Unity Certification Handbook

유니티 러닝센터

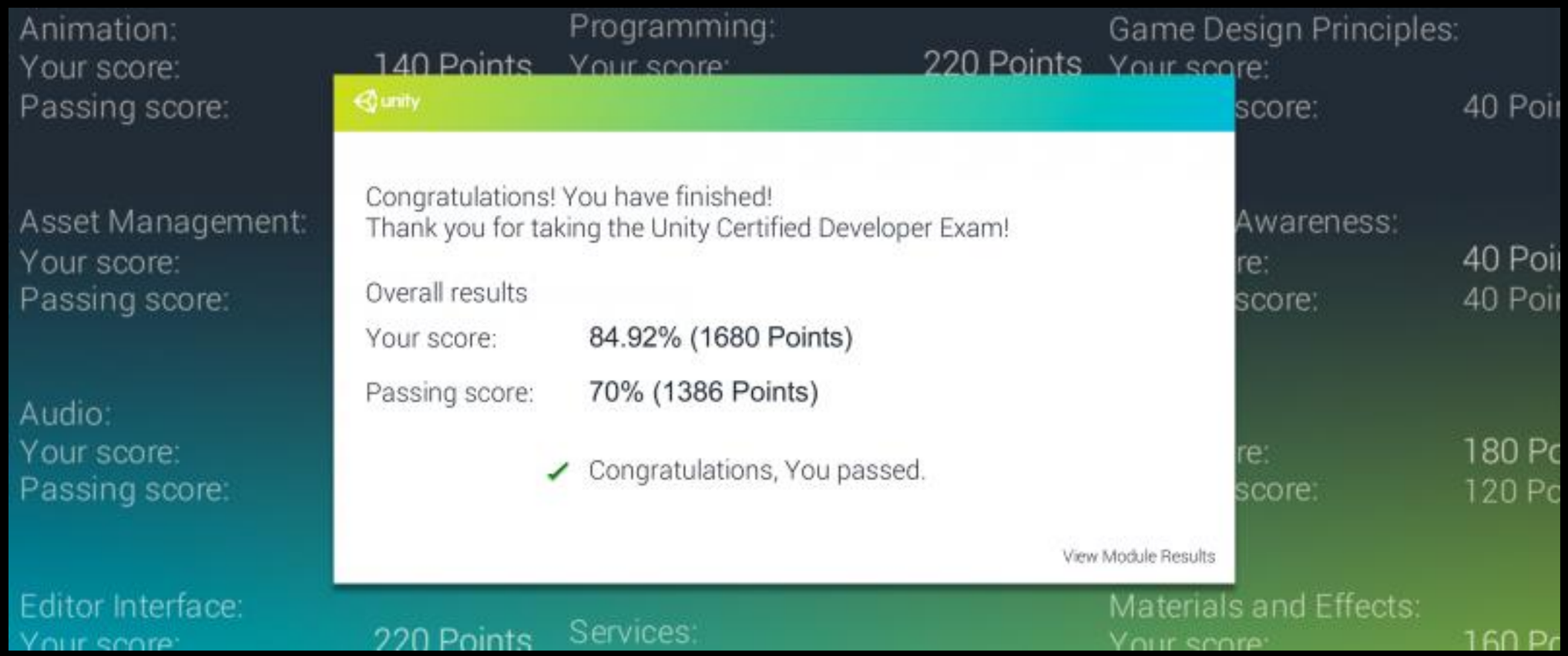
시험 개괄

- 유니티 에디터에 특화된 지식을 요구
  - X: 보편적인 개발 지식, 게임 프로그래밍
  - O: 유니티 에디터의 각 윈도우와 메뉴, 버튼의 기능
- 프로그래밍 파트는 매우 기초적인 수준

시험 개괄

- 섹션 별로 문제들이 나누어져 있음
- 섹션에서 Fail 을 받을 수 있으나, 합격 여부는 총점으로 따지므로 괜찮음
- 90분의 시간이 주어짐
  - 능숙한 사람은 40분 정도면 모든 문제를 푸는데 충분

시험 개괄



- 시험은 온라인으로 진행
- 빨리 풀고 빨리 집에 가도 무방하나, 뒤로감기 불가능
- 결과는 즉시 확인 가능

문제의 스타일

- 대부분은 유니티의 기초적인 사항
- 사용하는 법은 쉽지만, 의외로 잘 모르는 기능
- 같은 동작을 다른 방법으로 재현하는 방법
  - 예) 드래그&드롭이 아닌 방법으로, 프리팹 만들기

문제의 스타일

- 어떻게 새로운 패키지를 임포트 합니까?
  - (A) 메뉴 바에서, Components 를 선택 -> Import Package
  - (B) 메뉴 바에서, Assets 를 선택 -> Import Package
  - 정답은 (B)
- 평소에 생각없이 편하게 사용하던 기능들
- 하지만 이름을 딱히 외우고 쓰는게 아니라 틀리게 됨
- 이런 류의 문제는 그림도 첨부되어 있지 않음

