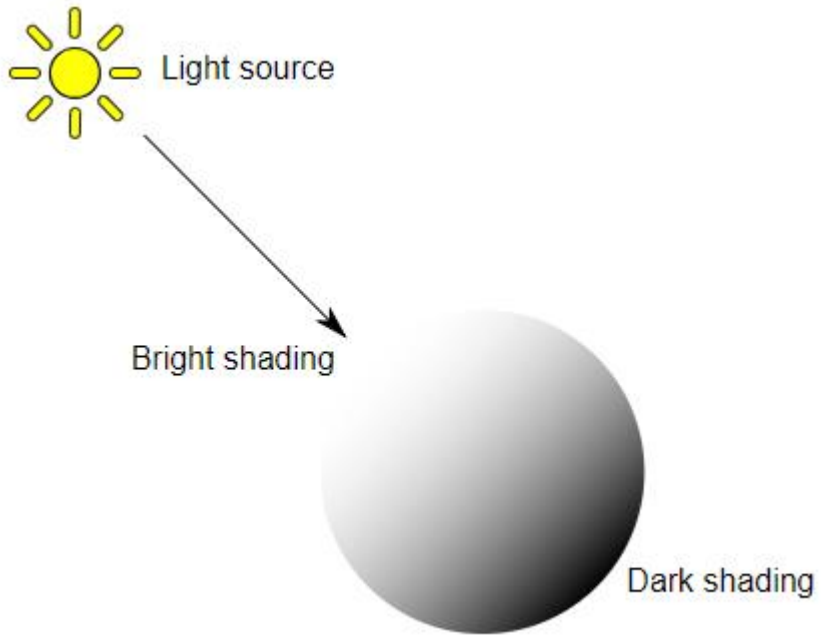
Chap3. Graphics

Unity

LIGHTING

01. Lighting overview



3d object의 shading을 위해서는
빛의 intensity, direction, color를 알아야 함

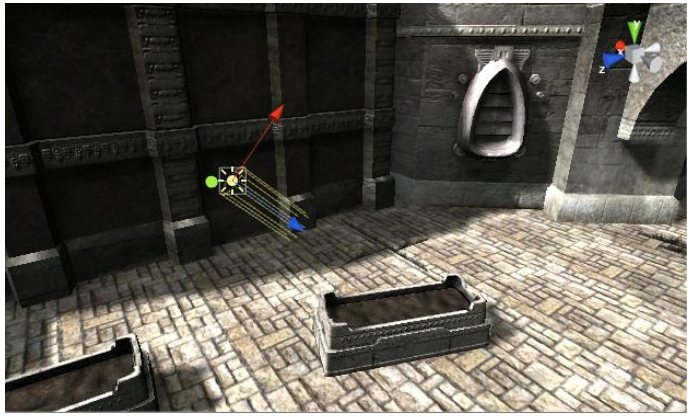
Scene내에 있는 Light object가 이 속성들을 제어

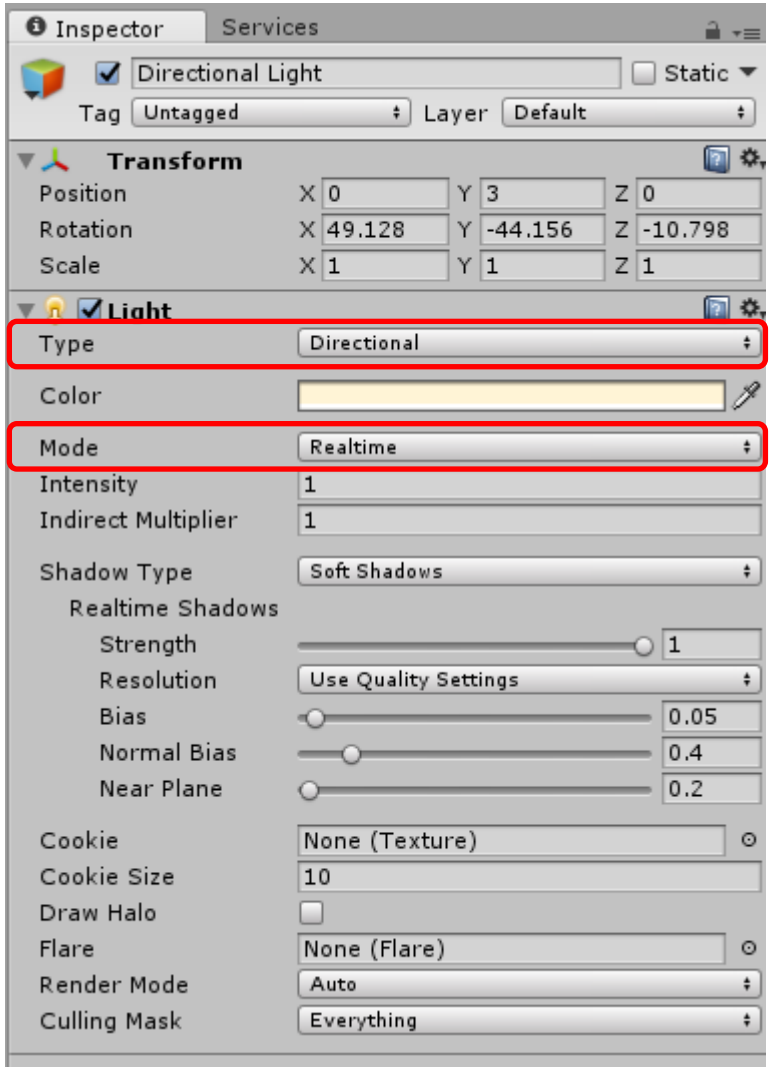
Light를 표현하기 위한 다양한 유형이 있음
(Point, Spot, Directional, Area Light, etc)

Unity에서 Lighting은 크게 realtime과 precomputed로 나뉨

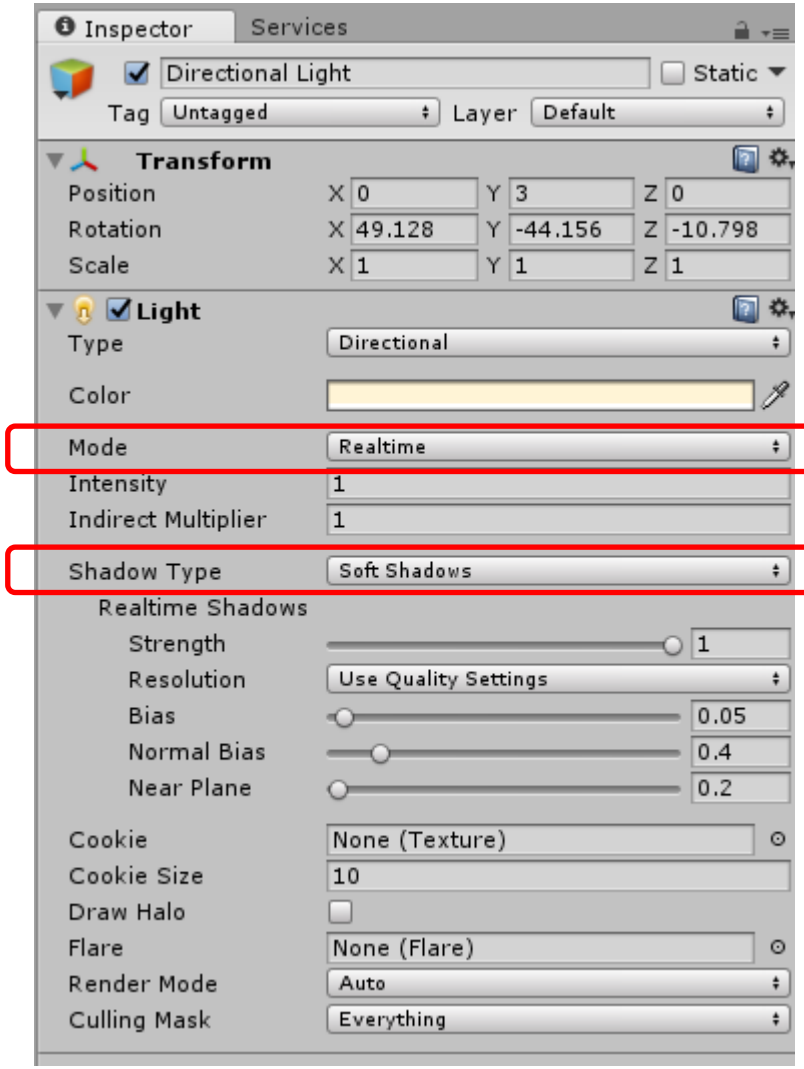
02. Directional Light

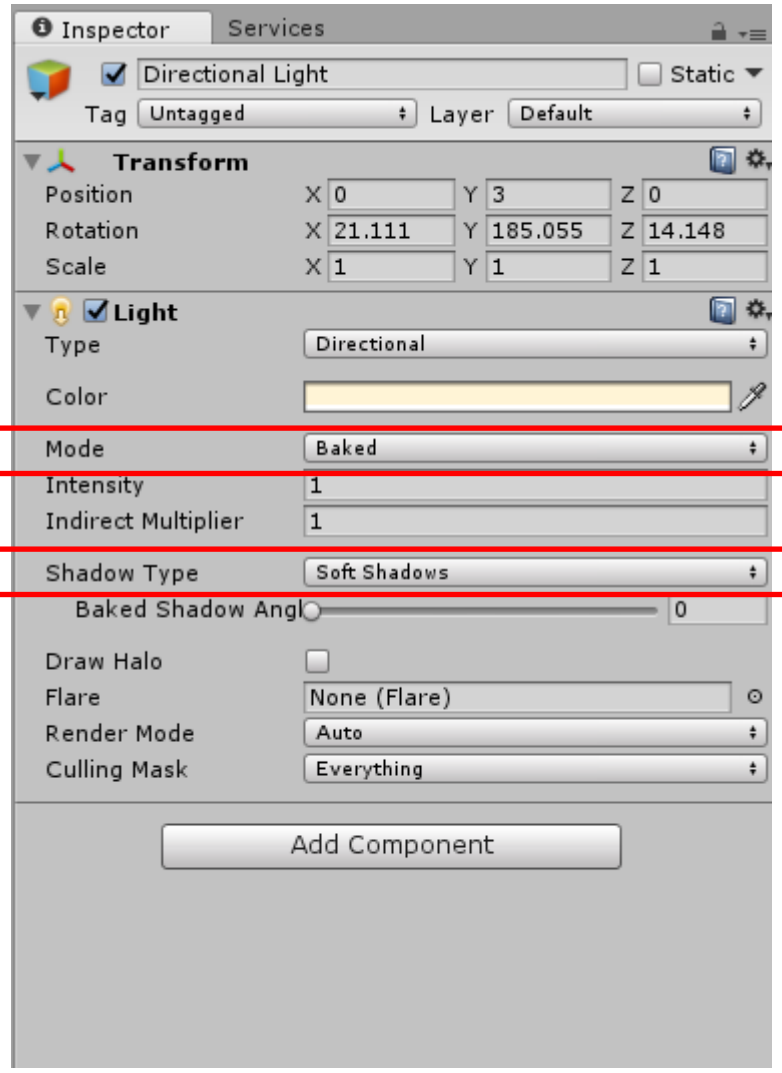
Scene의 모든 공간에 설정한 방향으로 빛을 발산
비추는 각도에 따라 그림자의 방향과 길이가 달라짐

