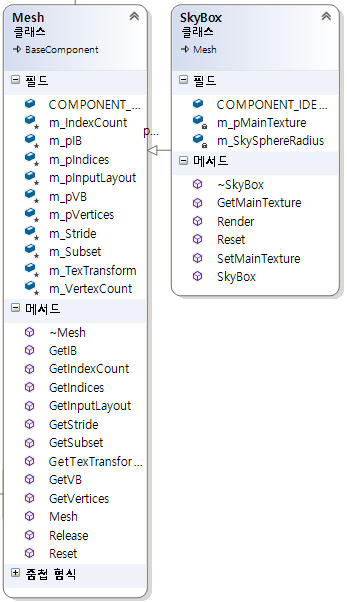
화면에 물체가 그려질 수 있도록 Render 정보를 제공하는 객체입니다.

현재 Render에서 하는 일이 없지만 이후 Command 방식을 도입하면 해당 객체가 그려 질 정보를 Command Queue 에 전달하는 역할을 할 예정입니다.

-MeshRenderer-

Mesh를 가진 객체를 Rendering 합니다.

### Mesh & SkyBox

-Mesh-  
물체의 기하정보를 지니고있습니다.  
Component에는 어울리지 않는 것 같아서 차후 Object 로 변경할 예정입니다.

-SkyBox-

해당 Component는 현재 MainCamera에만 부착 가능합니다.   
이것 역시 이름과 Component 에는 어울리지 않는 것 같아서 차후 CubeMap 으로 이름을 바꾸고 Scene에서 설정 가능하도록 변경 예정입니다.

# Object Data

## Material

물체의 재질 정보와 랜더링 상태 정보를 가지고있습니다.

Unity 의 Material 과 비슷합니다.

## Texture

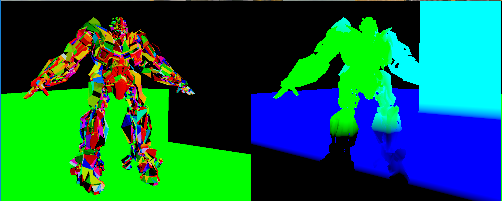
Texture 정보를 가지고있습니다.

차후Object 를 상속할 예정입니다.

# Rendering Pipeline

* 엔진에서 추가로 사용하고있는 Rendering Pipeline 에 대해서 설명합니다.

## G-Buffer ( GeometryBuffer )

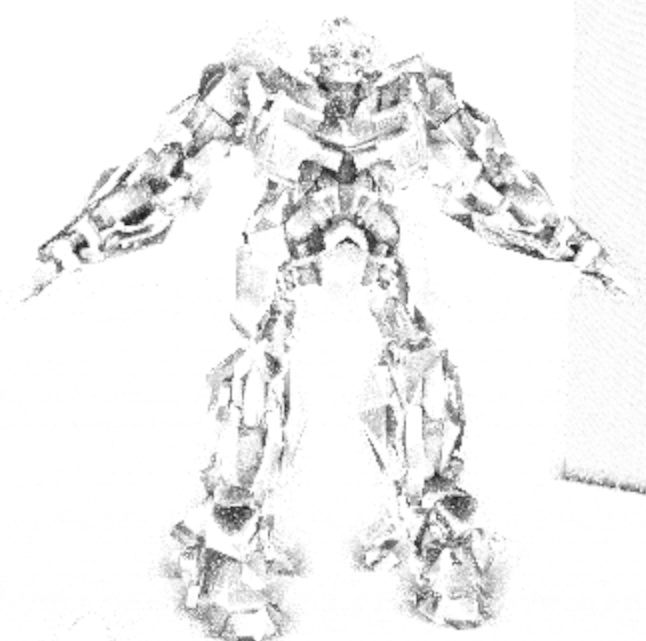
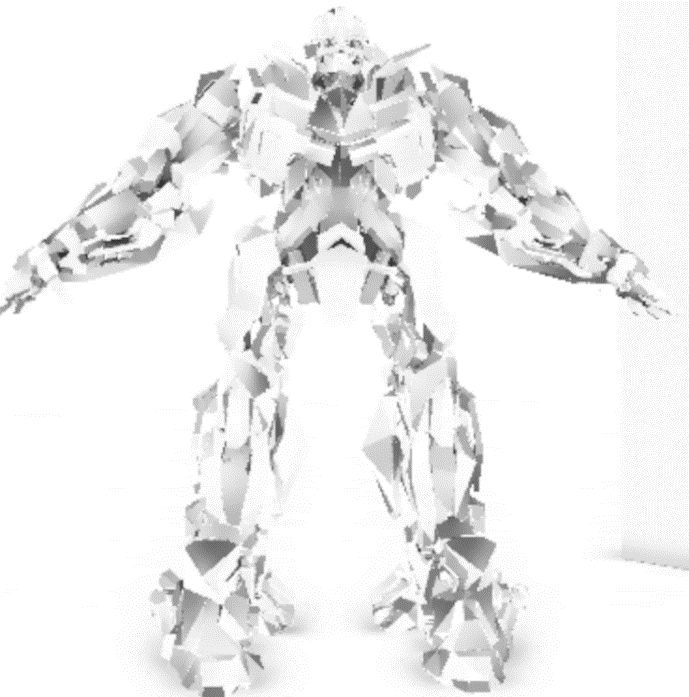