시험 범위

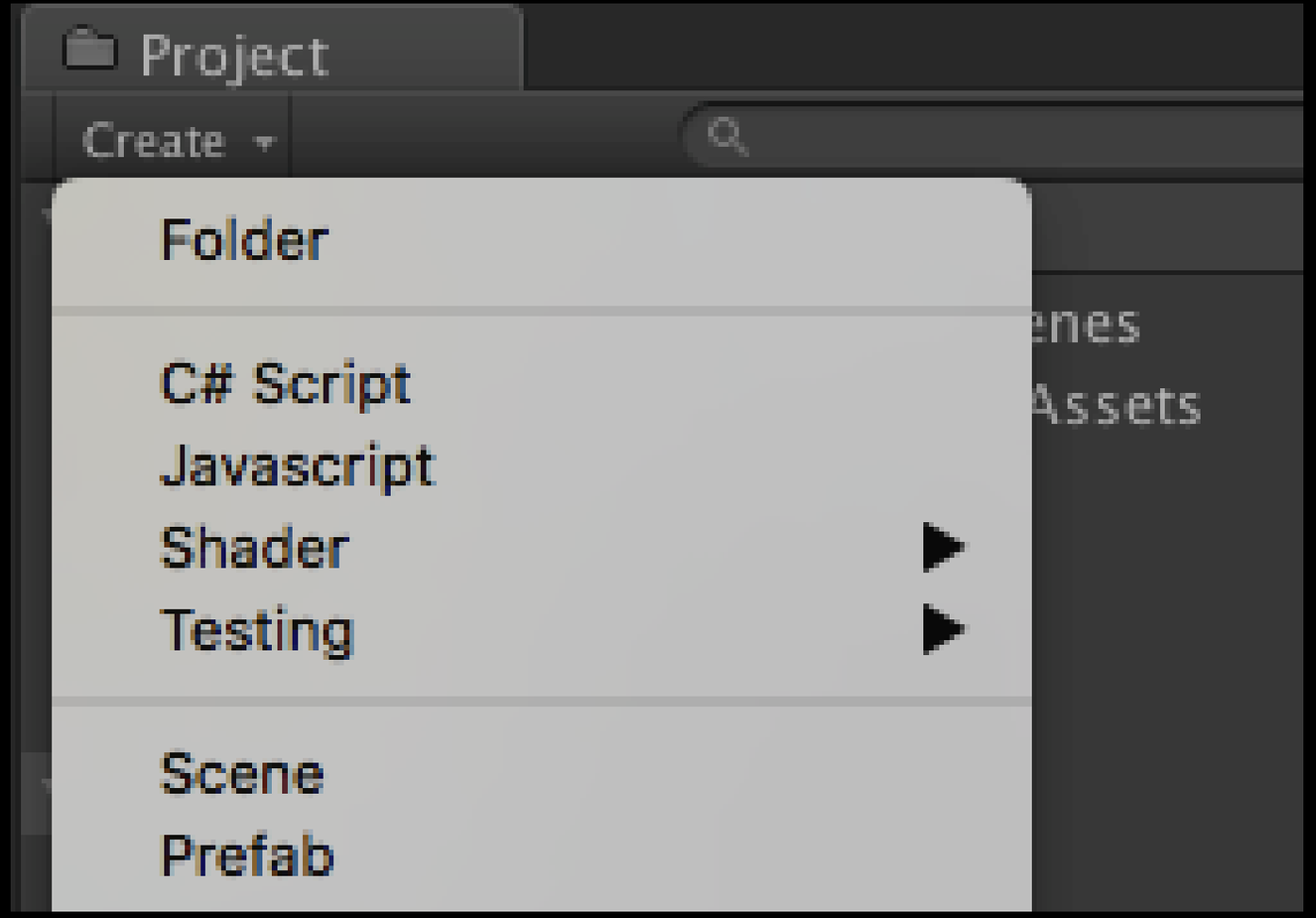
- 시험범위 (한글)
  - 링크 (드롭박스)
- 시험범위 (영어)
  - 링크 (유니티 공홈)

일반적인 사항

- 컴포넌트와 애셋의 혼동
  - Asset: 파일로 존재
    - Audio Clip, Animation, Animator Controller, Prefab
- Component: 게임 오브젝트에 부착
  - Audio Source, Animator

일반적인 사항

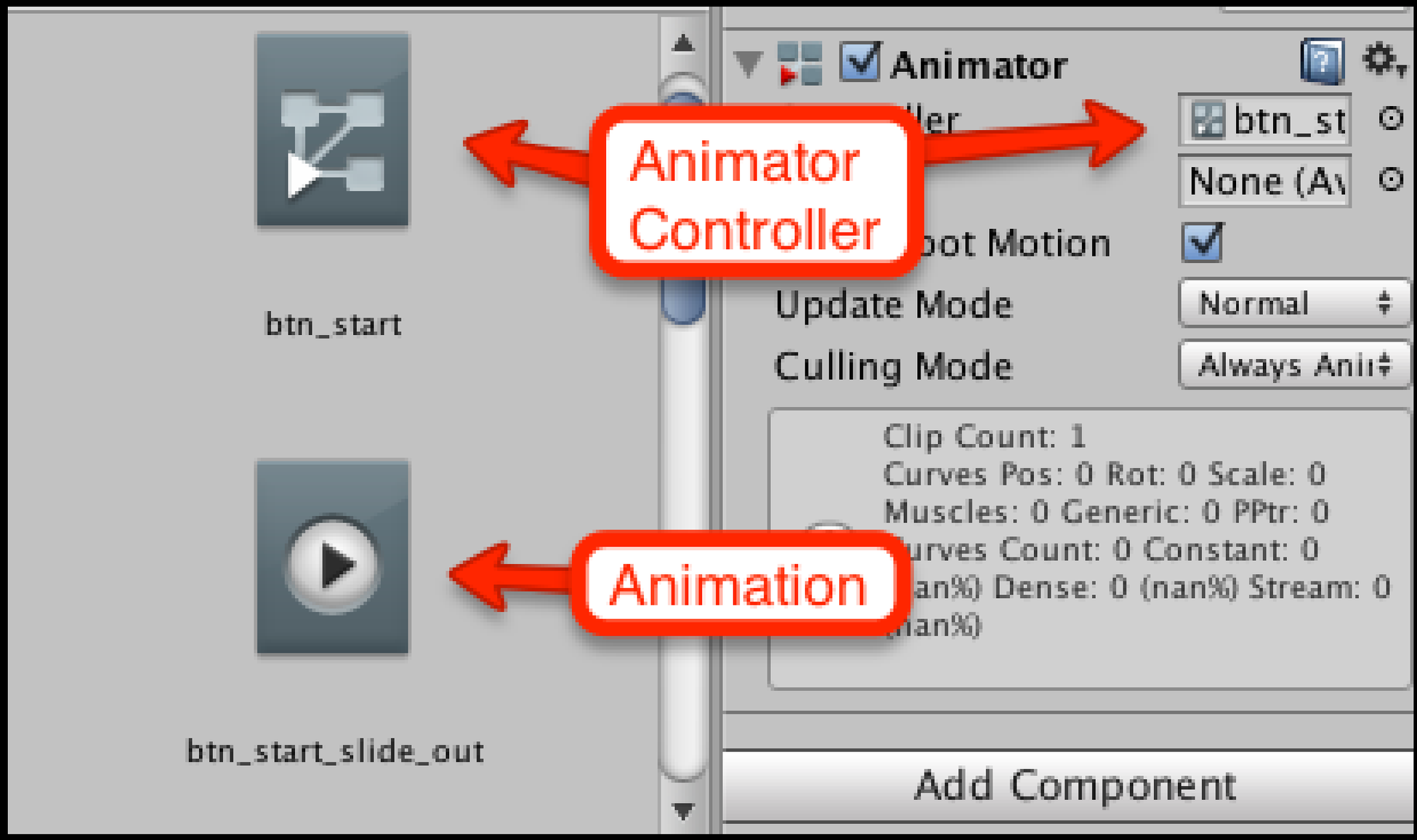
- 드래그 앤 드롭을 많이 쓰지만...
  - 몇몇 패널에는 Create 버튼이 있다는 사실을 잊지 말것
  - 드래그 앤 드롭으로 쉽게 가능한 것을, 다른 방법으로 재현하는 문제들