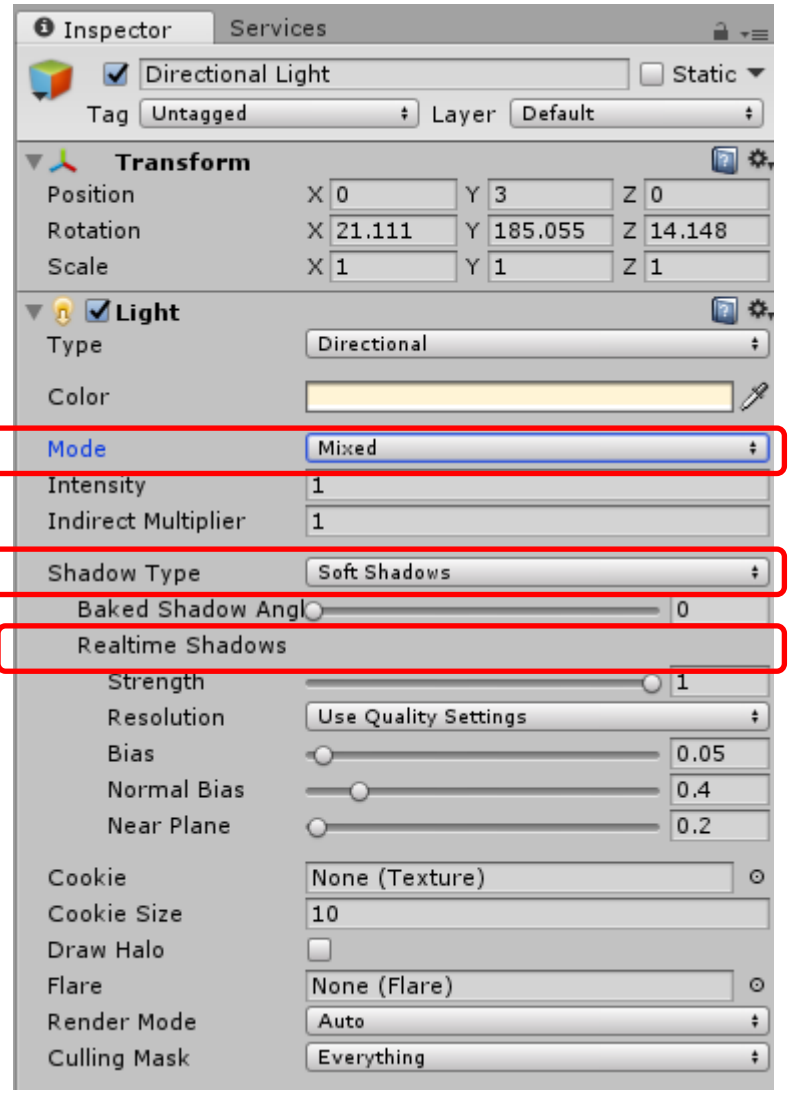




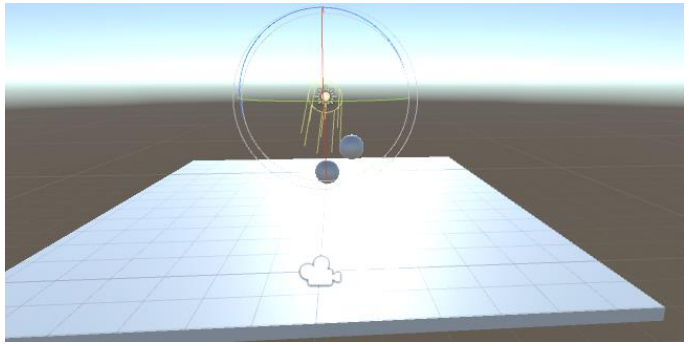
02. Directional Light / Inspector

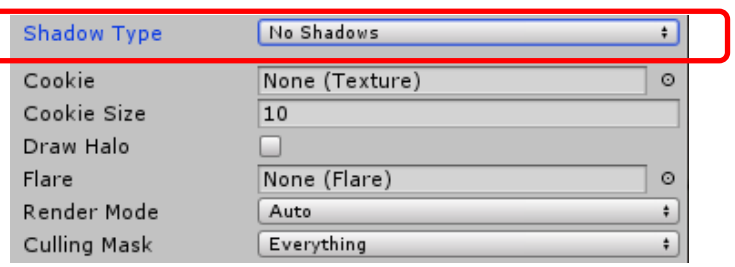


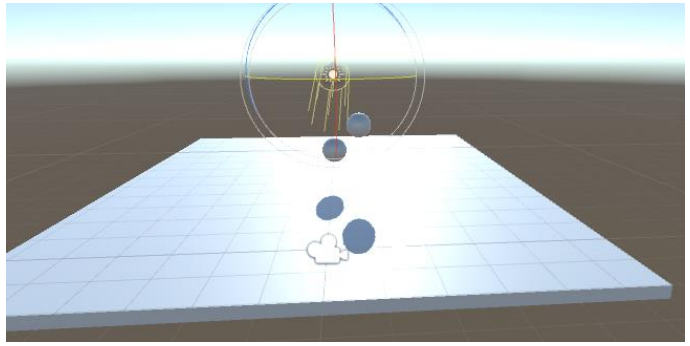


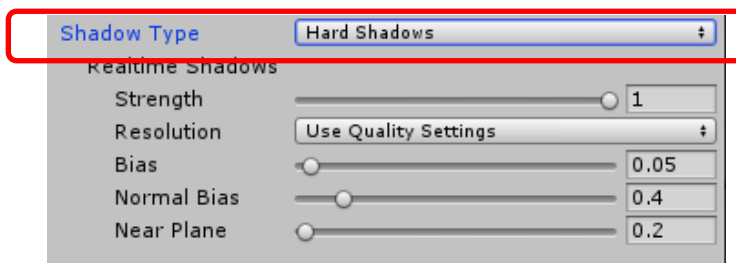


02. Directional Light / Shadow Type

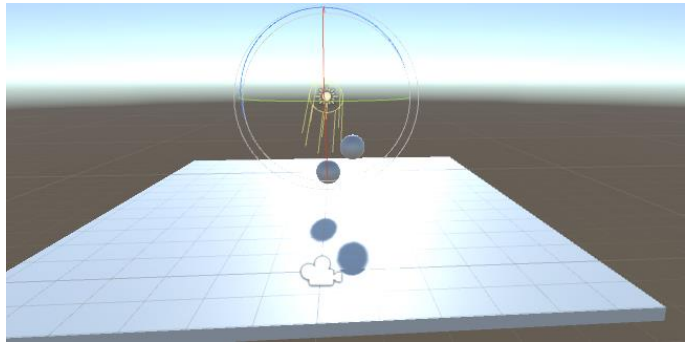






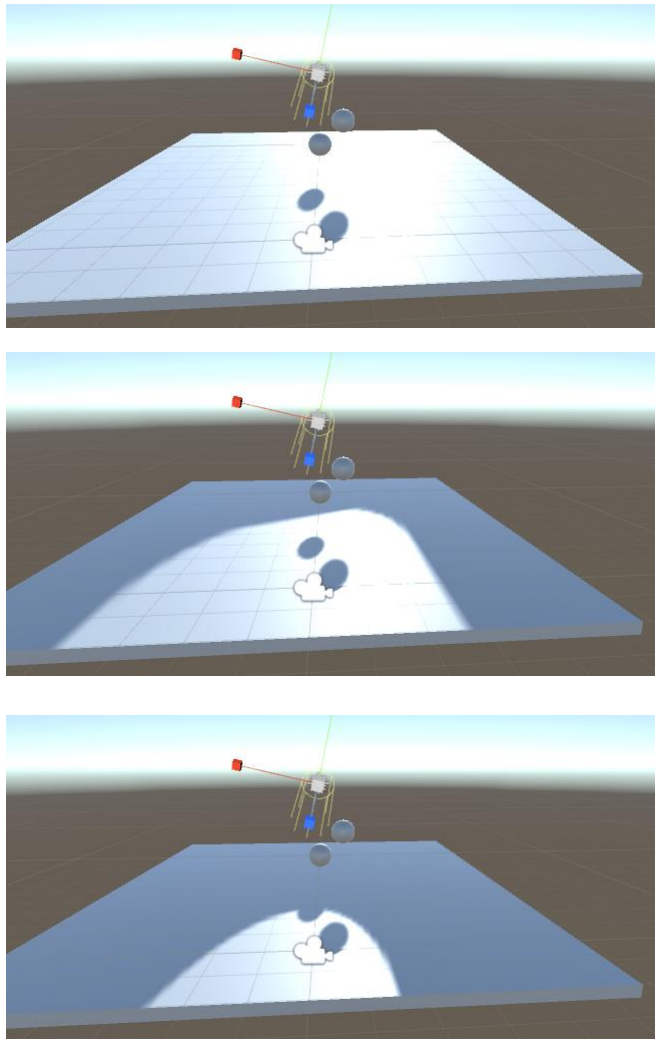


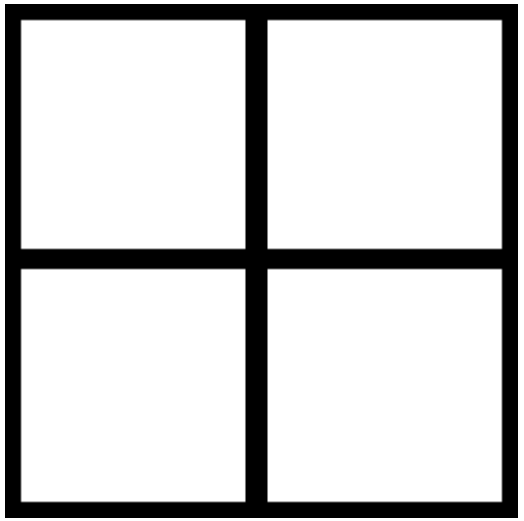
딱딱한 외곽선

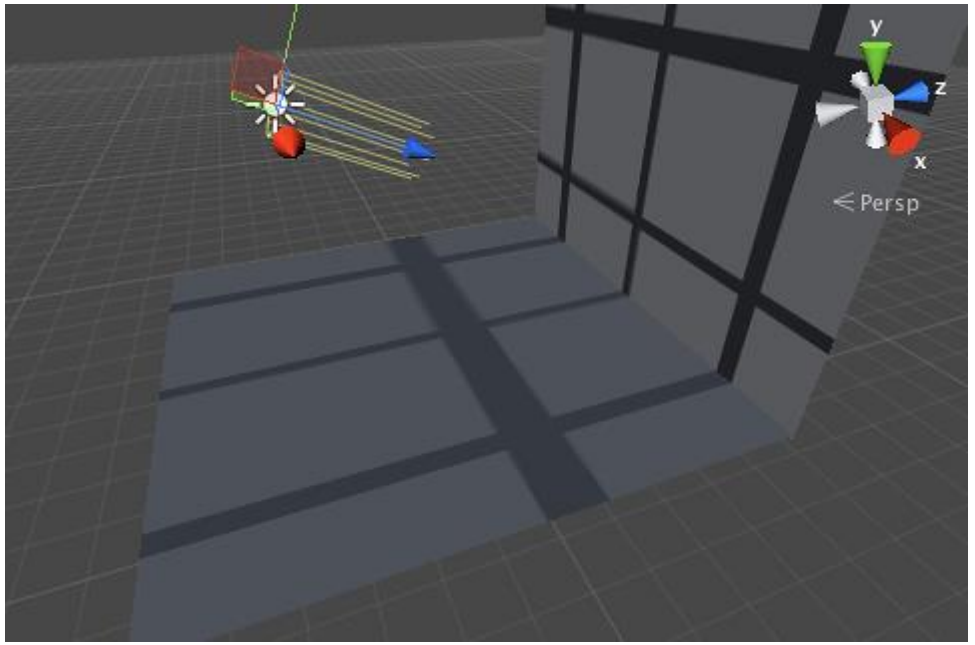


부드러운 외곽선

02. Directional Light / Cookie

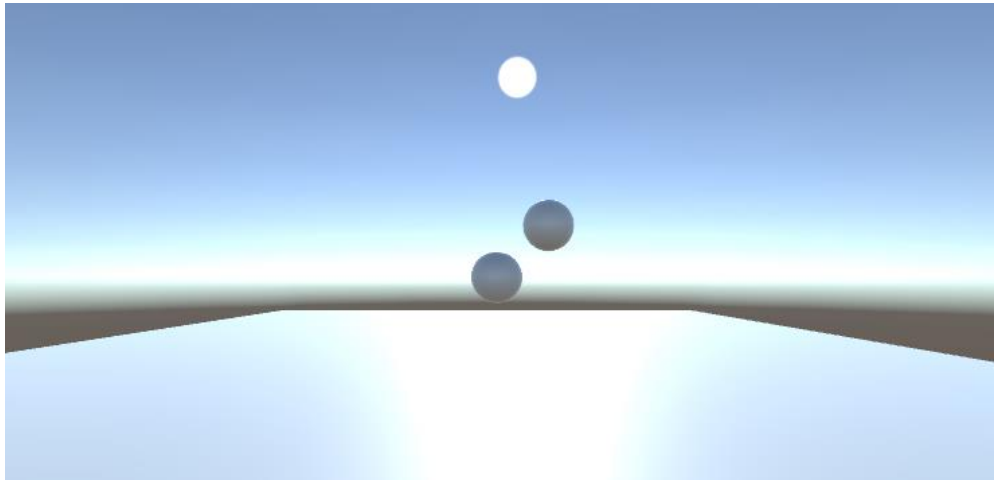


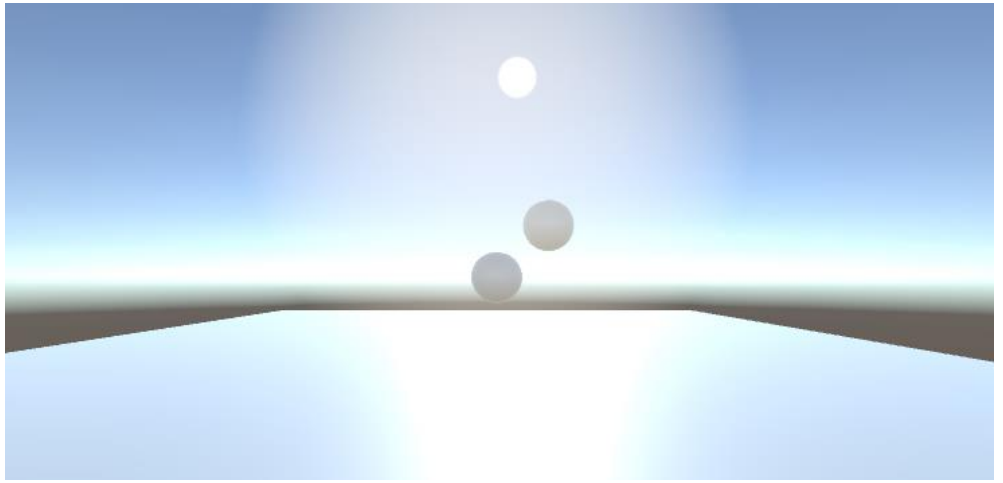




쿠키 : 그림자에 대한 조명 효과

02. Directional Light / Halo (후광)





Shadow Type	No Shadows
Cookie	None (Texture)
Cookie Size	10
Draw Halo	☐
Flare	None (Flare)
Render Mode	Auto
Culling Mask	Everything

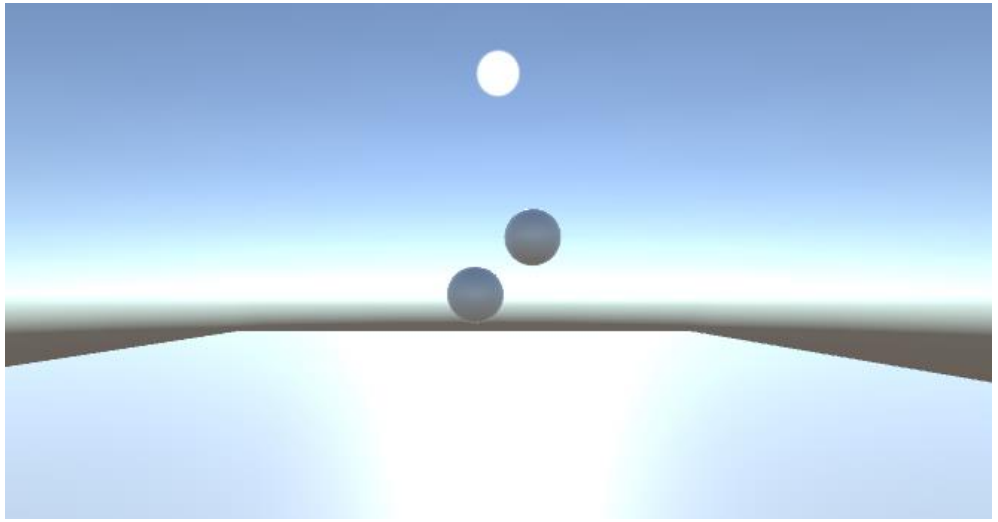
Shadow Type	No Shadows
Cookie	None (Texture)
Cookie Size	10
Draw Halo	☑
Flare	None (Flare)
Render Mode	Auto
Culling Mask	Everything

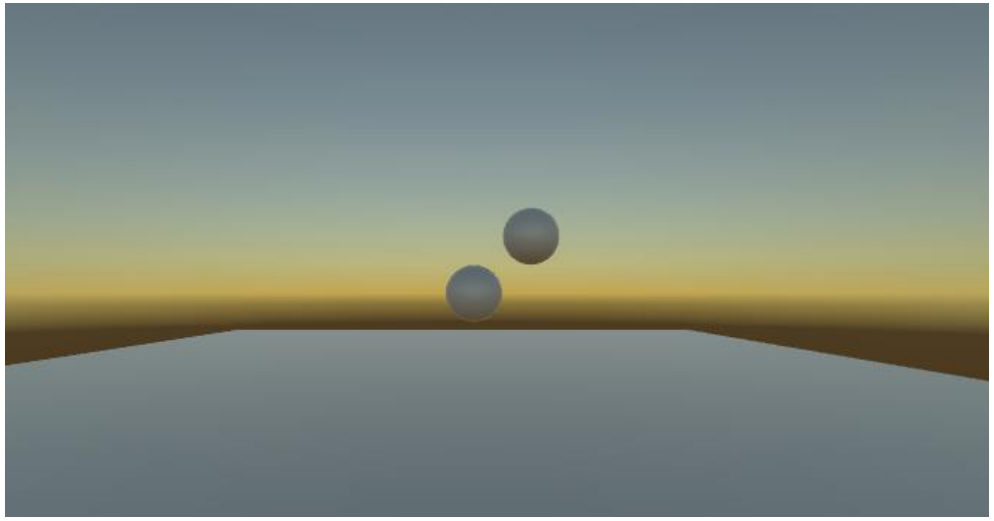
02. Directional Light / Lens Flare

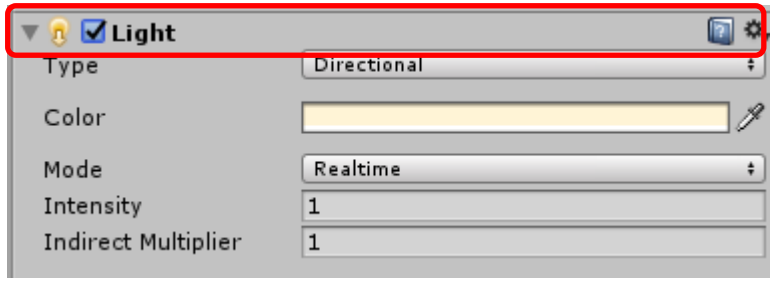
입사광이 렌즈 표면과 내벽에 반사, 산란되어 생기는 빛을 묘사
Asset > Create > Lens Flare

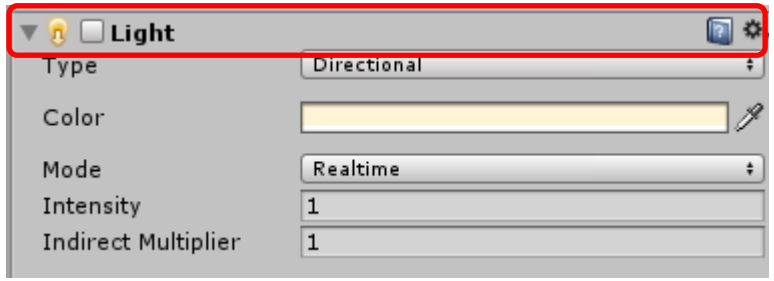
02. Directional Light / on and off

불을 꺼도 보인다.
광원이 어디에 더 있을까?