G-Buffer Pipeline 에서는 지연랜더링, SSAO 에서 필요한 정보인

세계공간 Normal, Position, Depth 정보를 담고있습니다. SSAO 에서 사용할 때는 시야공간으로 변환해서 사용합니다.

이후에 Normal map 에 x, y 좌표 정보만 담아서 압축하고 Alpha 정보 와 물체의 재질정보를 추가할 예정입니다.

## SSAO And SSAO Blur

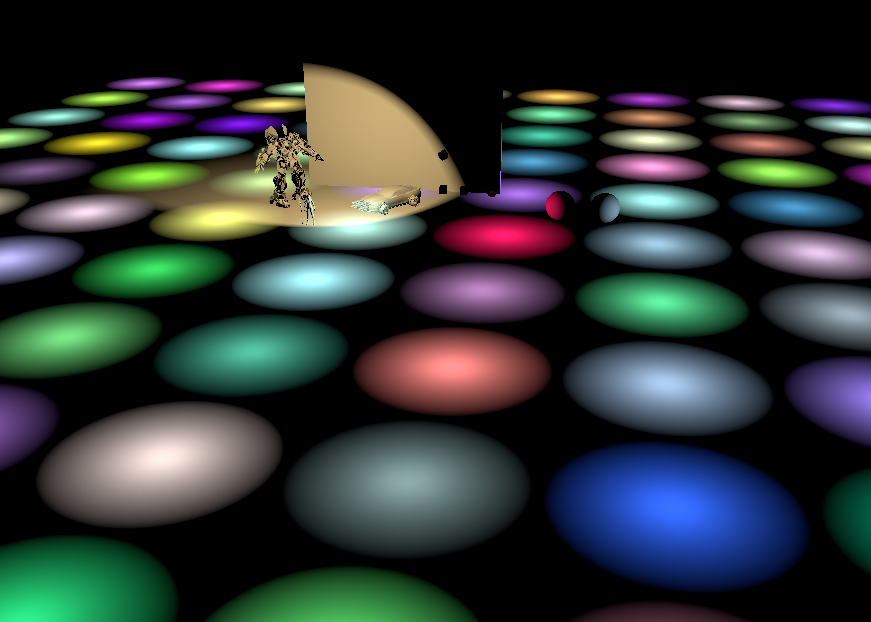


차폐 계산 SSAO 가장자리 보존 흐리기

화면 공간 주변광 차폐를 위해서 물체의 차폐를 계산한 결과를 가장자리 보존 흐리기를 통해서 담아 두고 있습니다.

이후 정적 맵에 미리 반직선 투사를 통한 주변광 차폐의 계산 결과를 텍스처 로 저장해두고 사용 할 수 있도록 추가 예정입니다.