일반적인 사항

- 일관되지 못한 번역
  - 영어 고유 명사와 한국어를 일관성 없이 혼용
    - Rigidbody: 강체
    - Hierarchy: 계층
    - Gizmo: 기즈모 (???)
- 혼란스러워도 침착할 것
  - 번역이 이상하면, 영어 원문을 추측하는게 정신건강에 이로움

Animator



Animator 컴포넌트
Animation 애셋
Animation Controller 애셋

Animator Controller

- Animator 컴포넌트
  - 애니메이션 시스템을 유니티 게임 오브젝트에 적용
  - Animator 와 달리, Animator Controller 는 애셋 파일!
- 애니메이터는 FSM 구조를 가짐
  - Finite State Machine (유한 상태 기계)
    - 한번에 하나의 상태를 유지할 수 있음
    - 상태와 상태 사이에는 Transition (변이) 가 존재
- 두가지 이상의 애니메이션을 혼합(Blending) 가능

Animator Controller

- Animator 를 게임 오브젝트에 적용하는 방법 두가지
  - 1. 단순 드래그앤 드롭
    - Animator Controller 를 오브젝트에 드래그-드롭
  - 2. 수동으로 추가
    - Add Component -> Animator 컴포넌트 추가
    - 원(CirIce) 버튼을 통해 Animator Controller 지정

Animator Controller

- Animator Controller 생성 방법 3가지
  - 1. 프로젝트 패널에서
    - 여백에 마우스 오른쪽 클릭
    - Create -> Animator Controller
  - 2. 프로젝트 패널에서
    - Create 버튼 클릭 -> Animator Controller

Animator Controller

- 3. 프로젝트 패널에서
  - 여러개의 Sprite 들을 선택
  - Scene 패널에 드래그-드롭
  - 자동으로 2D 애니메이션를 함께 생성해주는 경우