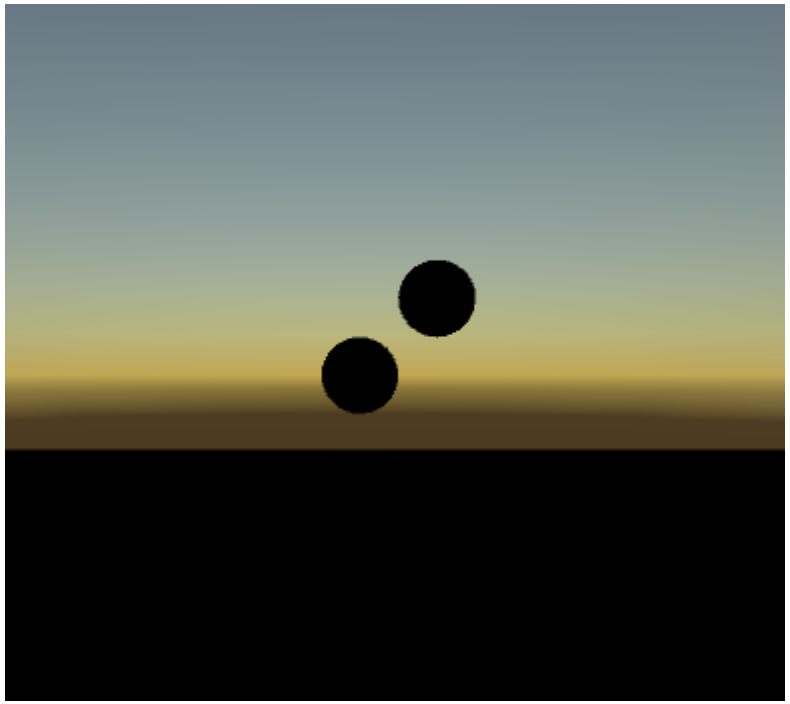


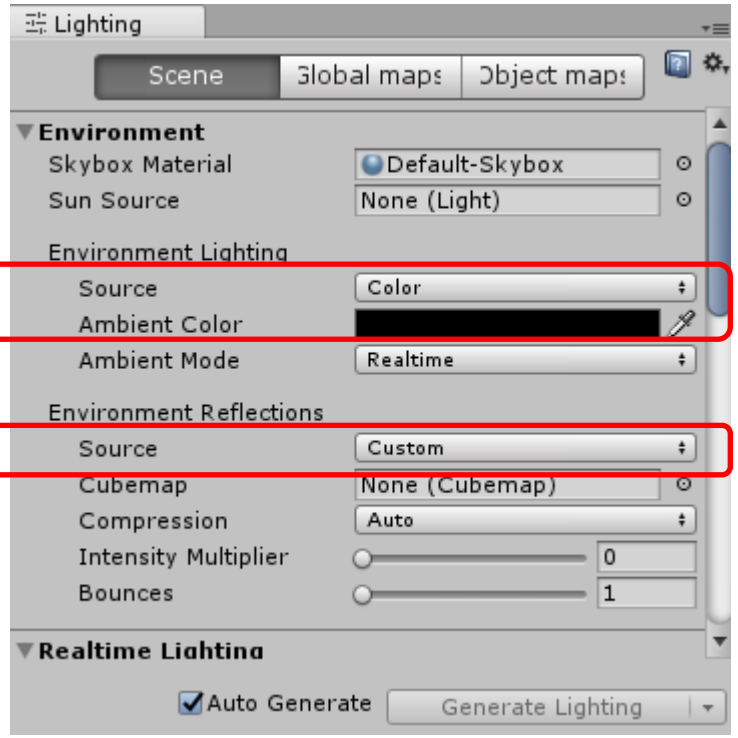






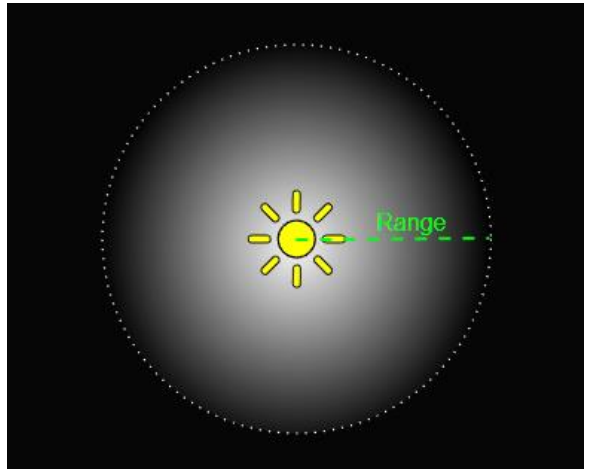
02. Directional Light / 다 꺼보자

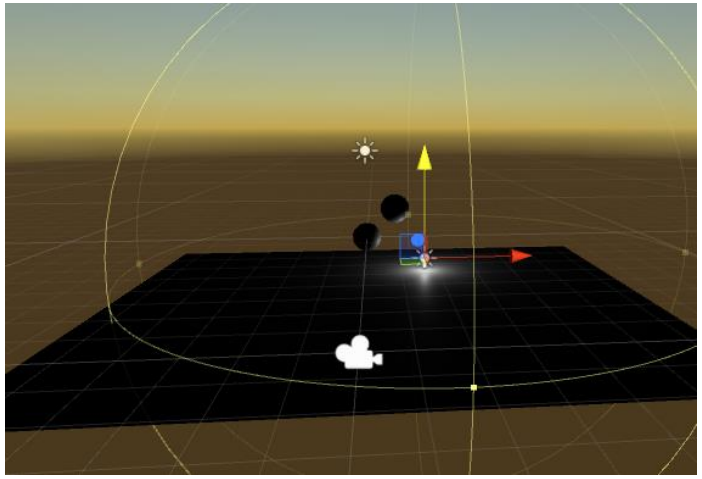


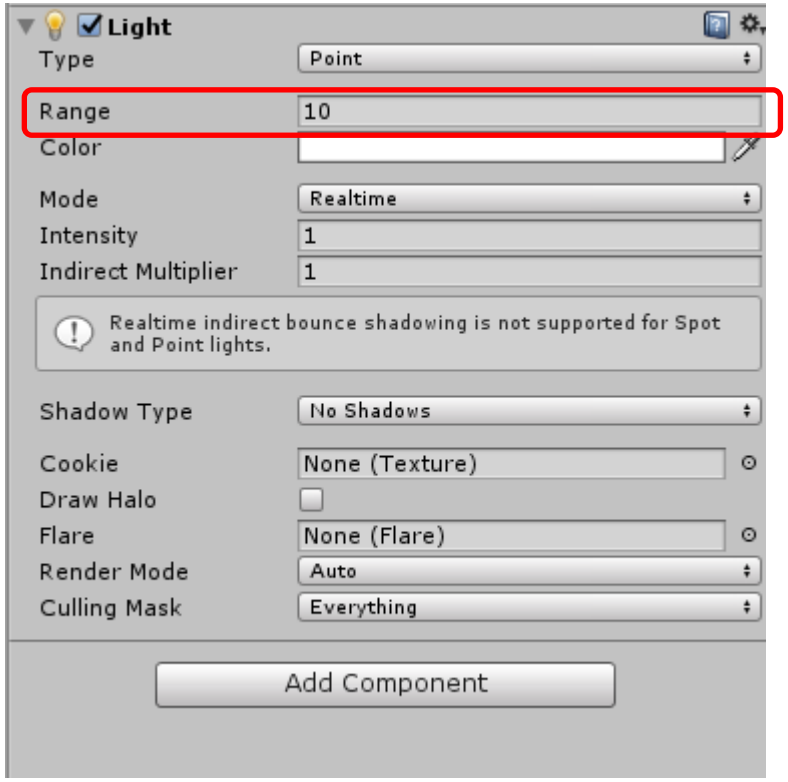


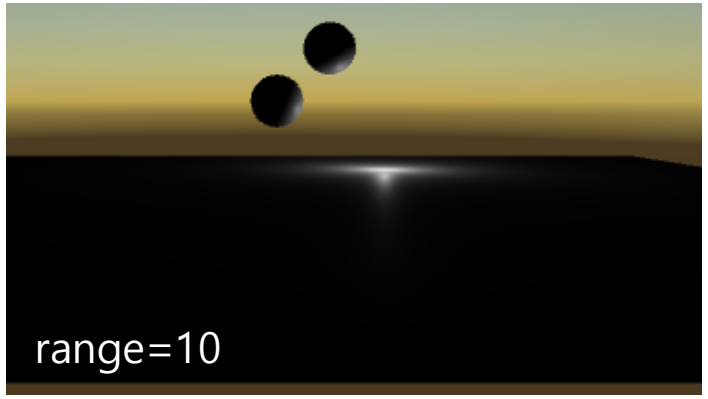
Point Light

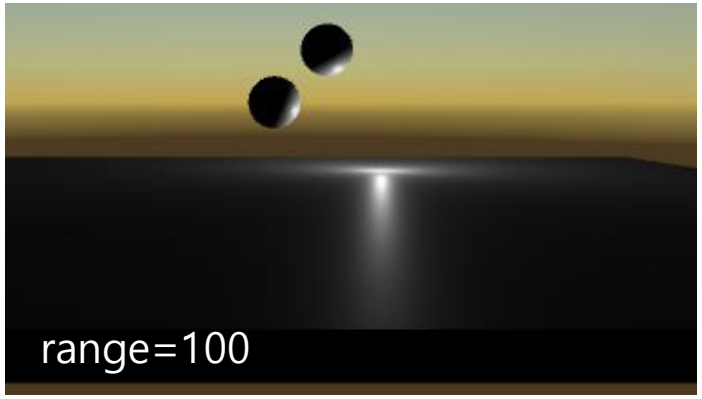
공간상의 한 점에서 빛을 발산
광원에서 멀어질 수록 강도가 약해짐



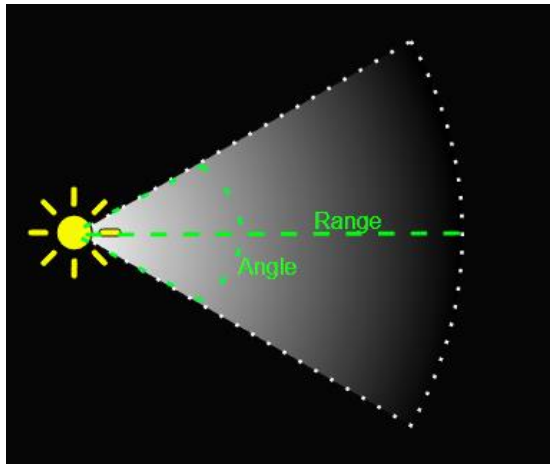




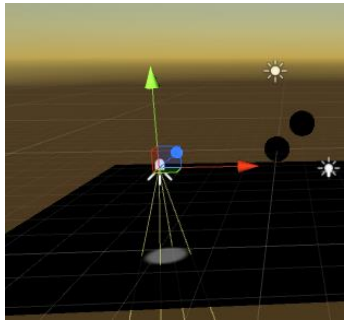


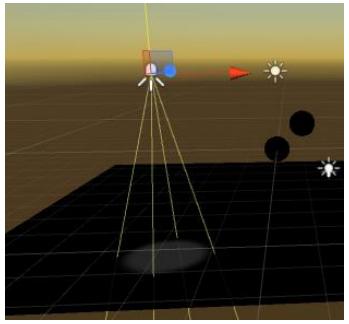


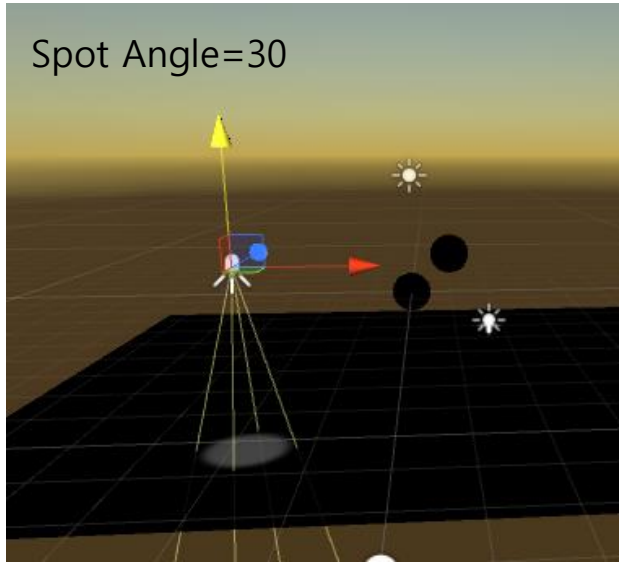
Spot light

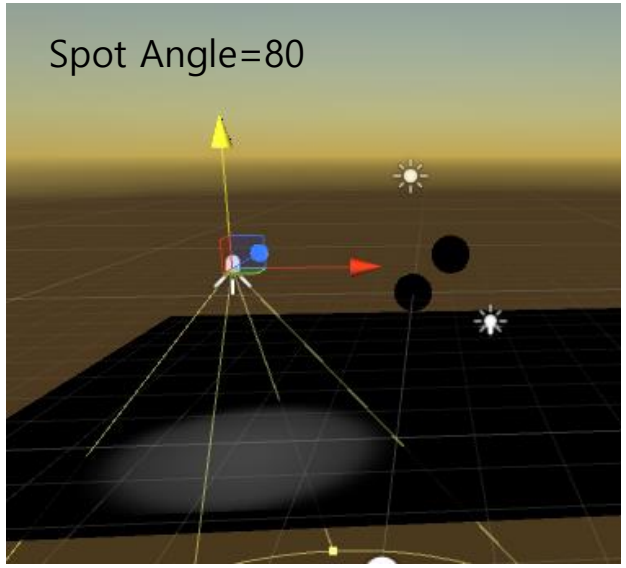


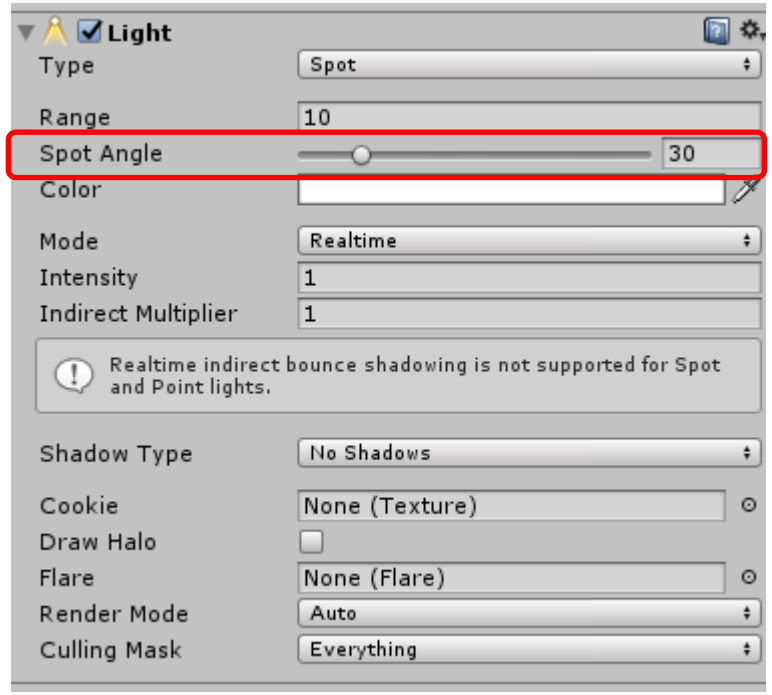
공간상의 한 점에서 특정 방향으로 빛을 발산
광원에서 멀어질 수록 강도가 약해짐





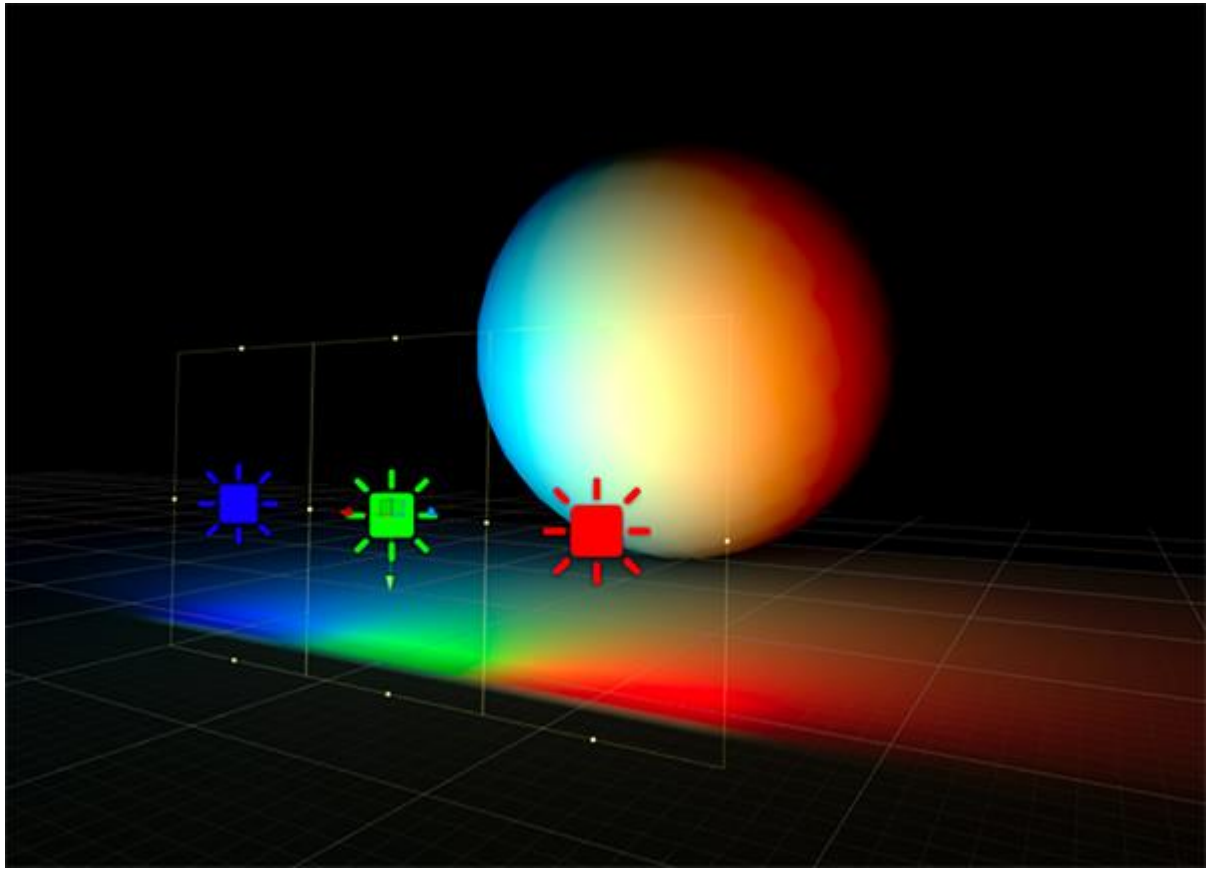






03. Area Light (Pro only)

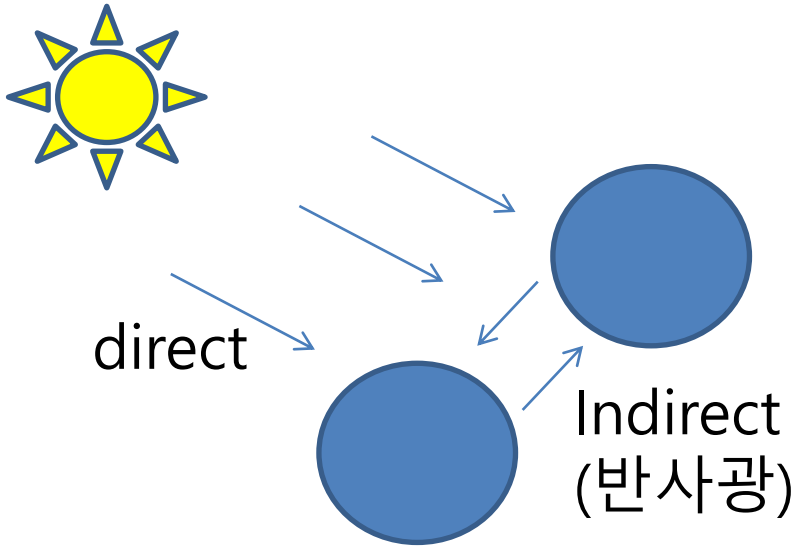
공간상의 직사각형에서 빛을 발산
광원에서 멀어질 수록 강도가 약해짐

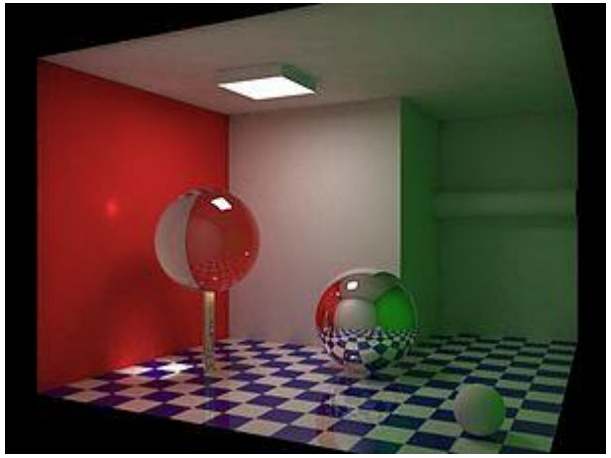


04. Global Illumination

GameObject가 반사하는 빛을 계산하여 Scene에 적용
보다 사실적인 장면을 연출

05. Lighting overview





Realtime Lighting (default): Directional, Spot, Point
- 매 프레임마다 update
- 단독으로 사용할 경우 반사광을 표현하지 못함
(즉 reality가 떨어짐)