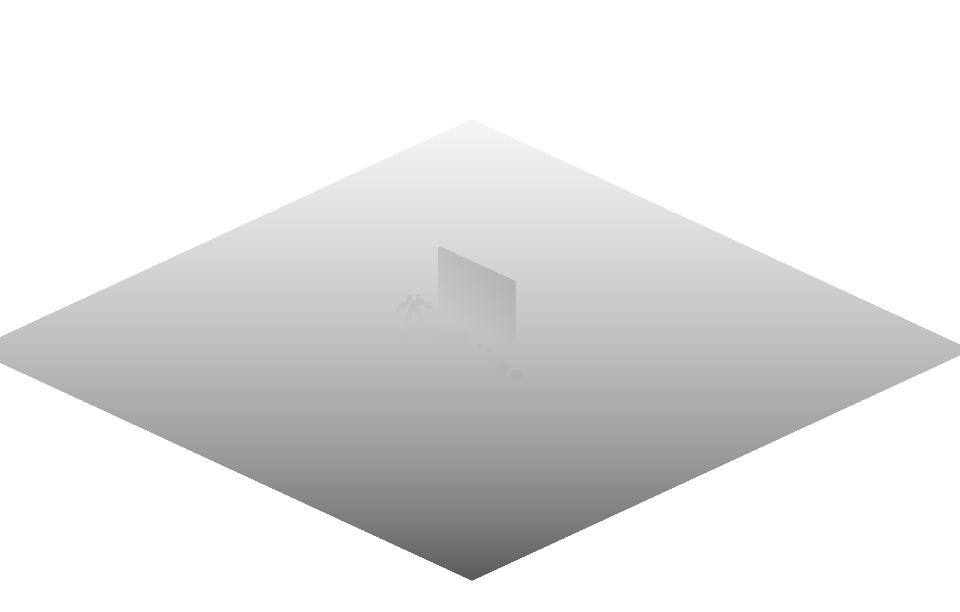
## Light Buffer



많은 Light 를 Rendering 해도 퍼포먼스를 유지하기위해서 Light 계산을 별도의 Pipeline 으로 뺏습니다.

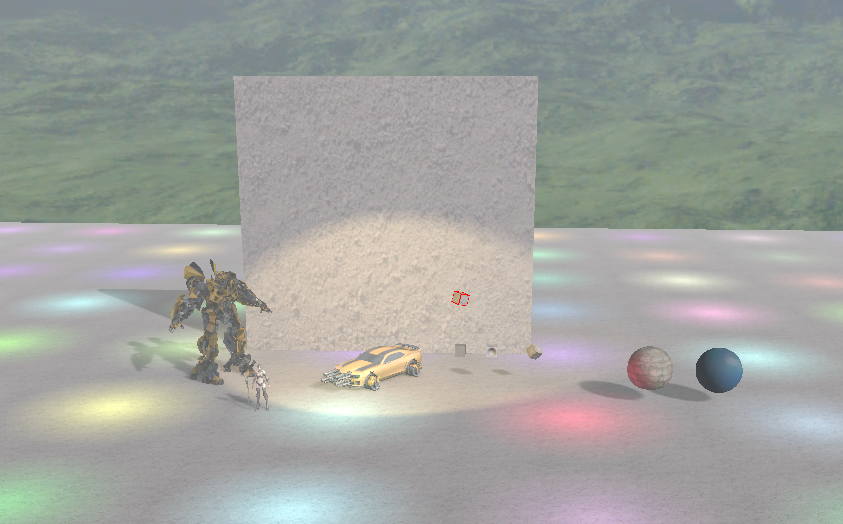
Directional Light 는 Scene 에 3개 초과로 존재 할 수 없다고 생각되어서 Spot Light 와 Point Light 만 계산하고 있습니다.

## Shadow Buffer



그림자를 구현하기 위해서 Scene 에 배치된 Global Light 의 시야기준으로 DepthStencil 을 이용해서 물체들의 깊이 값을 저장했습니다.

## Forward Render



최종적으로 이전 Pipeline 에서 뽑아낸 맵 들을 바탕으로 물체들을 그려줍니다.

Directional Light 의 조명 계산을 해주고 있습니다.

투명 물체를 그려주고 있습니다.

# 차후 개발 계획

## 엔진

* Terrain 구현 및 Seamless로 OpenWord 가 가능하도록 추가.
* GPU 에서 계산 및 생성 하는 Particle 추가.

## 랜더링

* Terrain 추가에 따른 Cascasded Shadow 추가.
* 환경맵에 대기 산란 셰이더 추가.

## IDE

* WPF(c#)를 연동해서 제작.