Animator | Scene | Game

Layers | Parameters

Name

Forward 0.0

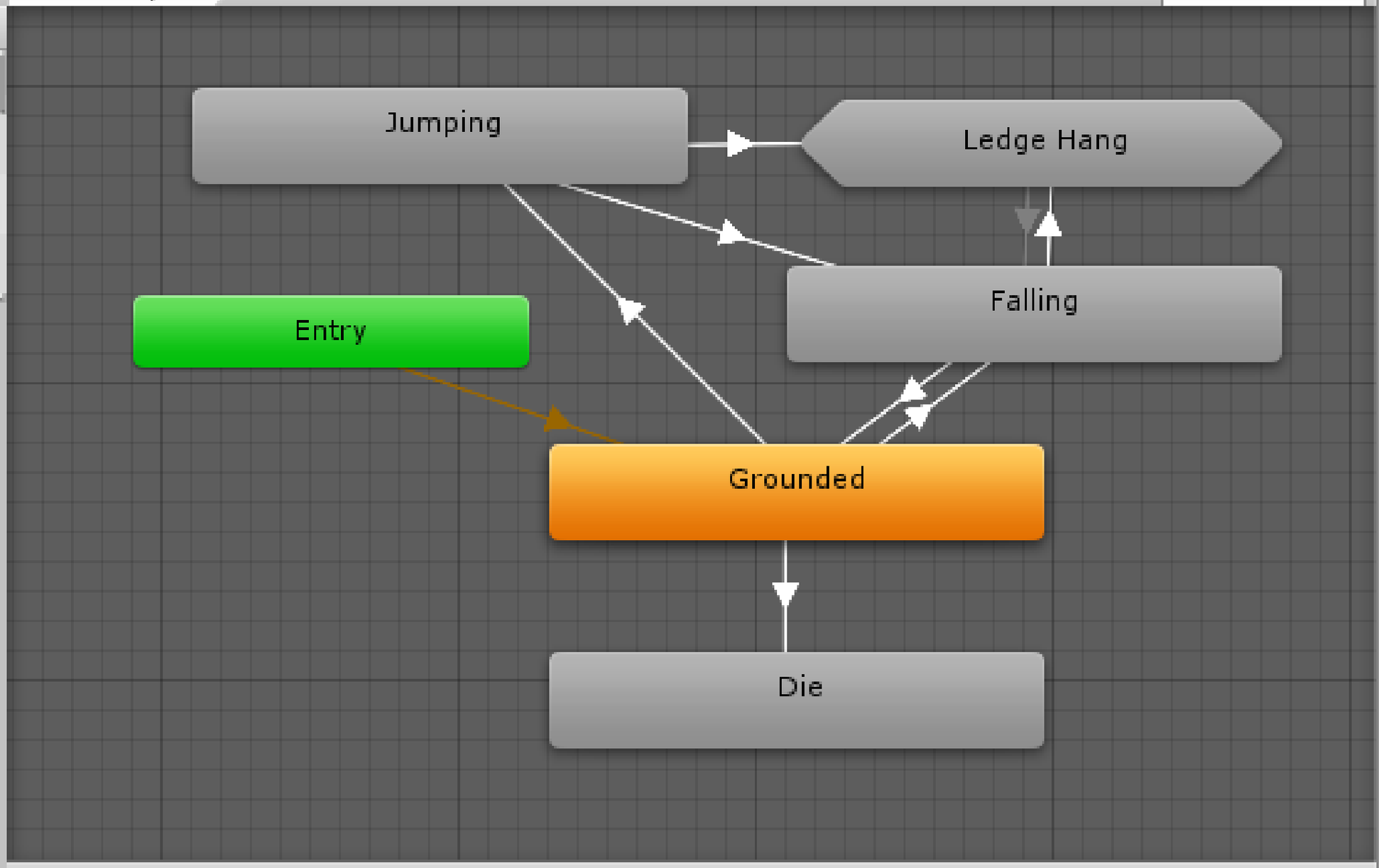
Strafe 0.0

Fire

Die

Base Layer

Auto Live Link

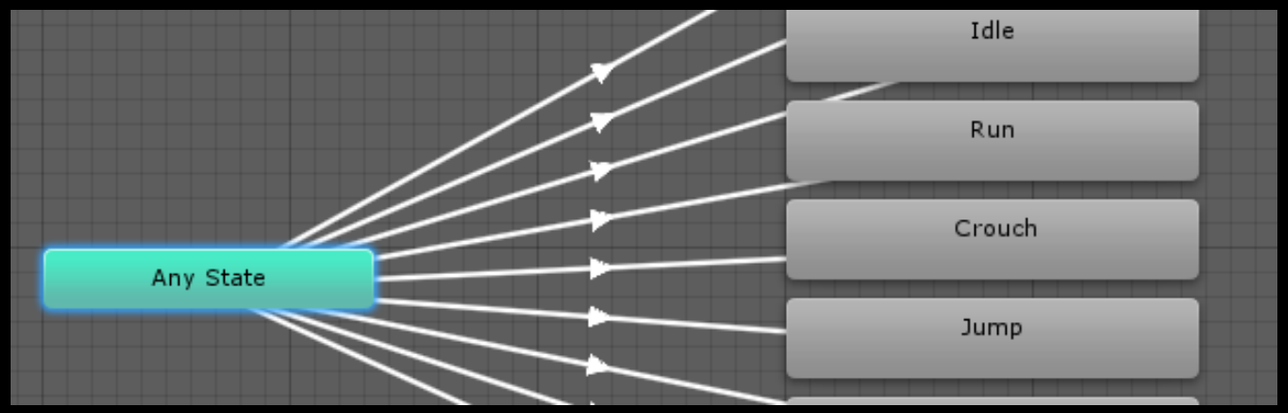


Animators/Player Animator.controller

State

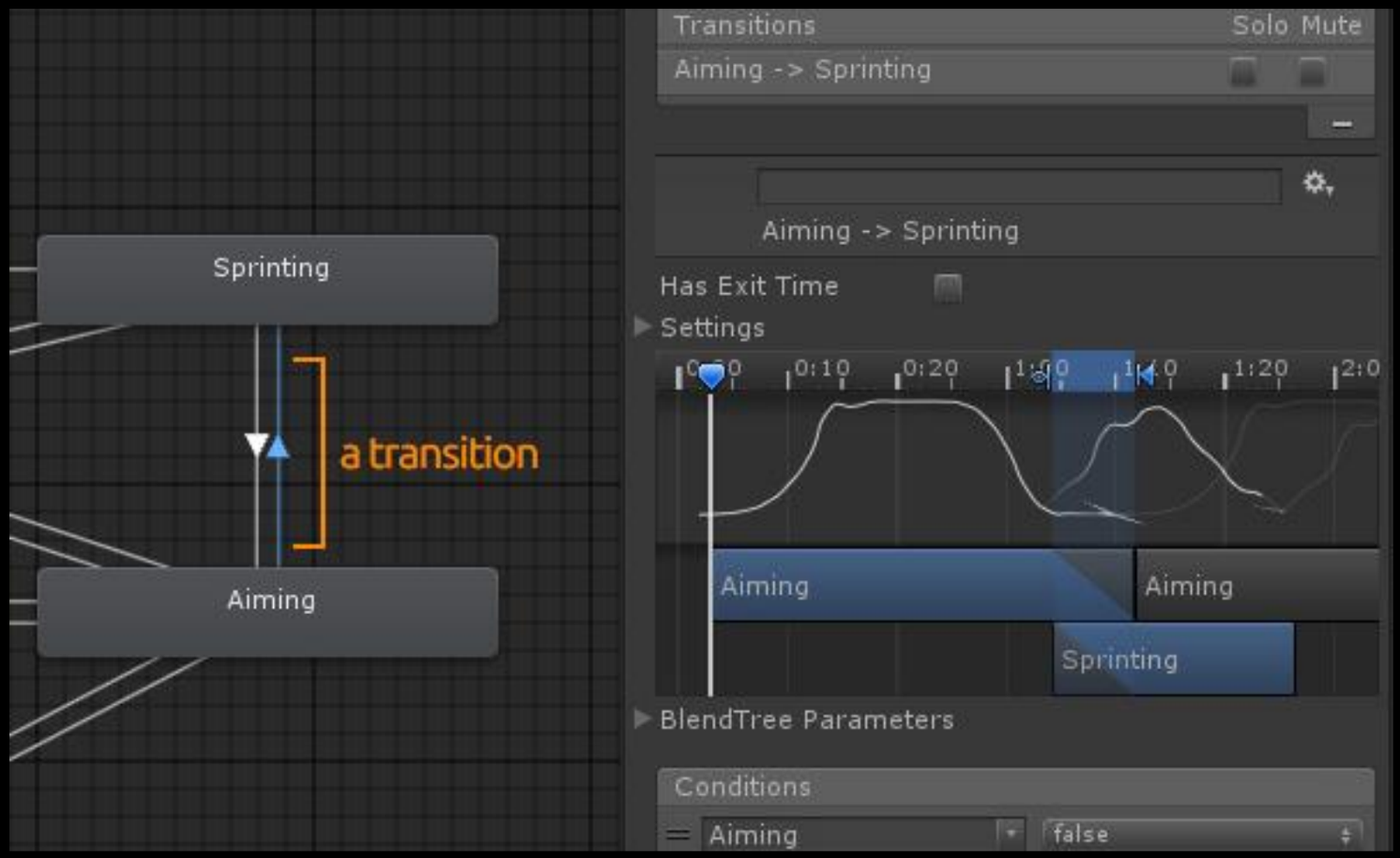
- State: 어떤 애니메이션이 재생될지 결정
  - FSM 에서는 한번에 하나의 State 만 재생
- Entry 노드: State Machine 의 진입로
- Exit 노드: State Machine 의 출구
- Default State: Entry 와 이어져 가장 먼저 실행됨