Precompute Lighting : GI, 즉 전역 조명에서 사용

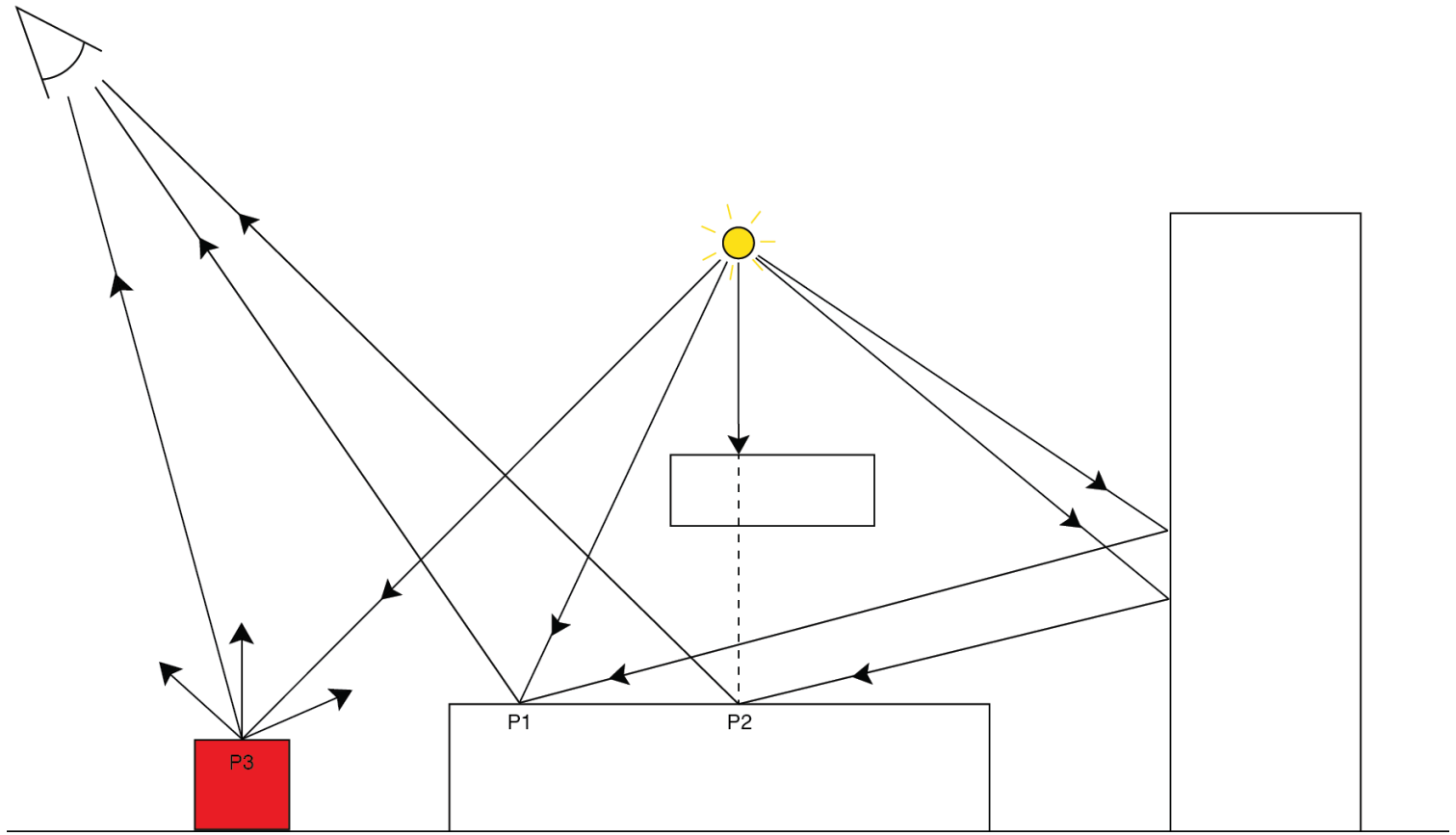
1) Baked Lightmap **GI** (GI는 전역조명)
lighting effect of Static Object → Lightmap에 저장
이 계산 과정을 baking 이라고 함
light map에는 direct / indirect 모두 저장됨
Game play 동안에는 light map이 변화되지 않음
realtime light는 이 light map위에 적용되나 no interaction
HW 성능을 적게 소모 (mobile platform에 적절)

2) Precomputed realtime **GI**
일부를 미리 계산 (반사되는 부분)
realtime light와 interaction을 위해 만들어진 방법

05. Lighting Overview

Realtime	조명효과를 매 frame마다 update
Baked	조명효과를 미리 계산 / 실시간에는 조명효과 계산 안함
Mixed	조명효과의 일부를 미리 계산

06. Lighting Effects



P1 - Direct + Indirect
P2 - Indirect (in shadow)
P3 - Emissive + Direct

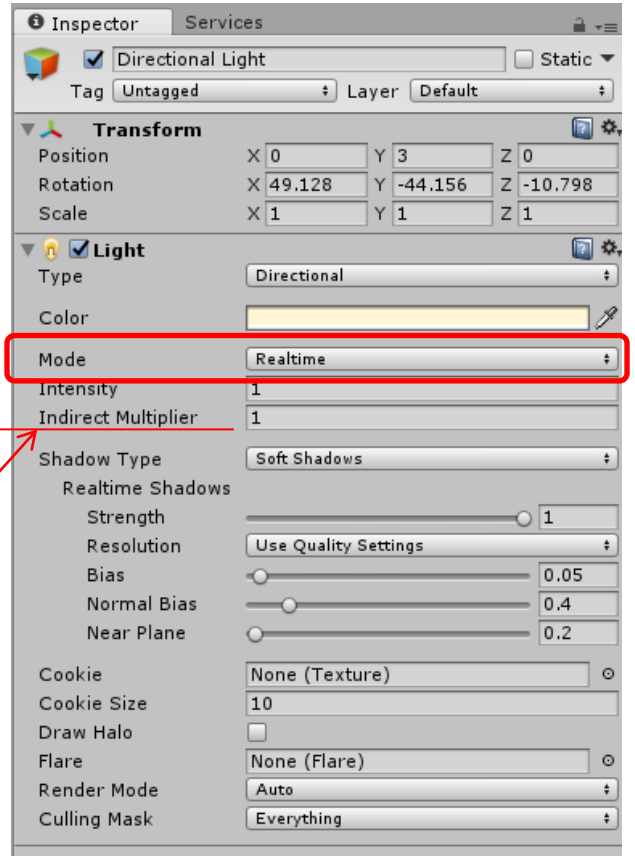
Major issues
1. Calculating shadow
2. Indirect lighting

Performance issues
1. Static & dynamic
2. Local & global
3. Realtime & non-realtime

07. Realtime Lighting

Realtime Lighting

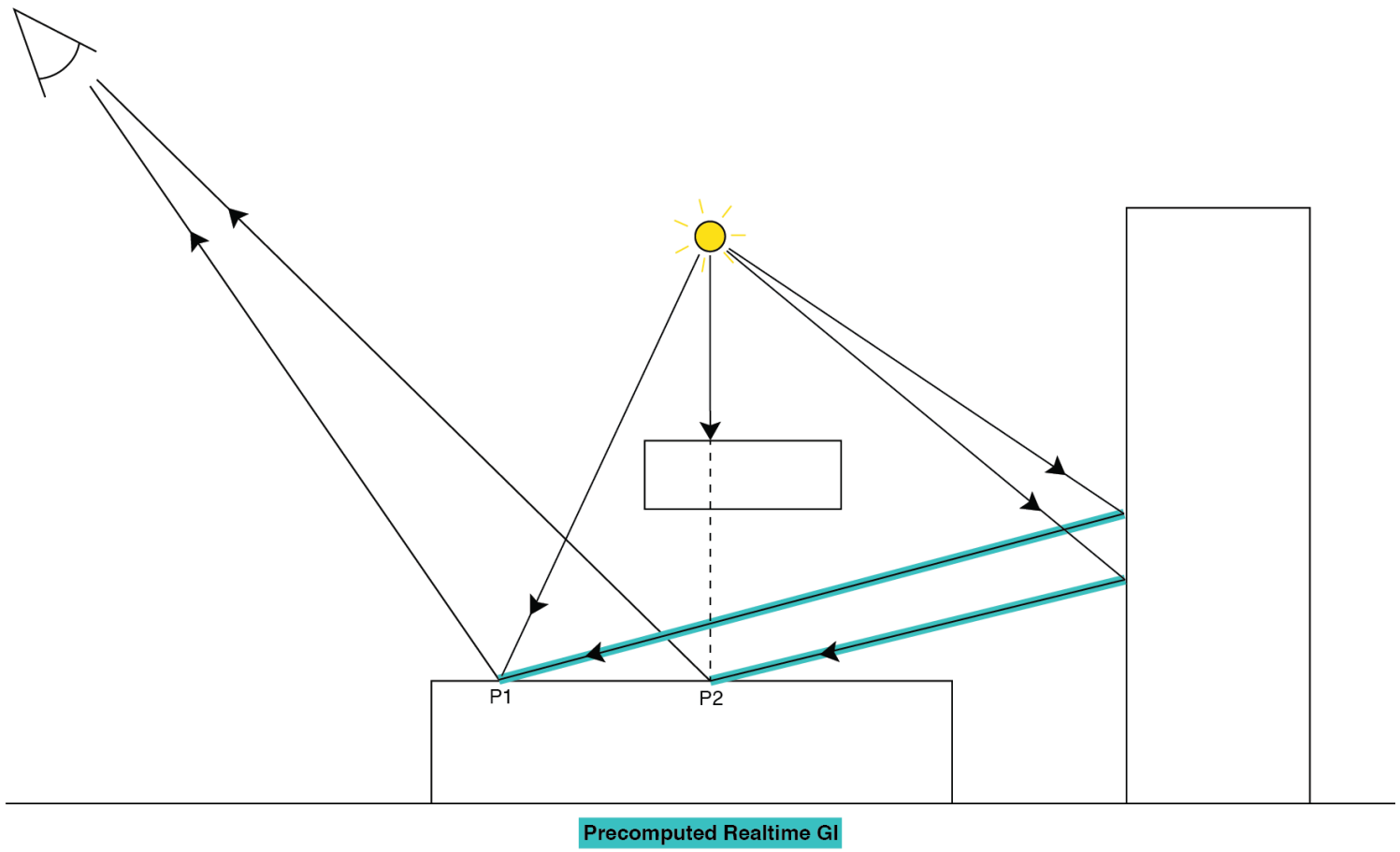
- 조명효과를 매 frame마다 update
- Mode property를 realtime으로 설정하면 적용됨
- Applicable realtime behavior
  - On/Off
  - transform
  - changing color and intensity
- Target : Both dynamic and static GameObject
- Realtime GI와 함께 사용하여 간접 조명도 제공



Realtime GI

- 아주 느리게 움직이는 조명
- Baked GI 대비 현저히 많은 리소스를 사용
- 고사양 PC 혹은 PS4, Xbox One 등에 적합
- 특정 광원에 Realtime GI를 비활성화하려면 이것을 0로 설정

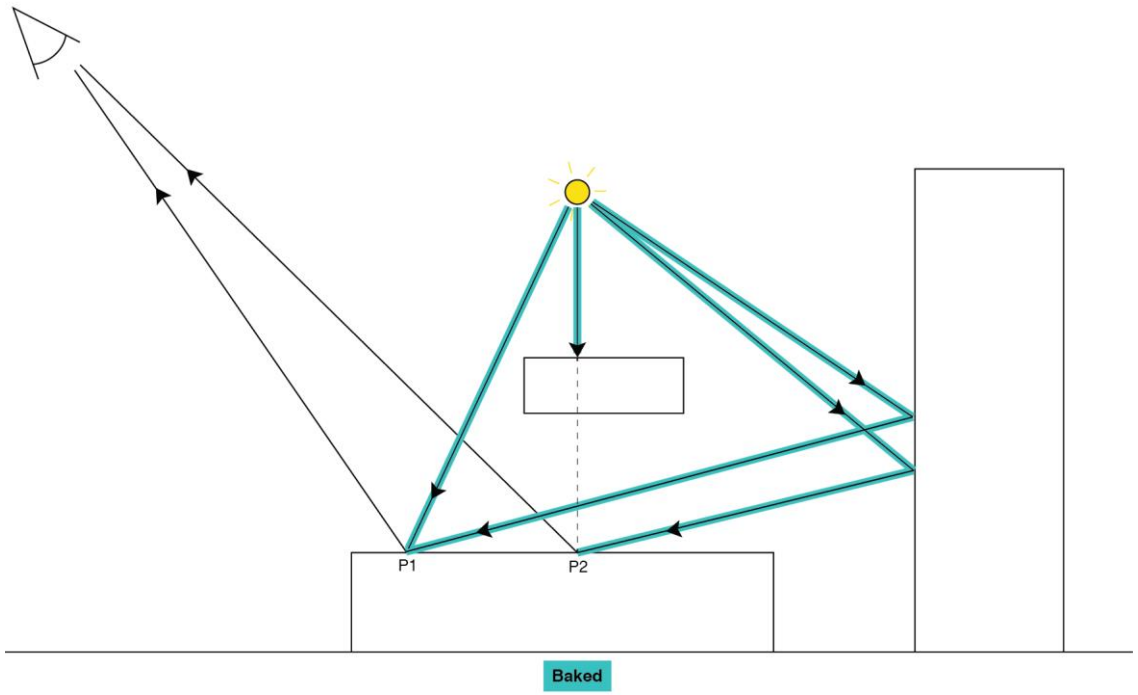
07. Realtime Lighting / Realtime GI

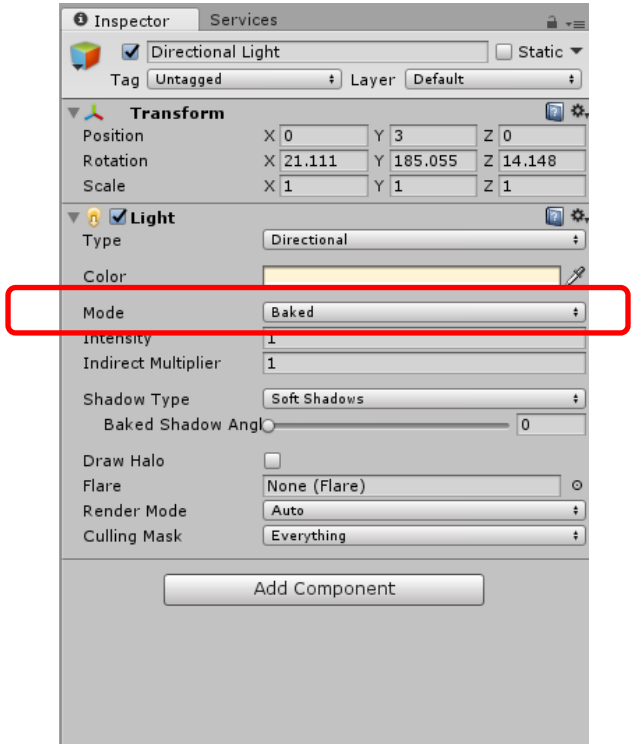


08. Baked Lighting

Baked Lighting

- 조명효과를 미리 계산하여 Lightmap에 저장
- 실시간 update는 안되므로 런타임 오버헤드가 훨씬 적다고 할 수 있음
- Mode property를 baked으로 설정하면 적용됨
- Target : Static GameObject (Dynamic GameObject를 위해 Light Probe 사용 / 이후 언급)
- Baked light는 dynamic GameObject에 빛을 비출 수가 없음
- 직접 및 간접 조명 모두 baking





09. Mixed Lighting

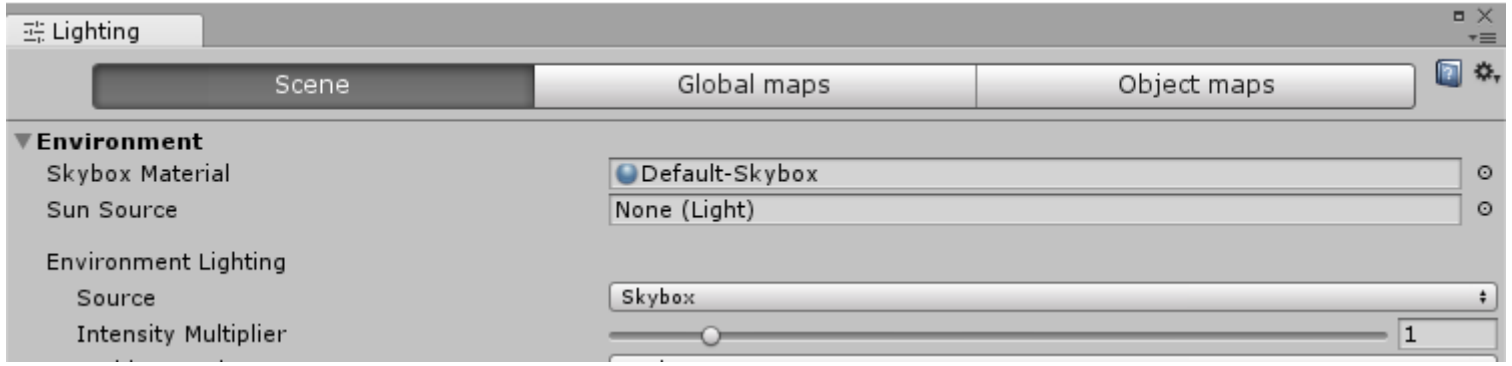
Mixed Lighting

- Realtime과 Baked 혼합 (즉 일부만 baking)
- Mode property를 mixed으로 설정하면 적용됨
- 조명의 color, intensity 등을 변경할 수 있으나 제약 사항이 따름
- Target : Static GameObject (Dynamic GameObject를 위해 Light Probe 사용 / 이후 언급)
- Baked light는 dynamic GameObject에 빛을 비출 수가 없음
- 직접 및 간접 조명 모두 baking

10. Lighting / Lighting Window

- Lighting Window
  - Main control point for Unity GI features
  - Window > Lighting > Settings

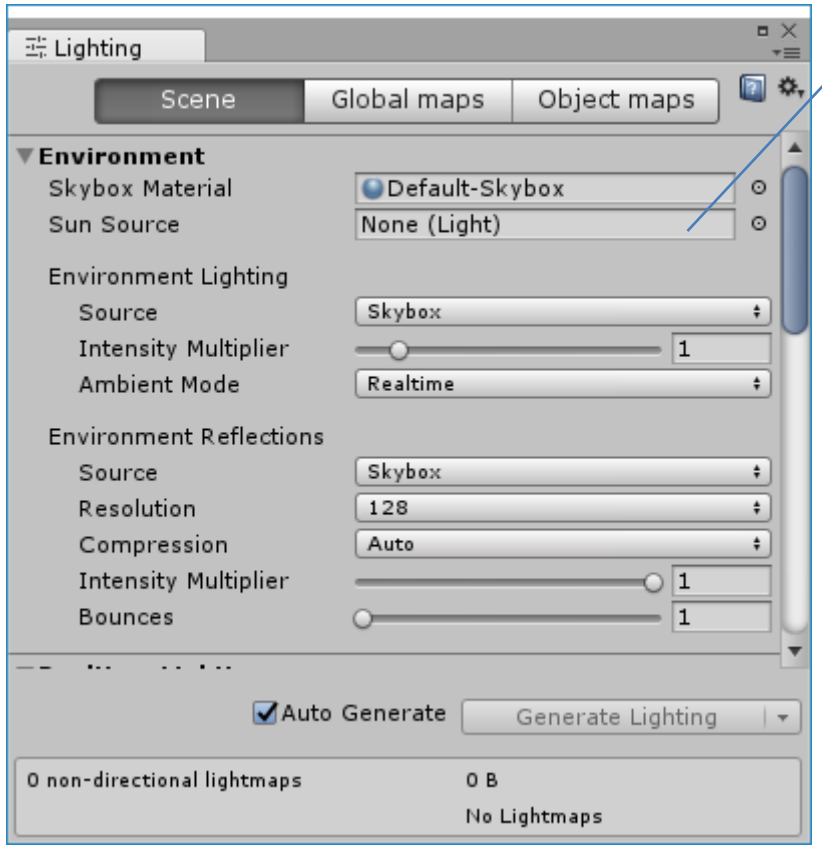
- Lighting Widow 개요
  - Scene Tab : Scene 전체에 적용되는 Settings, Light Effects & Optimization
  - Global maps : GI lighting 과정에서 생성된 lightmap asset file을 보여줌
  - Object maps : GI lightmap texture의 preview를 보여줌

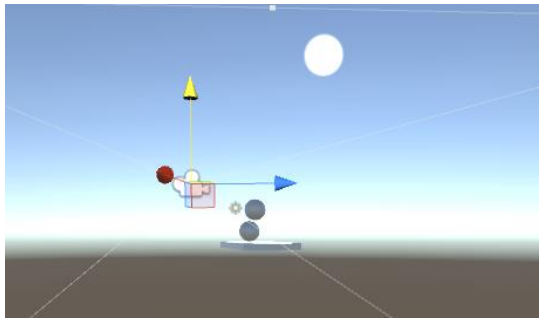


10. Lighting Window / Scene tab

- Scene tab 구성 요소 : Environment, Realtime Lighting, Mixed Lighting, Lightmapping Settings, Debug Settings

- Environment
  - Skybox, diffuse lighting and reflection



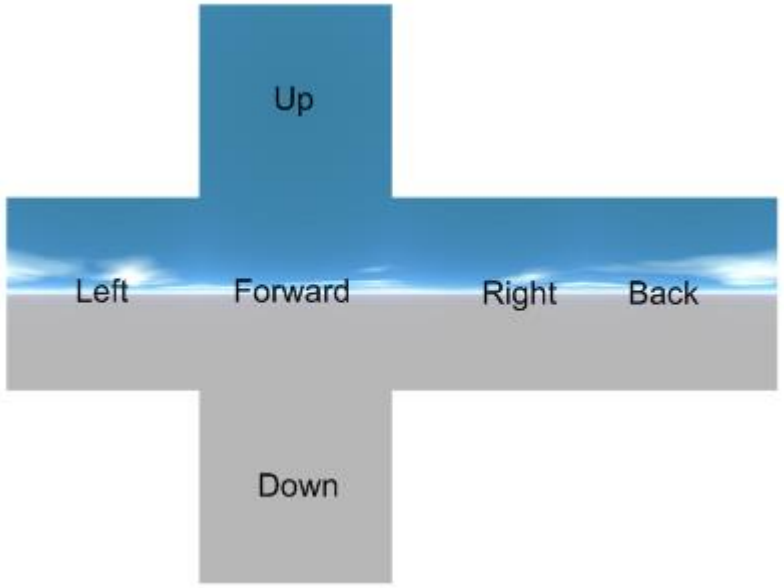


Environment Lighting 주변조명
Diffuse(확산되는) or Ambient(주위의) Light : 먼거리에서 적용되는 빛
Source : Color, Gradient, Skybox

Environment Reflection 주변반사 Reflection Probe (반사 프로브) Baking
Reflection Probe : 반사되는 빛을 저장

Source : Skybox, Custom
Compression : reflection texture 압축여부

참고 : Skybox



Skybox는 카메라가 볼 수 있는 하늘의 여섯 방면(front, back, left, right, up, down)의 이미지를 Cube 형태로 배치해서 표현.

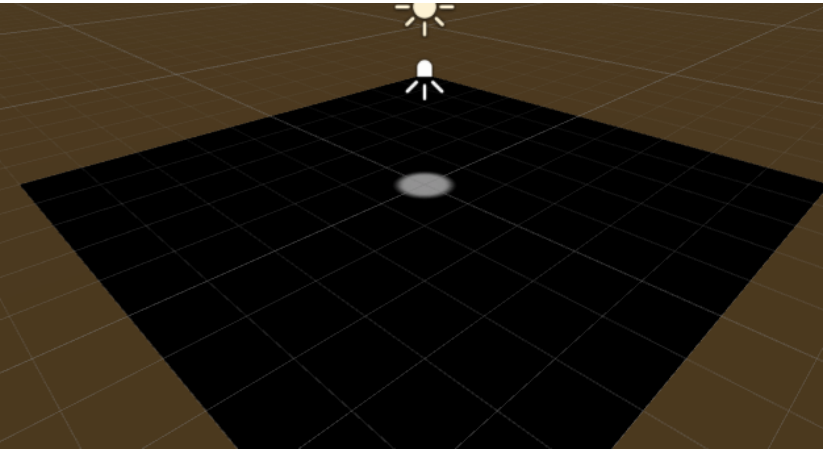
유니티 5에서 3가지 형태로 확장.

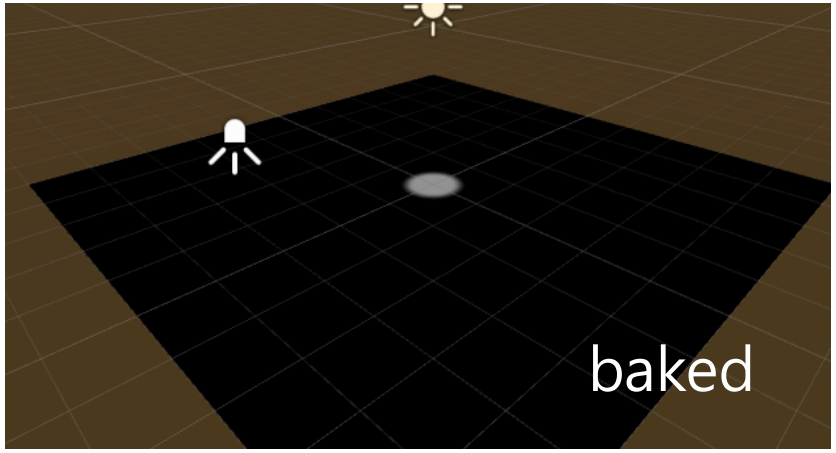
1. 6 sided
2. Cubemap
3. Procedural : 하늘의 색상, 대기 농도, 노출(밝기), 태양의 위치와 크기 등을 설정가능 (Default)

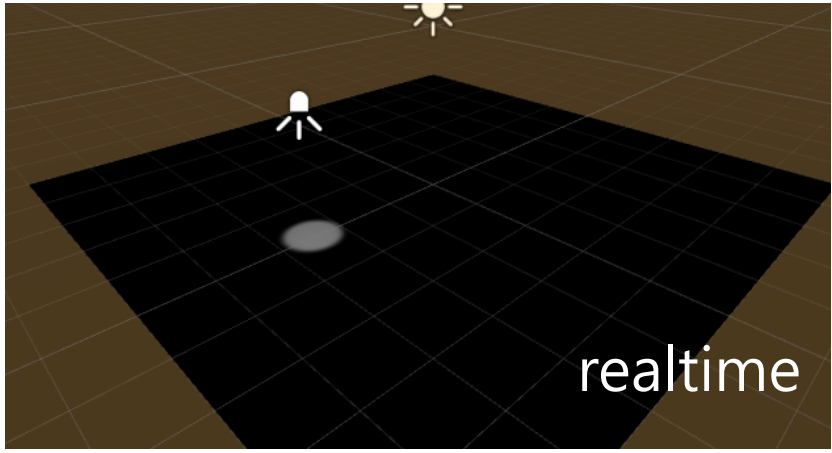
QUESTION

Baked Light를 사용하여 Runtime에 조명 문제가 발생하는 상황을 임의로 만들어보세요

1. Disable Directional Light
2. Remove reflection effects on light window
3. Add plane (3D object)
4. Add spotlight
5. Check plane “Lightmap static” in light section
6. Change spotlight type to baked
7. Write script which can move object (vertcal and horizontal)
8. Add the script (in step 7) to spotlight as a component
9. Play and move the light



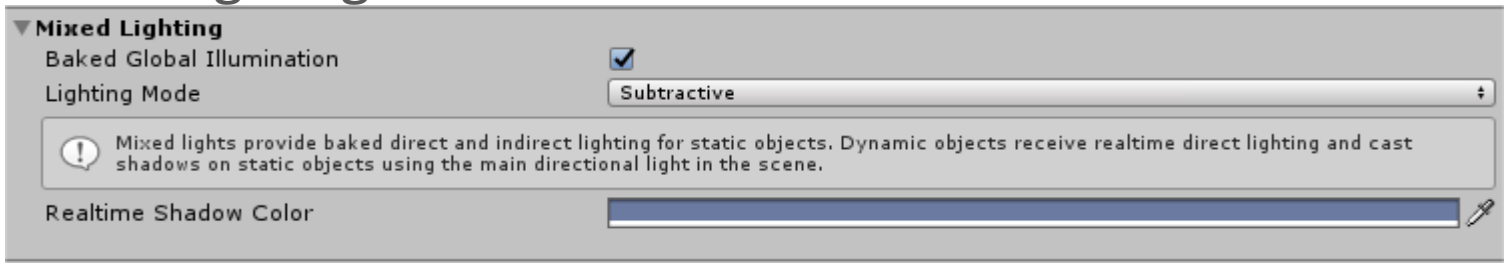




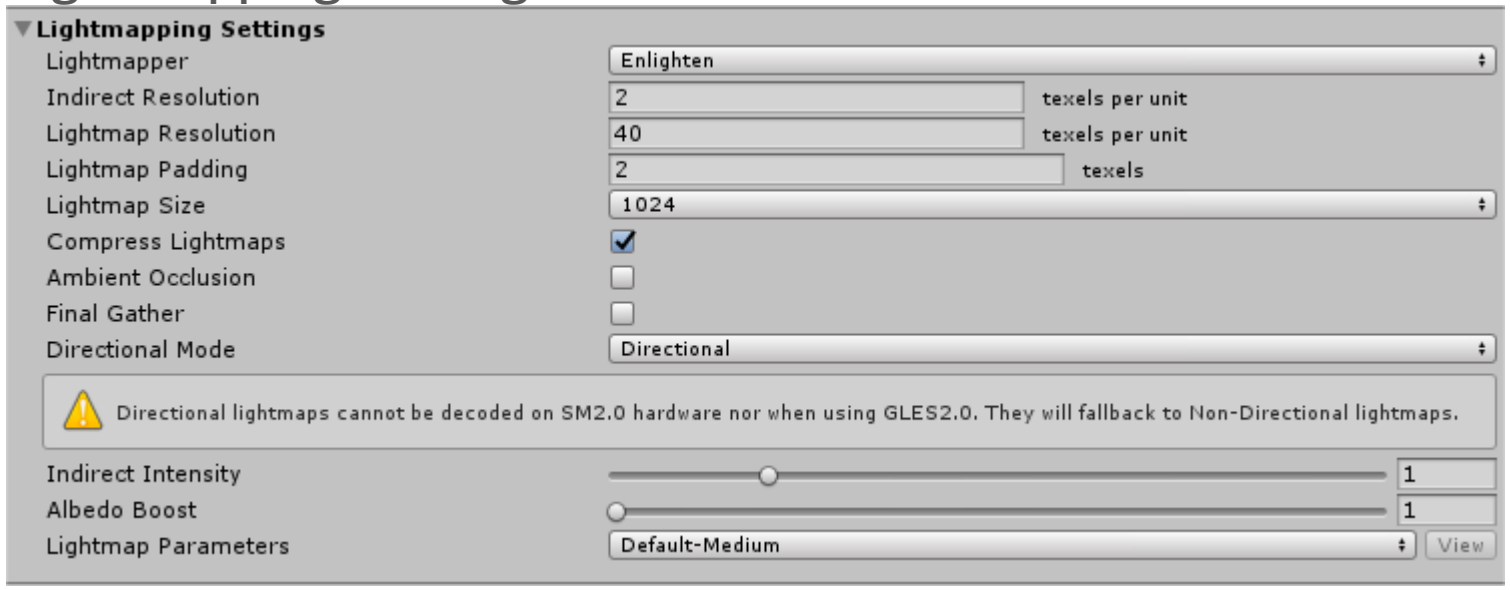
11. Lighting Window / Scene tab

- Realtime Lighting

- Mixed Lighting



- Lightmapping Setting



MATERIAL, SHADER, TEXTURE

01. 개요

- Rendering : Material / Shader / Texture

- **Texture** GameObject의 표면에 그릴 비트맵 이미지
  - 표면의 기본 컬러 (알베도), 반사도 또는 거칠기 등을 지정
  - PSD, PNG, JPEG, GIF, TIFF, BMP, TGA 등 다양한 포맷 지원
  - 가로, 세로의 크기가 2n일 때 처리 속도가 가장 빠름
  - Texture를 import한 다음(project view에 import), 해당 모델로 drag & drop (이렇게 하면 자동으로 Materials 폴더가 생성됨)
  - 효율적인 관리를 위해서는 Material을 따로 모아 두는 것이 유리 (Create --> Material)
- **Material** Texture를 그리는 방법을 정의 (3D model과 Texture를 연결)
  - Texture reference(선택), tiling info, color tint (색조)
  - 즉, 어떤 texture를 어떤 간격으로 붙이고 어떤 재질로 표현하느냐를 설정
  - Shader 지정
- **Shader** 조명, Material 설정에 따라 표면의 각 픽셀 컬러를 계산하는 스크립트
  - Standard Shader : 케릭터, 풍경, 환경, 솔리드 및 투명, 딱딱하고 부드러운 표면 등
  - 액체, 나뭇잎, 굴절 유리, 파티클 효과 등의 기타 특수 효과를 위한 빌트인 shader가 있음

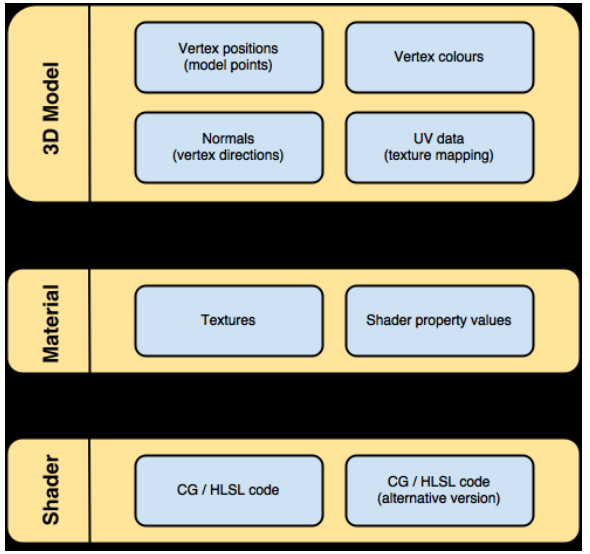
02. Texture

- Texture
  - GameObject의 표면 정보는 mesh geometry로 정의
  - 이 mesh geometry 상에 bitmap 이미지를 늘려 적절한 위치에 고정시킴