State



- Any State
  - 여기서 Transition을 이을 경우
  - 현재 어떤 State 에 있던 상관하지 않고
  - 조건만 만족하면 지정한 상태로 바로 이동

Transition

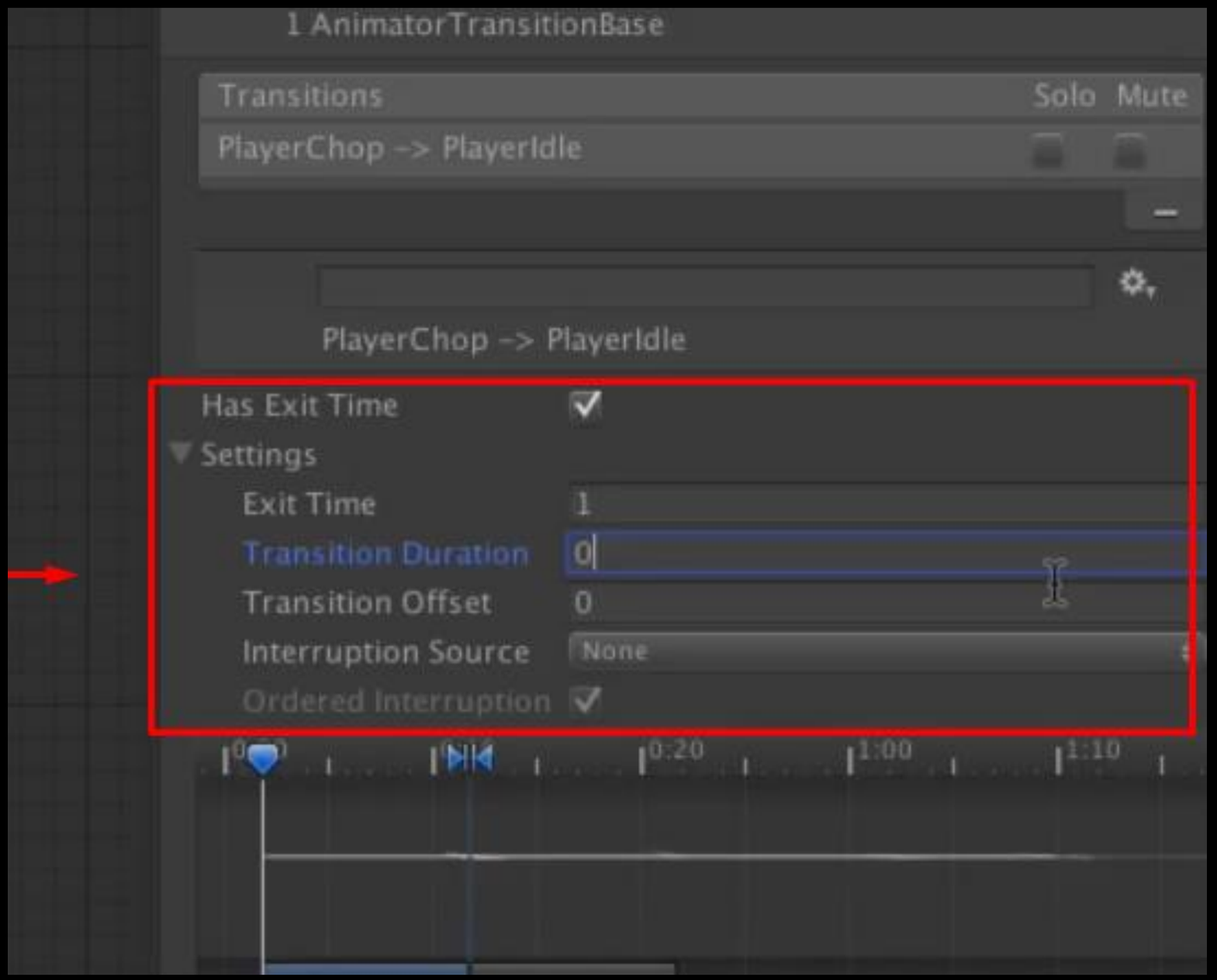


- State 에서 다른 State 로의 전환
- 파라미터를 통해 언제 발동 될지 조건 지정 가능

Transition

- 트랜지션 만들기
  - State 노드에 마우스 오른쪽 클릭
  - Make Transition 선택후,
    다른 State 노드에 화살표를 잇기