- 3D Model Rendering
  - 특정 texture를 object에 적용하는 방법은 Material을 통해 정의
  - Material은 Shader를 이용하여 표면에 rendering
  - Shader는 조명 및 컬러 효과를 구현하여 다양한 상황을 시뮬레이션

  - Texture는 2의 제곱수에 해당하는 크기로 만들어야 함
  - 순서 : Texture import --> Material 할당 --> mesh, particle, gui 등에 적용

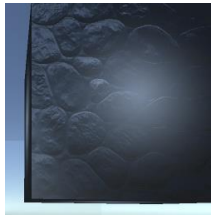
- 참고
  - 2D Graphics에서의 texture는 Sprite이며 평평한 mesh에 적용됨
  - GUI는 Standard Texture를 통해 coloring
  - 파티클은 작은 2D 그래픽임



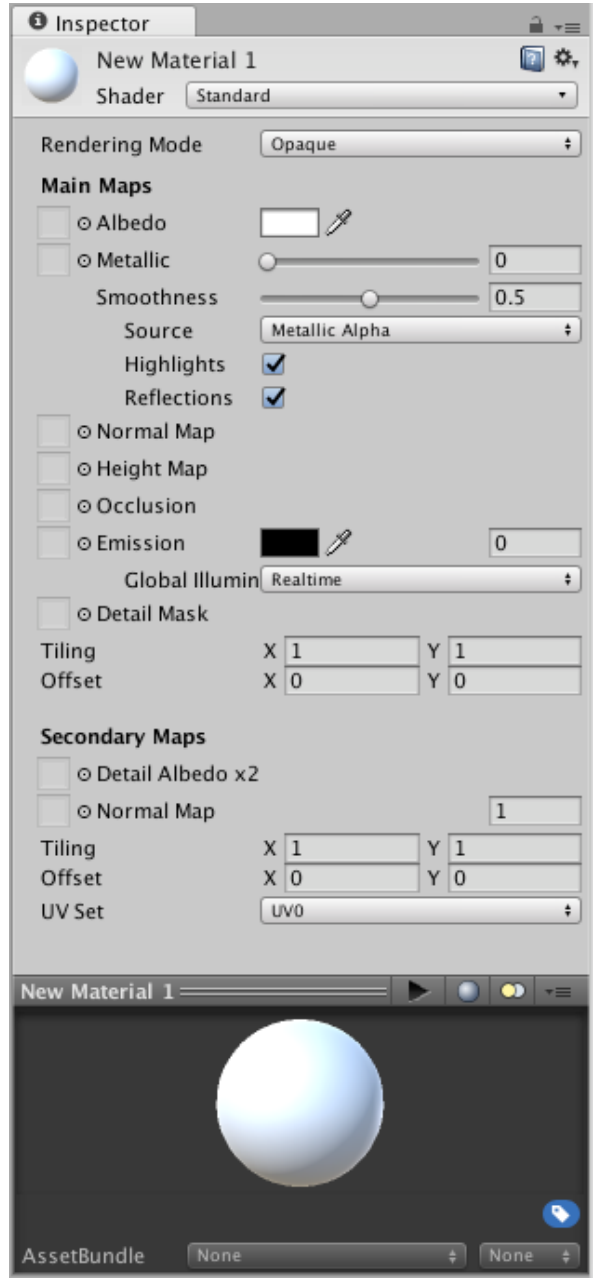
03. Material

- 새 Material 생성 :
  - Assets --> Create --> Material

- Material 설정



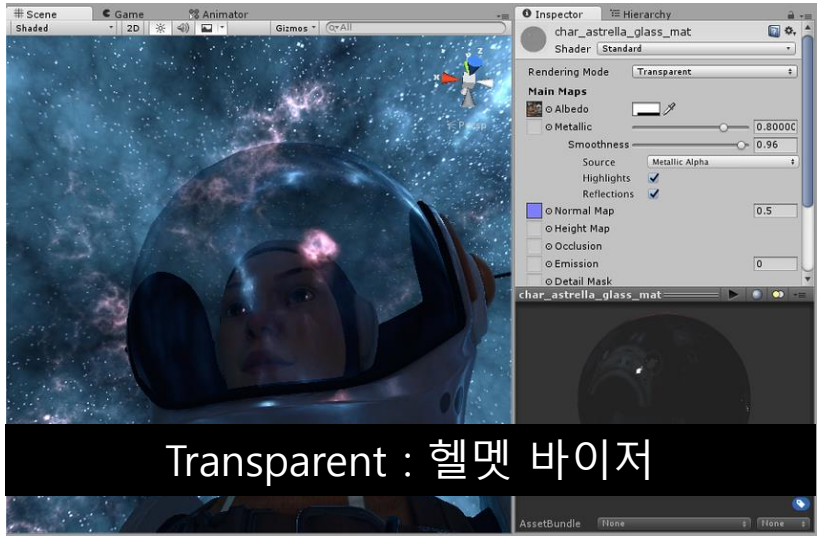
  - Shader 선택 (Default Shader : Standard Shader)
  - Rendering Mode 선택 : Opaque, Cutout, Transparent, Fade
  - Albedo Color and Transparency : 표면의 base color 제어
  - Metallic 금속 표면 처럼 보이게 설정
  - Normal Map 표면의 굴곡을 표현 (RGB → XYZ)
  - Height Map 보통 texture로 높낮이 표현 (Grayscale)
  - Occlusion Map 간접광이 미치는 범위 지정 (Grayscale)
  - Emission 자체발광



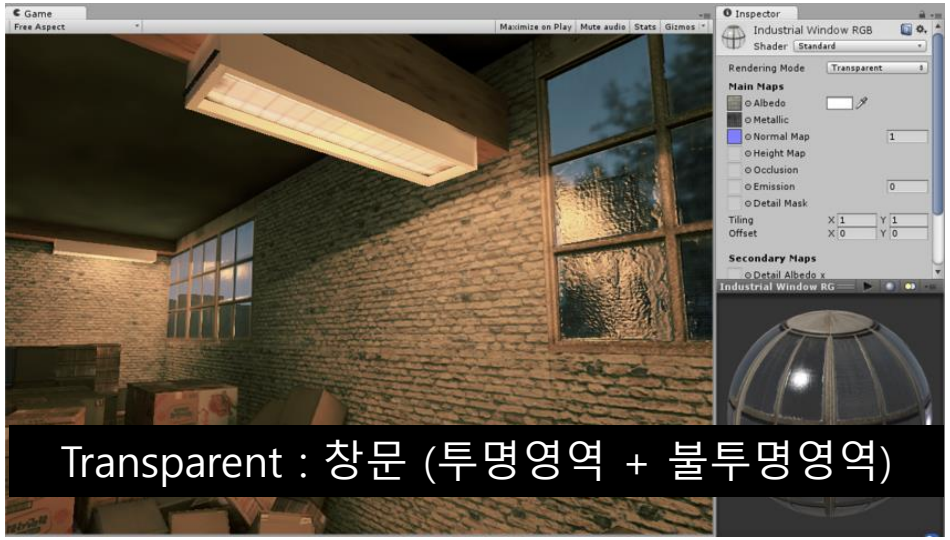
03. Material / Rendering Mode

Rendering Mode에서의 선택 사항

Opaque : 투명한 영역이 없는 Solid Object (default)

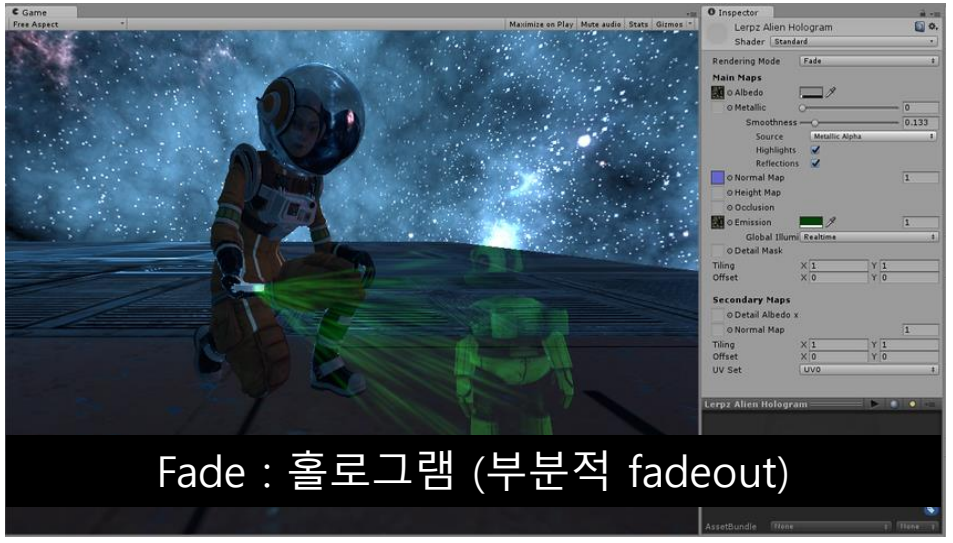


Transparent : 헬멧 바이저



Transparent : 창문 (투명영역 + 불투명영역)

Cutout:알파값의 임계치를 기준으로 투명, 불투명역역을 구분



Fade : 홀로그램 (부분적 fadeout)

03. Material / Rendering Mode (Test)

Texture 추가 (black:투명) → Material 생성 → Material에 texture 할당 → Object(Cube)에 Material할당

Inspector | Services

tiger-1395138_960_720 Import Settings

Open

Texture Type: Default
Texture Shape: 2D
sRGB (Color Texture): ☑
Alpha Source: Input Texture Alpha
Alpha Is Transparency: ☐
▶ Advanced
Wrap Mode: Repeat
Filter Mode: Bilinear
Aniso Level: 1

Default
Max Size: 2048
Compression: Normal Quality
Format: Auto
Use Crunch Compress: ☐

Revert | Apply

Inspector | Services

New Material
Shader: Standard

Rendering Mode: Opaque

Main Maps
Albedo
Metallic: 0
Smoothness: 0.5
Source: Metallic Alpha
Normal Map
Height Map
Occlusion
Detail Mask
Emission
Tiling: X 1 Y 1
Offset: X 0 Y 0

Secondary Maps
Detail Albedo x2
Normal Map: 1
Tiling: X 1 Y 1
Offset: X 0 Y 0
UV Set: UV0

Forward Rendering Options
Specular Highlights: ☑
Reflections: ☑

Advanced Options
Enable GPU Instancing: ☐
Double Sided Global Il: ☐

Inspector | Services

New Material
Shader: Standard

Rendering Mode: Opaque

Main Maps
Albedo
Metallic: 0
Smoothness: 0.5
Source: Metallic Alpha
Normal Map
Height Map
Occlusion
Detail Mask
Emission
Tiling: X 1 Y 1
Offset: X 0 Y 0

Secondary Maps
Detail Albedo x2
Normal Map: 1
Tiling: X 1 Y 1
Offset: X 0 Y 0
UV Set: UV0

Forward Rendering Options
Specular Highlights: ☑
Reflections: ☑

Advanced Options
Enable GPU Instancing: ☐
Double Sided Global Il: ☐

Inspector | Services

Cube — Static
Tag: Untagged — Layer: Default

Transform
Position: X 1.94 Y 1.77 Z 1.93
Rotation: X 0 Y 0 Z 0
Scale: X 1 Y 1 Z 1

Cube (Mesh Filter)
Mesh: Cube

Box Collider
Edit Collider
Is Trigger: ☐
Material: None (Physic Material)
Center: X 0 Y 0 Z 0
Size: X 1 Y 1 Z 1

Mesh Renderer
Lighting
Light Probes: Blend Probes
Reflection Probes: Blend Probes
Anchor Override: None (Transform)
Cast Shadows: On
Receive Shadows: ☑
Motion Vectors: Per Object Motion
Lightmap Static: ☐

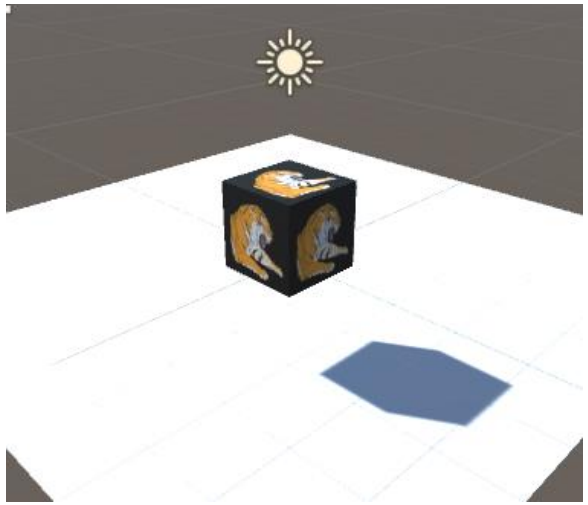
To enable generation of lightmaps for this Mesh Renderer, please enable the 'Lightmap Static' property.

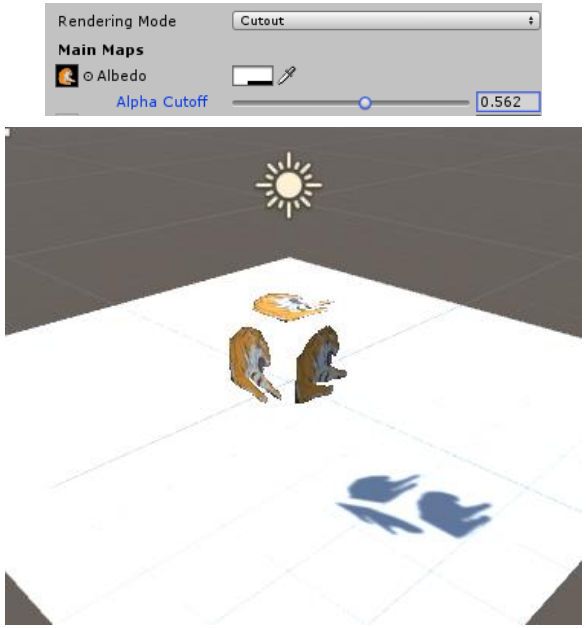
Materials
Size: 1
Element 0: New Material

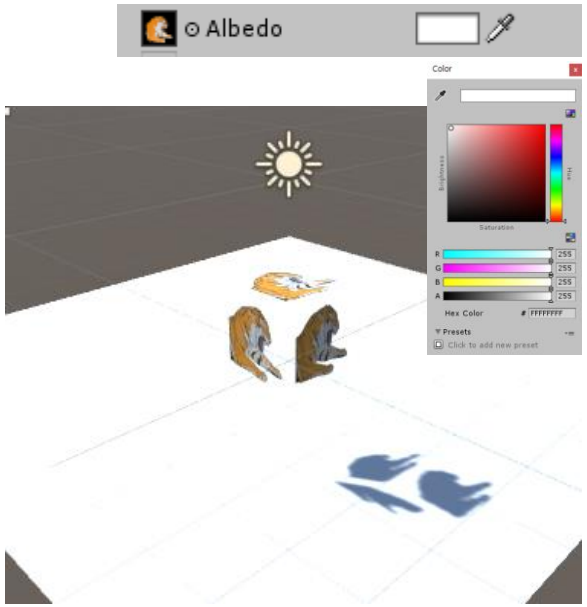
New Material
Shader: Standard

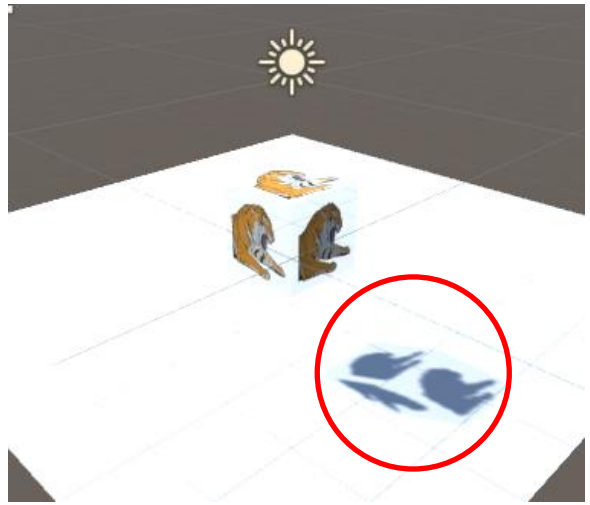
Add Component

03. Material / Rendering Mode (Test)









Opaque

Texture의 alpha값 반영안됨

Cutout

Alpha값을 변화시키면 펑 사라짐
(Cutoff 값을 기준으로)

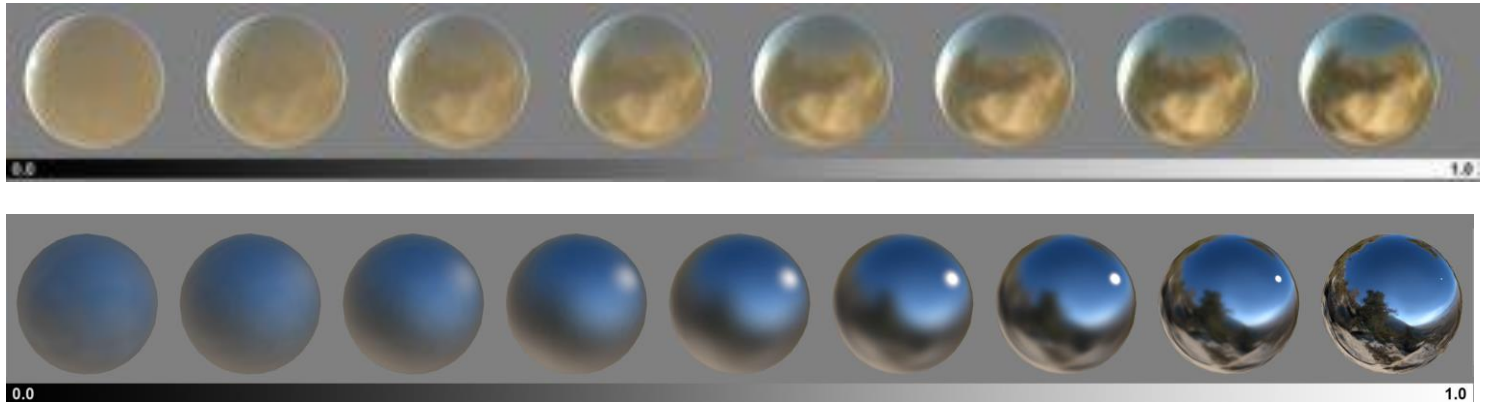
Fade

Texture의 alpha값은 반영됨
Alpha값을 변화시키면 서서히 사라짐

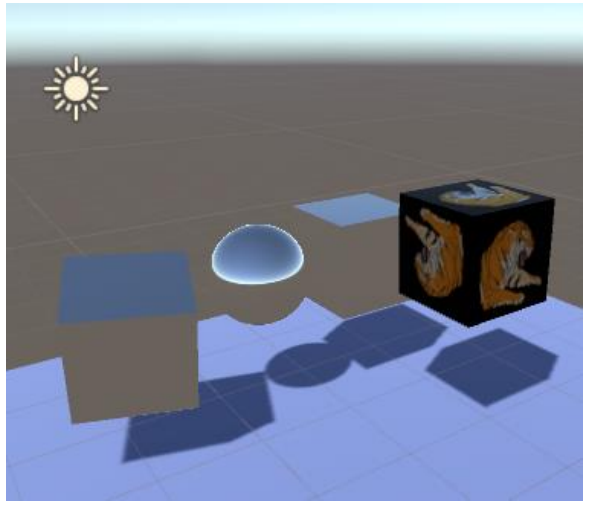
Transparent

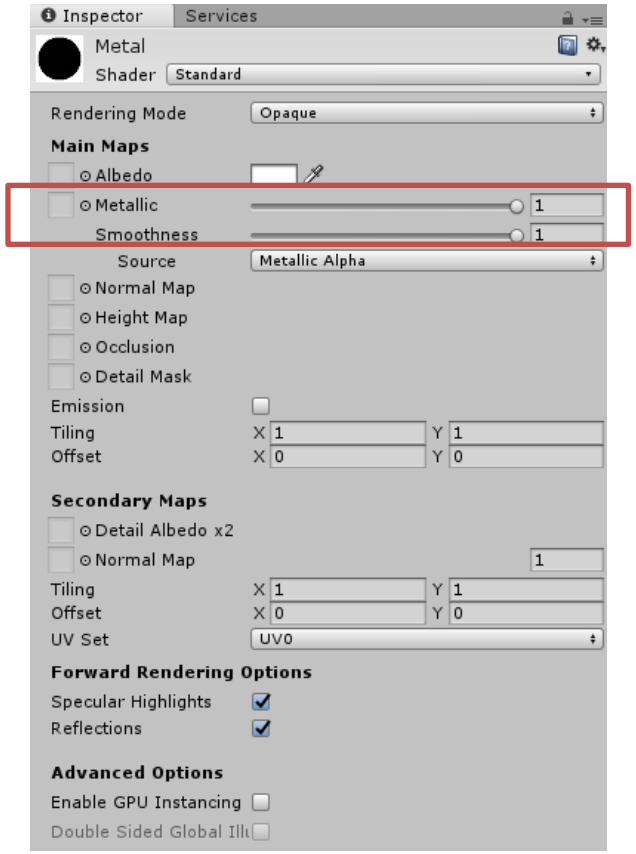
투명한 질감이 살아 있음
(모서리 부분 등에서 이질감)

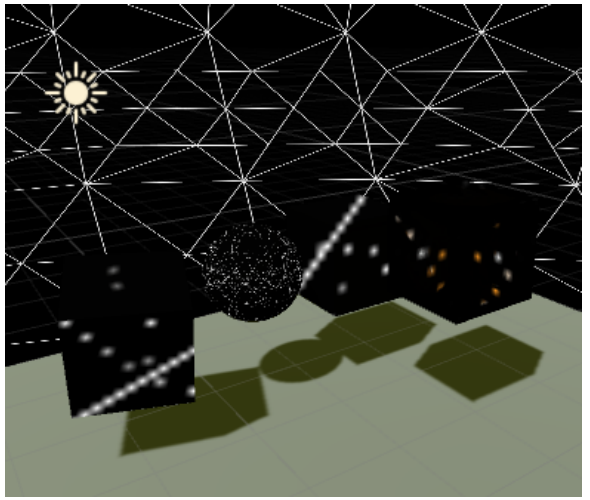
03. Material / Rendering Mode (Test)



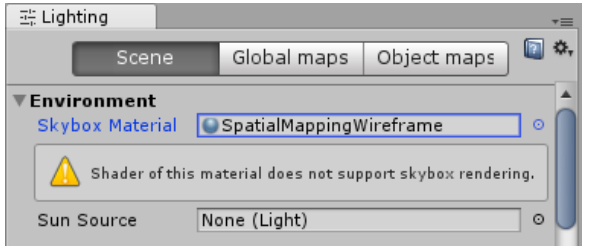
03. Material / Metallic



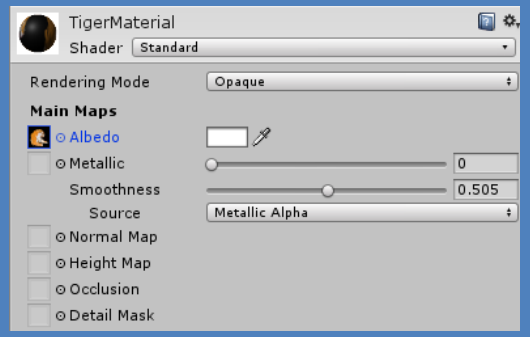


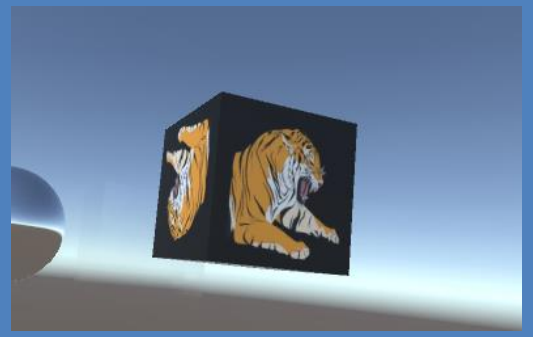


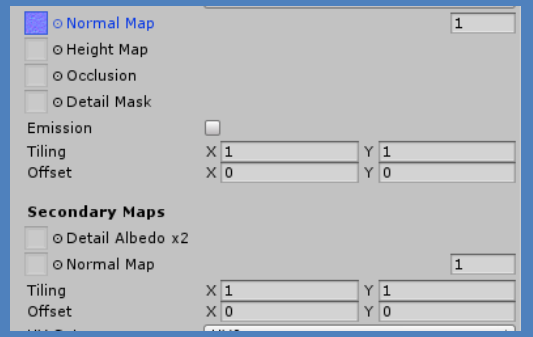
Other skybox



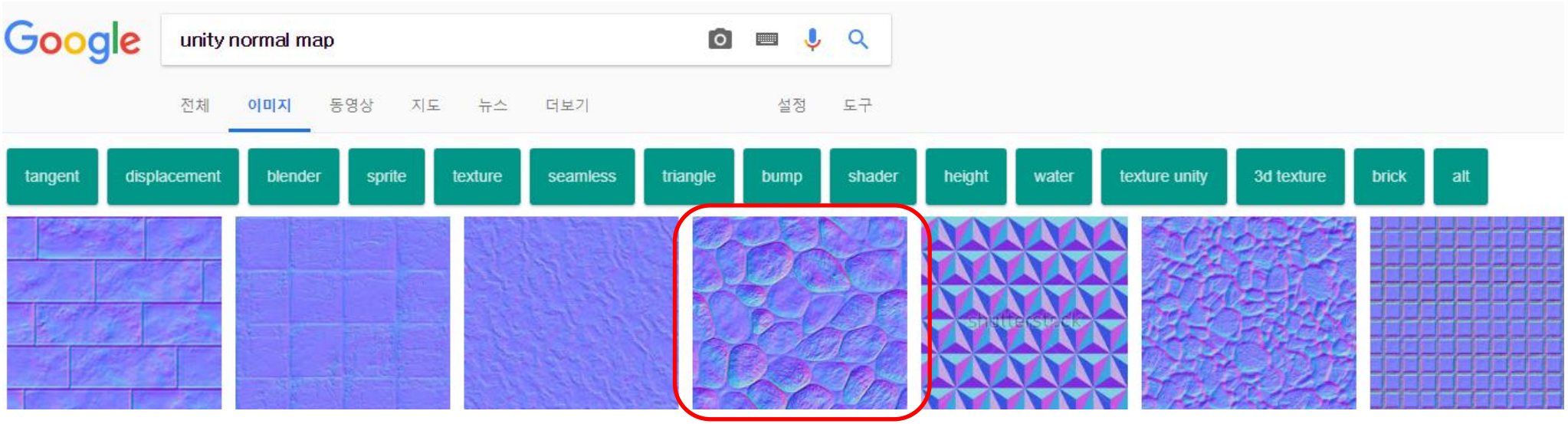
03. Material / Normal Map





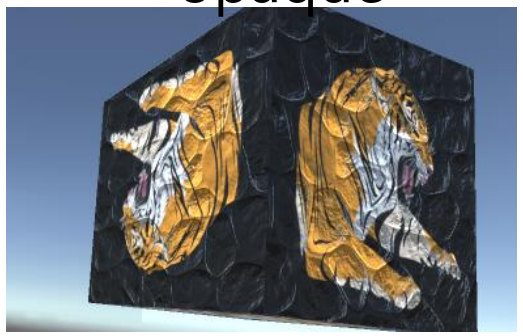


Texture의 거친 표면을 표현하기 위하여 …
Normal Map 텍셀(Texture pixel)의 RGB 값이 방향 벡터의 X, Y, Z 값을 나타냄

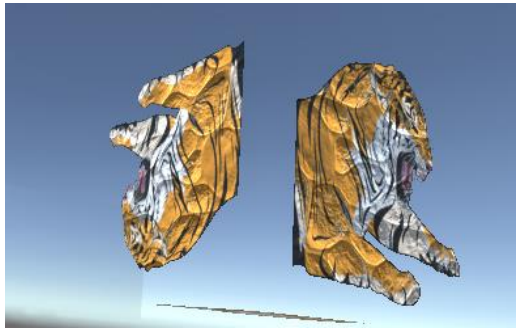


03. Material / Normal Map

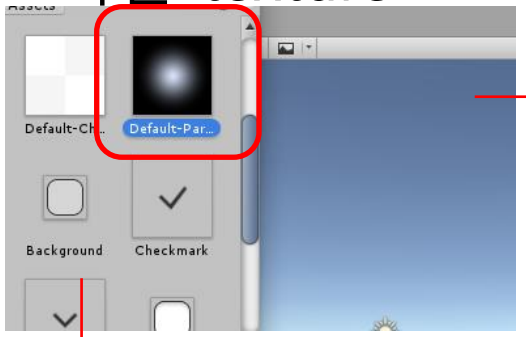
opaque

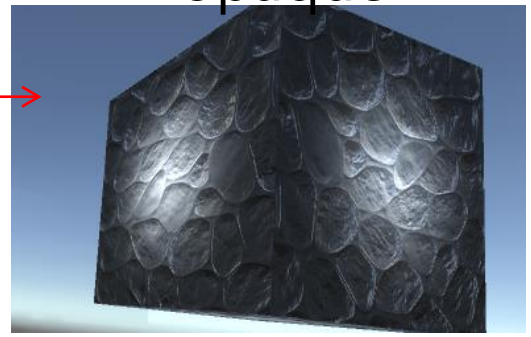


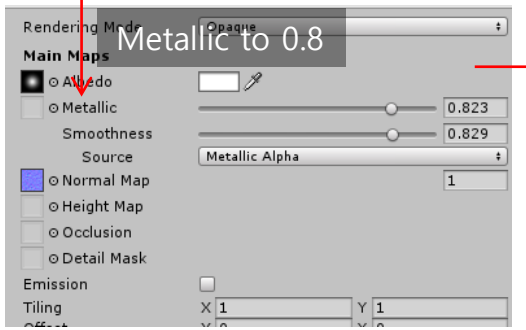
cutout



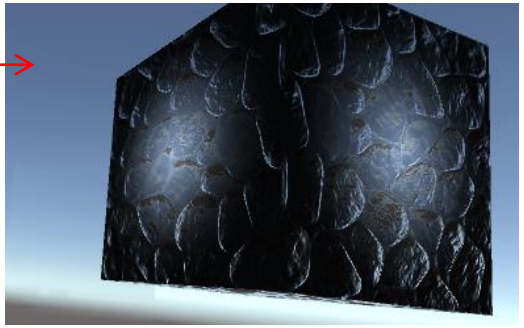
다른 texture



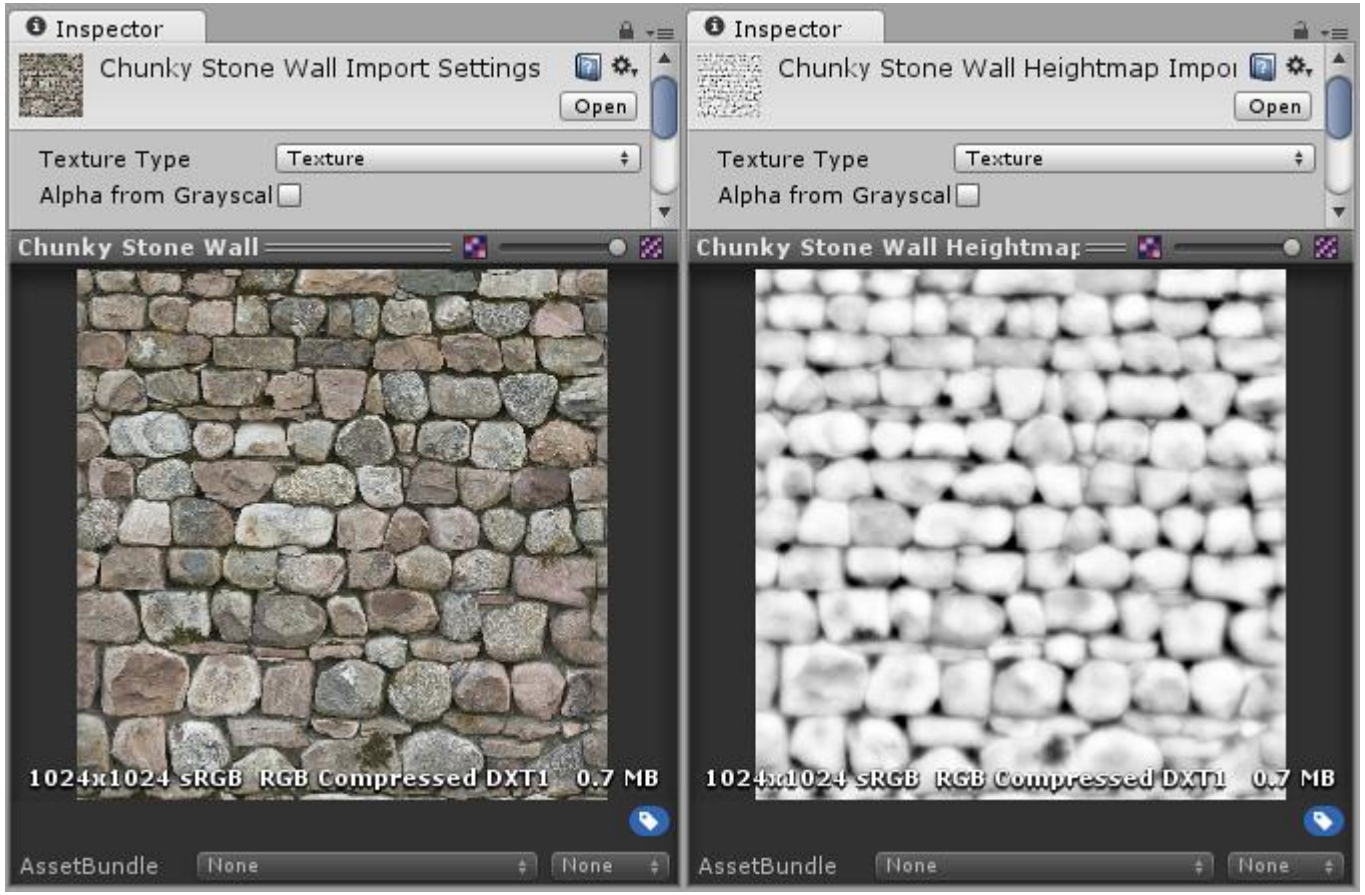




Metallic to 0.8



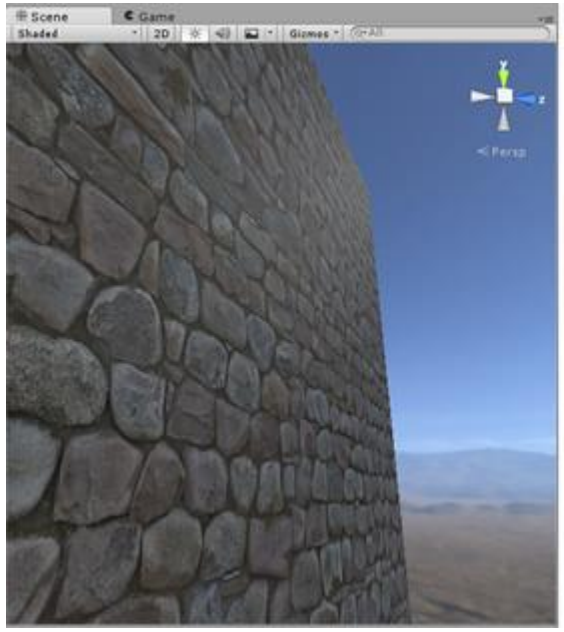
03. Material / Heightg Map



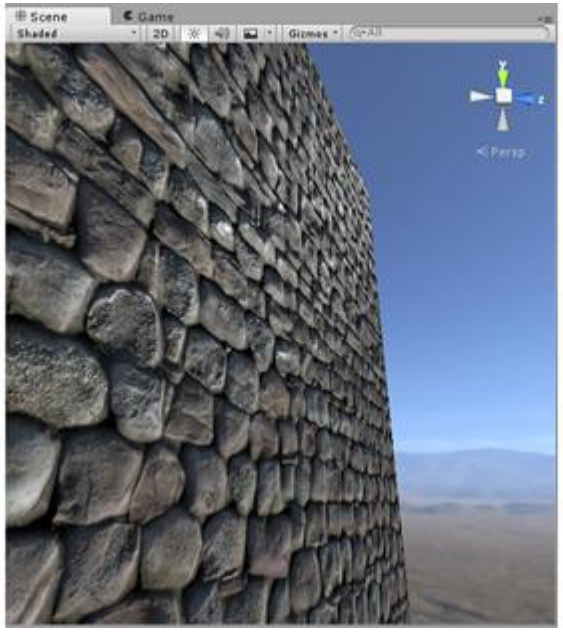
하이트맵은 흰색 영역이 텍스처의 높은 영역을 나타내고 검정색은 낮은 영역을 나타내는