Transition


1 AnimatorTransitionBase
Transitions
Solo
Mute
PlayerChop -> PlayerIdle
PlayerChop -> PlayerIdle
Has Exit Time
Settings
Exit Time
1
Transition Duration
0
Transition Offset
0
Interruption Source
None
Ordered Interruption
0:20
1:00
1:10

Transition

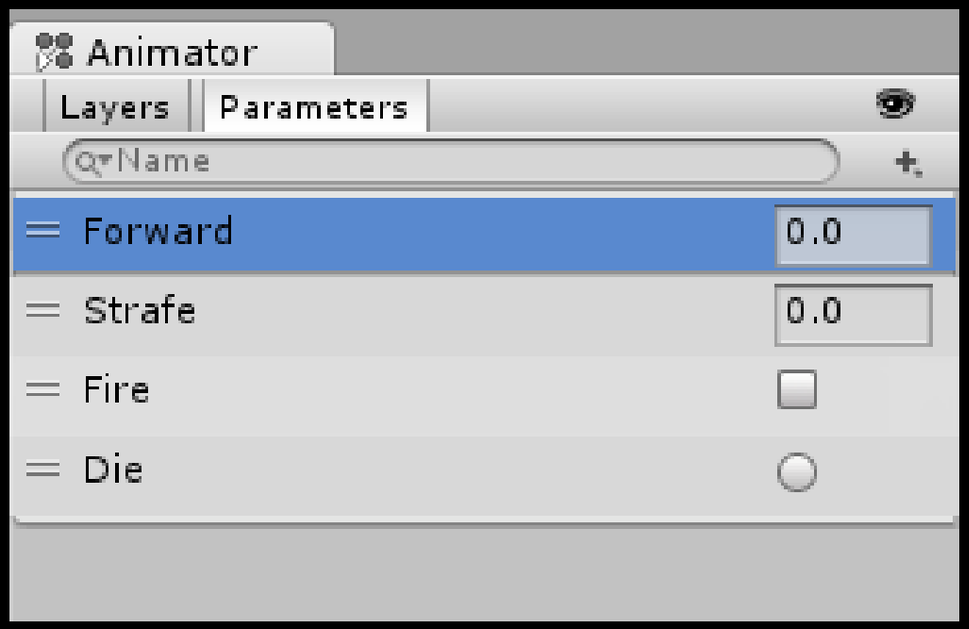
- 트랜지션 속성(프로퍼티)
  - Has Exit Time
    - Exit Time 만큼 지나면, 자동으로 다음 상태로 이동 (조건 무시)
  - Exit Time: 얼만큼 뒤에 자동으로 트랜지션 하는지
  - Fixed Duration
    - Exit Time , Duration 을 퍼센티지(%)가 아닌 초 단위로 설정
- Conditions: 파라미터를 통해 트랜지션의 조건을 지정

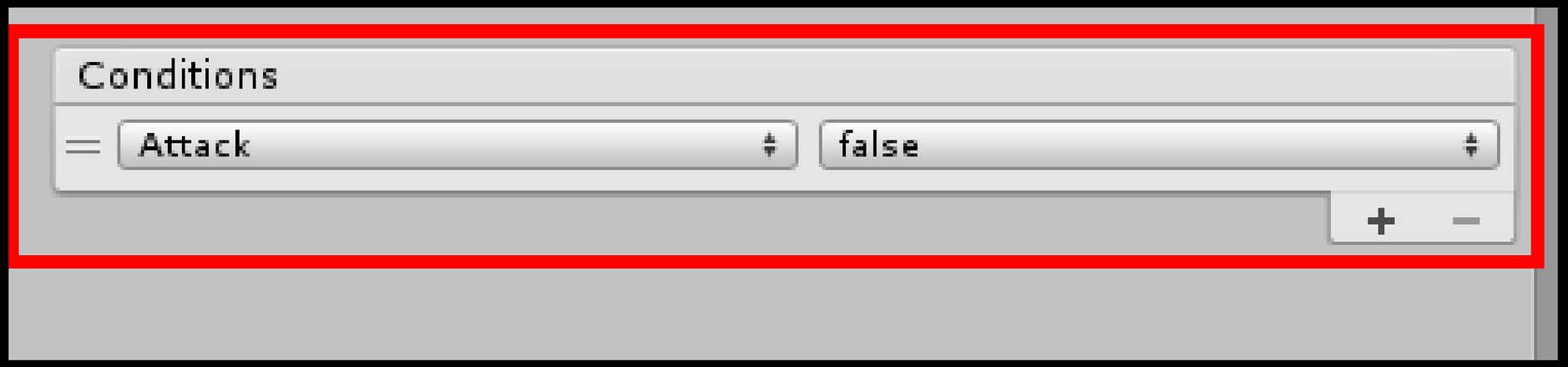
State

- 새 상태를 추가하는 두가지 방법
  - 드래그 앤 드롭
    - 프로젝트에서 애니메이션 파일을 선택
      - Animator Controller Window 의 빈 여백에 드래그-드롭
  - 수동으로 생성 및 할당
    - Animator Controller Window의 빈 여백에 마우스 오른쪽 클릭
    - 컨텍스트 메뉴에서 Create State->Empty를 선택

파라미터 Parameter

- 파라미터 Parameter
  - 애니메이터를 제어하기 위한 값
    - Transition 의 Condition 에 쓰임
  - Float, Int, Bool, Trigger
  - Trigger 는 True 가 된 직후 False





Asset

Audio Clip

- Audio Clip 은 Audio Source 에 의해 재생되는 오디오 데이터 (오디오 파일) 애셋
- Audio Source 는 오디오 애셋이 아닌 컴포넌트!