그레이스케일 이미지

03. Material / Heightg Map



Albedo only



Albedo + normal

Albedo + normal+height map

03. Material / Occlusion Map

복잡한 Occlusion Map 사례

캐릭터의 소매에서 주변광에 노출되거나 조명에서 숨겨지는 영역을 식별

before

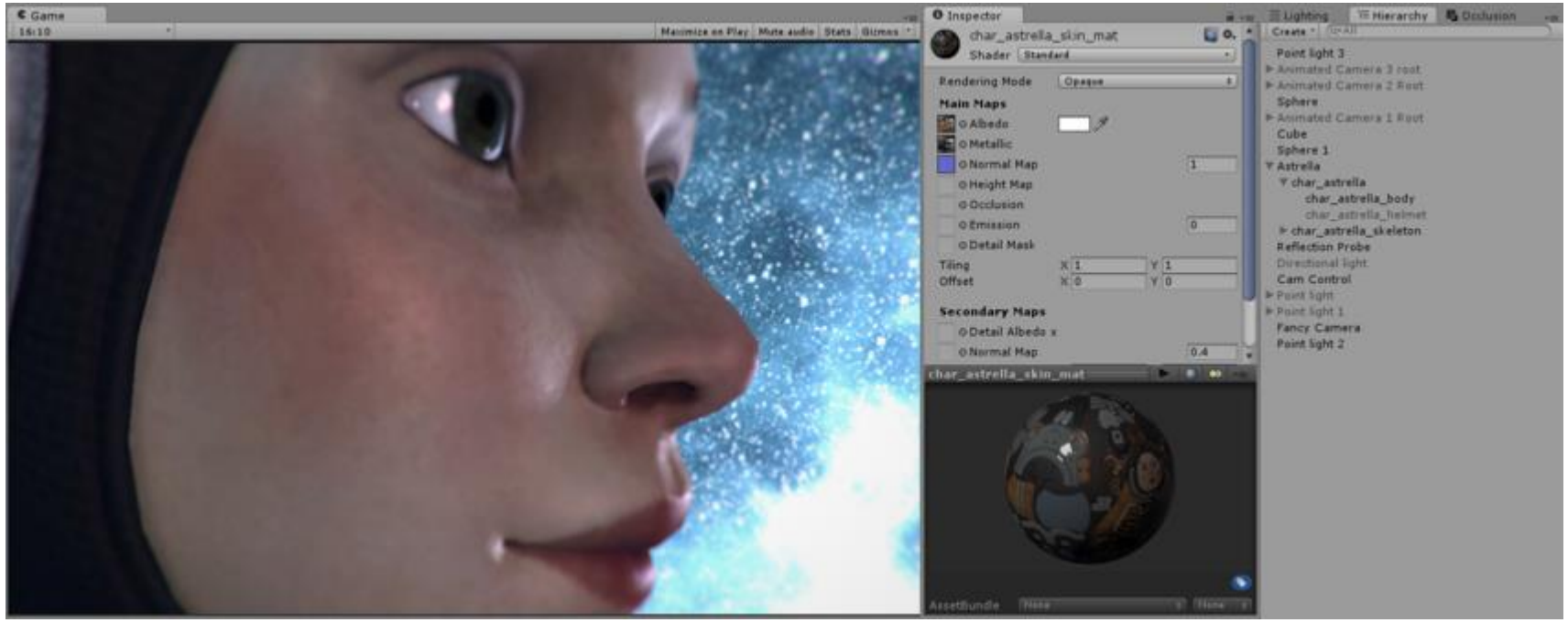
after

03. Material / Emission

•**None** - 오브젝트는 이미시브처럼 보이지만 주변 오브젝트의 조명에는 영향이 없습니다.
•**Realtime** - 이 머티리얼의 자체 광원이 씬의 실시간 전역 조명 계산에 더해지므로 움직이는 오브젝트를 포함한 주변 오브젝트의 조명이 자체 광원에 영향을 받습니다.
•**Baked** - 머티리얼의 이미시브 광원이 씬의 정적 라이트맵에 베이크되므로 다른 주변 정적 오브젝트가 이 머티리얼에서 빛을 받는 것처럼 보이지만 동적 오브젝트에는 영향이 없습니다.

03. Material / 보조맵

보조 맵(또는 디테일 맵)을 사용하면 위에 나열된 메인 텍스처 위에 두 번째 텍스처 집합을 오버레이
-가까이에서 볼 때 디테일이 선명하게 보이고 더 멀리서 볼 때 일반적인 디테일 수준으로 보이게-



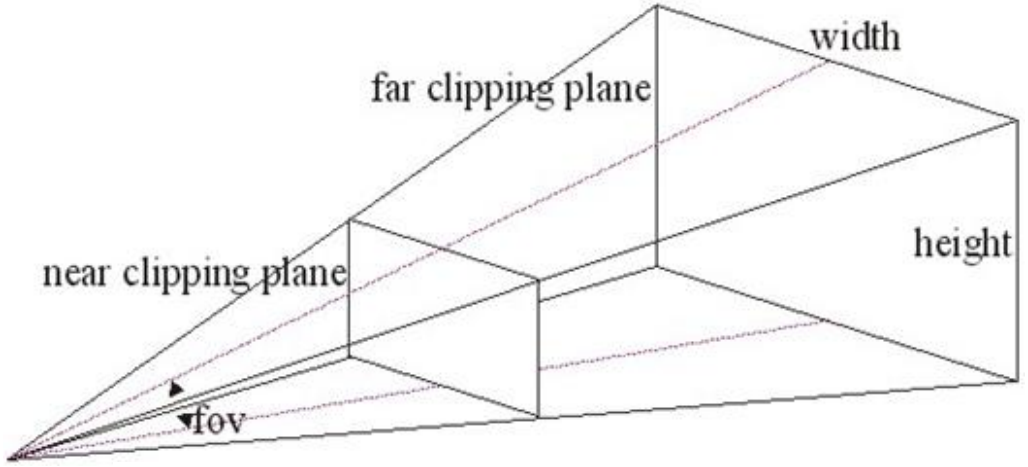
Shader

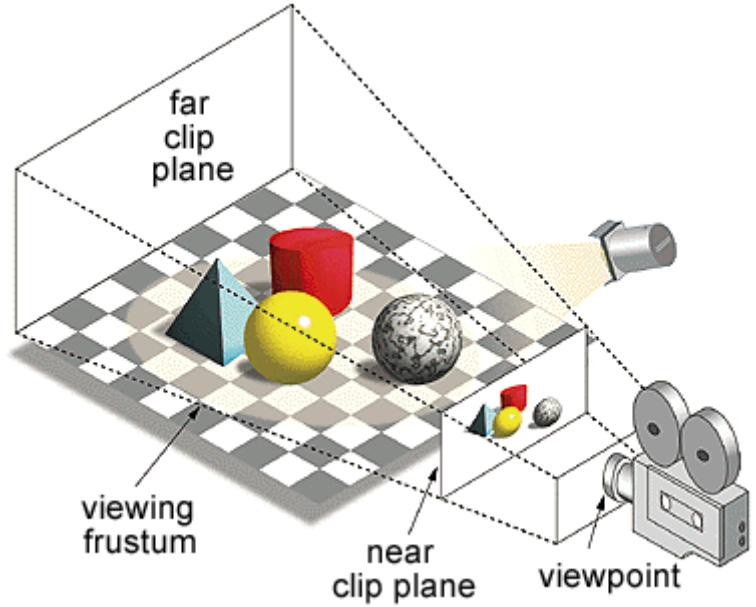
- Shader : 3D model 표면의 표현 방법에 대한 알고리즘 --> 스크립트
- Material과 연계된 Property

- Built-in Shader
  - FX : 조명과 글래스 효과
  - GUI : UI를 위한 Shader
  - Nature : 나무, 터레인
  - Particles : 파티클을 위한 Shader
  - Skybox : skybox를 위한 shader
  - Sprite : 2D Sprite용 Shader
  - Toon : Cartoon 스타일 렌더링을 위한 shader
  - Unlit : 모든 조명과 그림자를 적용하지 않는 shader
  - Legacy : 이전 shader

CAMERA

개요

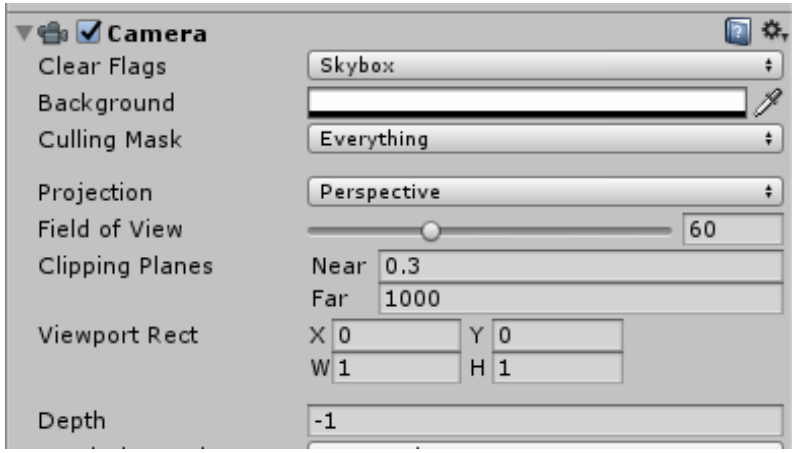


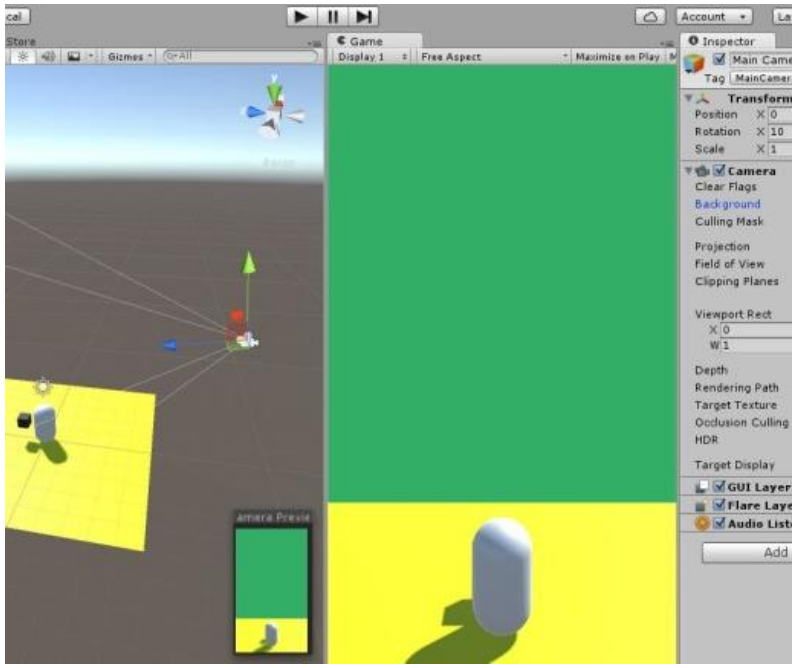


전체 object들 중 왼쪽의 Frustum(절두체) 내에 있는 Object들이 화면에 표시됨

Clipping plane : 잘려진 면

Camera / Inspector





Clear Flags 게임오브젝트를 그리고 남은 여백 처리
- Skybox : 여백을 skybox로
- Solid Color : 여백을 단색으로
- depth only : Sub camera의 여백을 투명처리
- Don't clear : clear하지 않음 (이전내용 유지)

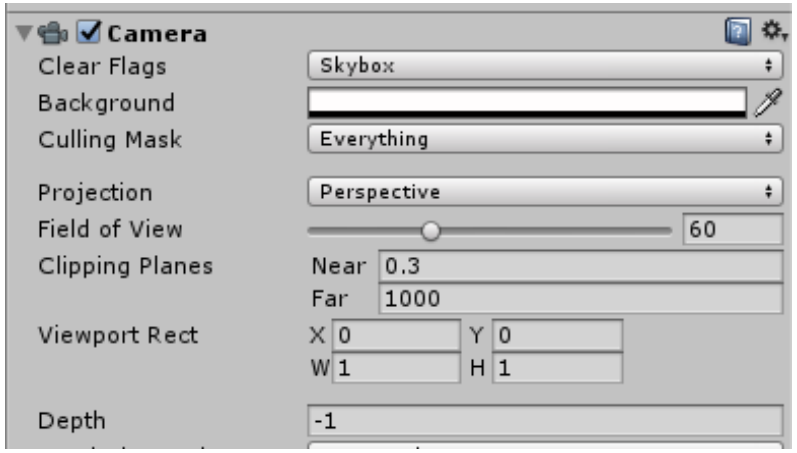
Background : 여백을 채울 색상을 지정

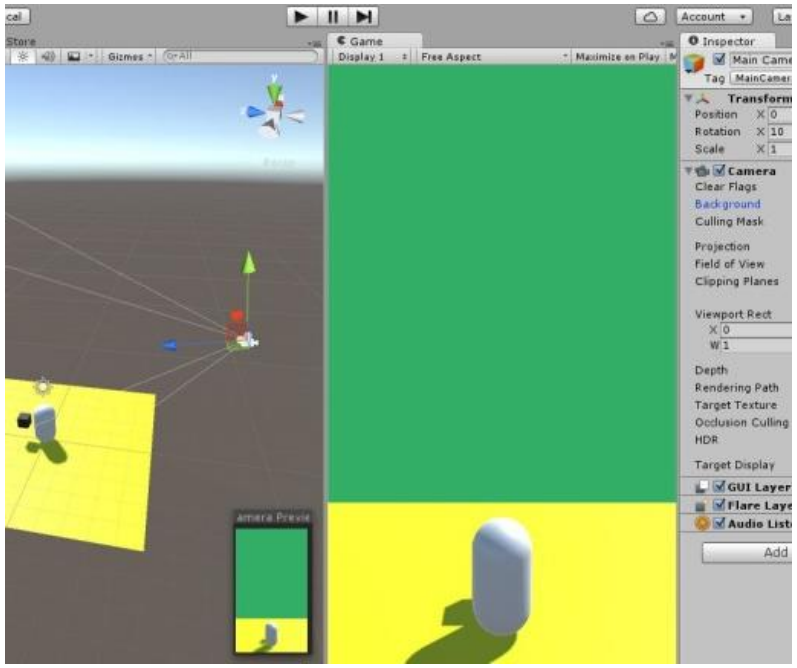
Culling mask : 레이어 지정

Projection : 투사 방식 결정
- Perspective : 원근
- Orthogonal : 직교

FOV : Y 축 방향에 대한 각도 지정
X 축은 가로세로비에 따라 자동 지정

Camera / Inspector





Size : 카메라 영역 크기

Clipping Plane : Clipping plane 위치 지정

Viewport Rect : 카메라 화면이 스크린에 보여질때의 위치와 크기 지정

Depth : 이 카메라로 그리는 순서 지정

PARTICLE

개요

Particle

작고 단순한 이미지, 혹은 mesh

Particle System

액체, 연기, 구름, 화염 및 마법 주문 등과 같이

불규칙적인 움직임 혹은 규모를 가지는 객체들을 표현하기 위한 시스템

Particle 특징

Features

- 파티클의 수명은 설정 가능하다.
- 파티클의 수명은 생성되거나 방출될 때 부터 시작된다.
- 파티클은 구체, 원뿔, 반구, 혹은 임의의 메시 모양의 공간 영역 안에서 임의의 위치에서 방출된다.
- 초당 방출되는 파티클 수도 설정 가능하다.
- 개별 파티클은 프레임이 업데이트 될때 마다 파티클의 움직이는 방향과 거리를 결정할 수 있다.
- 시간에 따라 파티클의 컬러(투명도 포함), 크기, 회전, 수명 등은 변경될 수 있다.
  (서서히 사라지게 할 수 있다)