Audio Clip

- 오디오 클립 압축률
  - 압축률이 높을 수록
    - 파일용량이 작으나, 압축을 푸는 과정에서 성능이 많이 요구됨
  - 압축률이 낮을 수록
    - 음질이 좋고 압축을 푸는 과정이 없어 오버헤드가 없음
    - 메모리 요구량이 커짐
- 즉 압축률이 높아 용량이 작을 수록
  - 오히려 성능은 더 먹게 되므로, 혼동 하지 말것
  - 압축률 PCM < ADPCM < Vorbis/MP3

Audio Clip

- Audio Clip 의 압축 형식
  - PCM
    - 무압축: 높은 품질, 큰 크기. 낮은 CPU 점유.
    - 매우 짧은 효과음에 사용
  - ADPCM
    - 압축률 높음: 낮은 품질, 작은 크기.
    - PCM 보다 높지만, Vorbis/MP3 보다 CPU 점유가 낮음
    - 자주 재생될 필요가 있는 총기, 발자국 등에 사용.

Audio Clip

- Audio Clip 의 압축 형식
  - Vorbis/MP3
    - Quality Slider 로 압축률 지정 가능
      - 가장 낮은 품질, 가장 작은 파일 용량
      - 보통 길이의 배경 음악에 적당
  - HEVAG
    - PS Vita 에 사용
    - ADPCM 과 유사한 스펙

Default Game Objects

- 프리미티브 오브젝트 (Primitive Objects)
  - Primitive Objects 뜻: 가장 기초적인 형태의 오브젝트
  - Cube 큐브
  - Plane 플레인
  - Quad 쿼드
  - Sphere 스피어
  - Capsule 캡슐
  - Cylinder 원기둥

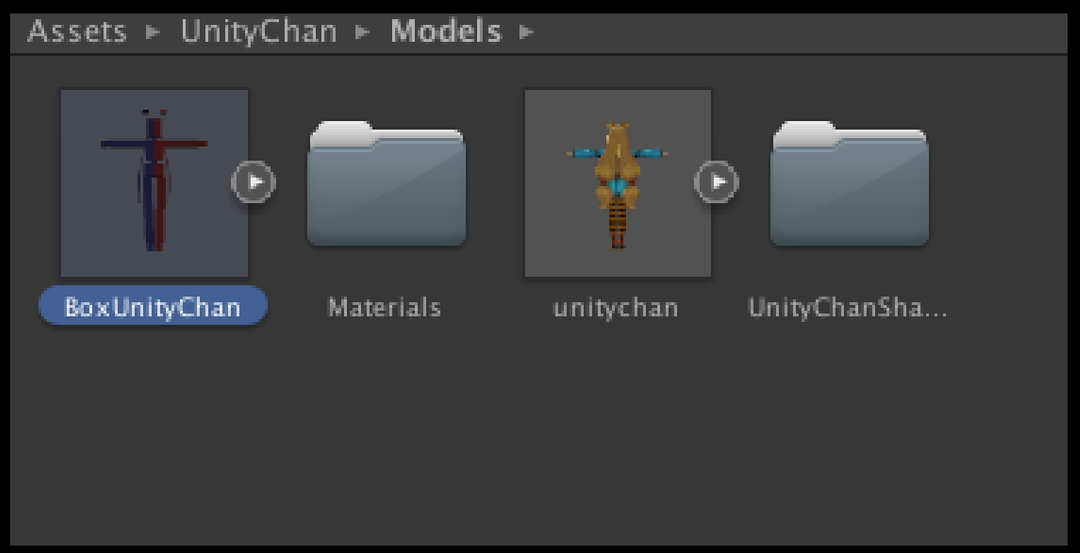
Default Game Objects

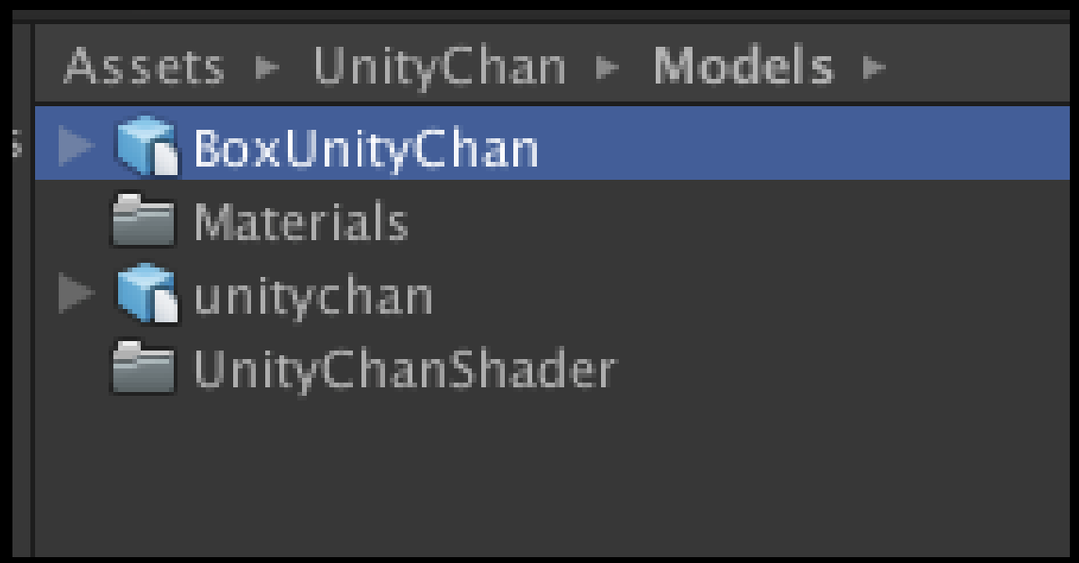
- 애셋 임포트
  - 각 폴더와 파일에 대해 .meta 파일 생성
  - Image Files – BMP, TIF, TGA, JPG, PSD
  - 3D Model – .FBX, .dae, .3DS, .dxf, .obj
  - Meshs & Animations – 자동으로 3D 모델과 함께 임포트
  - Audio 파일들

Script Type

- 유니티 스크립트의 대표적인 3가지 타입
  - MonoBehaviour 상속: 컴포넌트로서 사용
  - Scriptable Object 상속: 애셋 파일에서 인스턴스화
  - Editor 상속: 유니티 에디터 커스터마이징

Models





- 아이콘 형태만 보고, 애셋의 종류를 맞추는 문제 존재

Prefabs

- 미리 완성된 게임 오브젝트를 애셋으로 만들어, 필요할때 재사용 가능한 것

- 생성하는 세가지 방법
  - 하이라키에서 프로젝트 패널로 드래그 앤 드롭 (가장 대중적)
  - 상단의 메뉴에서 Asset > Create Prefab
  - 프로젝트의 Create 버튼 > Prafab

Prefabs

- 사용법
  - 하이라키나 씬에 드래그 드롭하여 찍어내기
  - 스크립트를 통한 인스턴스화
    - Instantiate(원본_프리팹,위치,회전)
- 프리팹으로 찍어낸 오브젝트는 파란색으로 보임

Scene

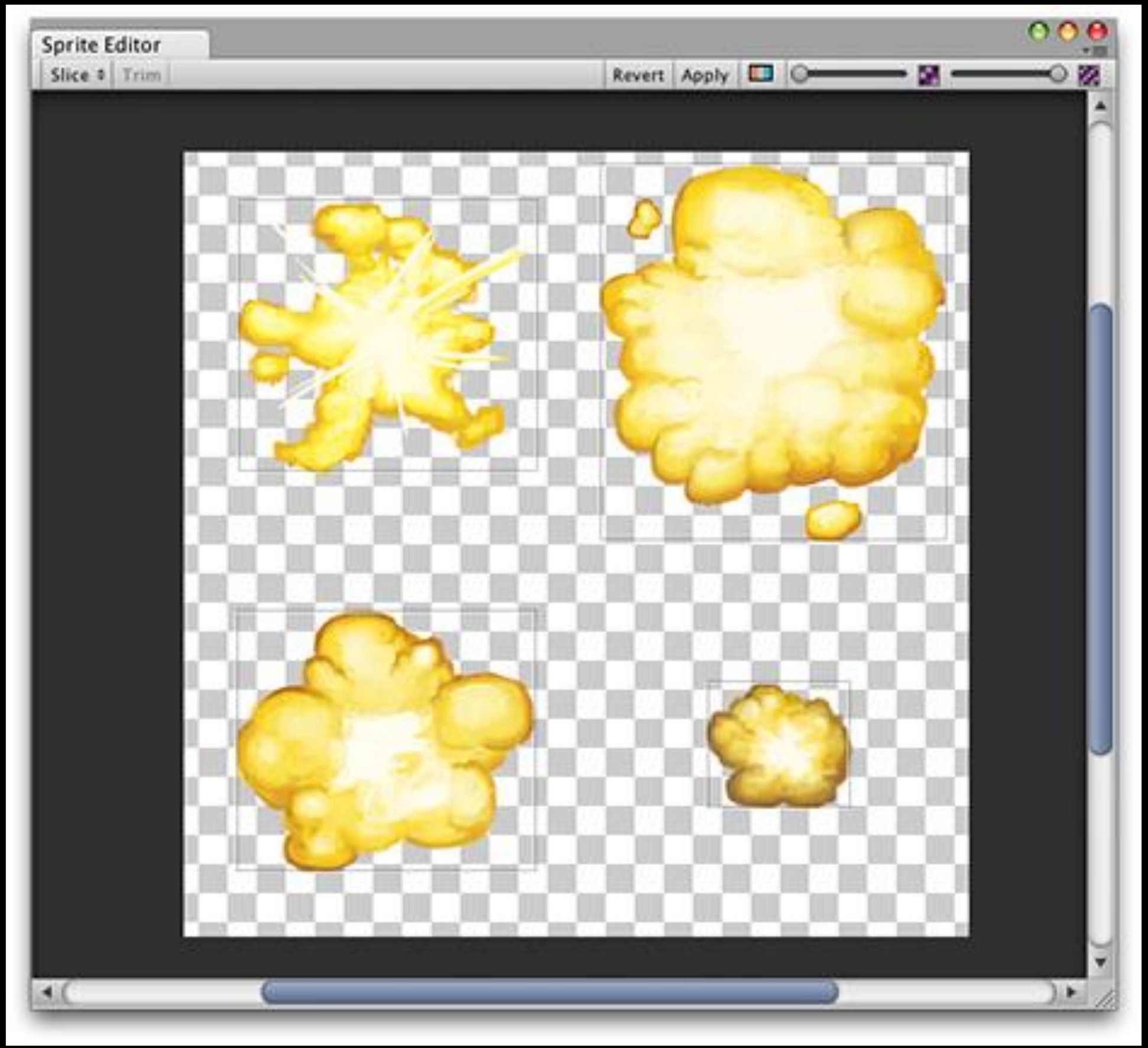
- 씬 Scene 은 게임을 위한 오브젝트를 담는 단위
- 각각의 레벨이나 메뉴 등을 만든다
- 하나의 씬을 하나의 독립적인 레벨로 생각하라

Scene

- 씬의 생성
  - 유니티로 처음 프로젝트를 실행한 순간 저장되지 않은 씬이 생성된다
  - File > New Scene
- 씬 저장
  - File > Save Scene
  - Crtl/Cmd + S
- 씬 로드
  - 씬 파일을 더블 클릭

Sprite

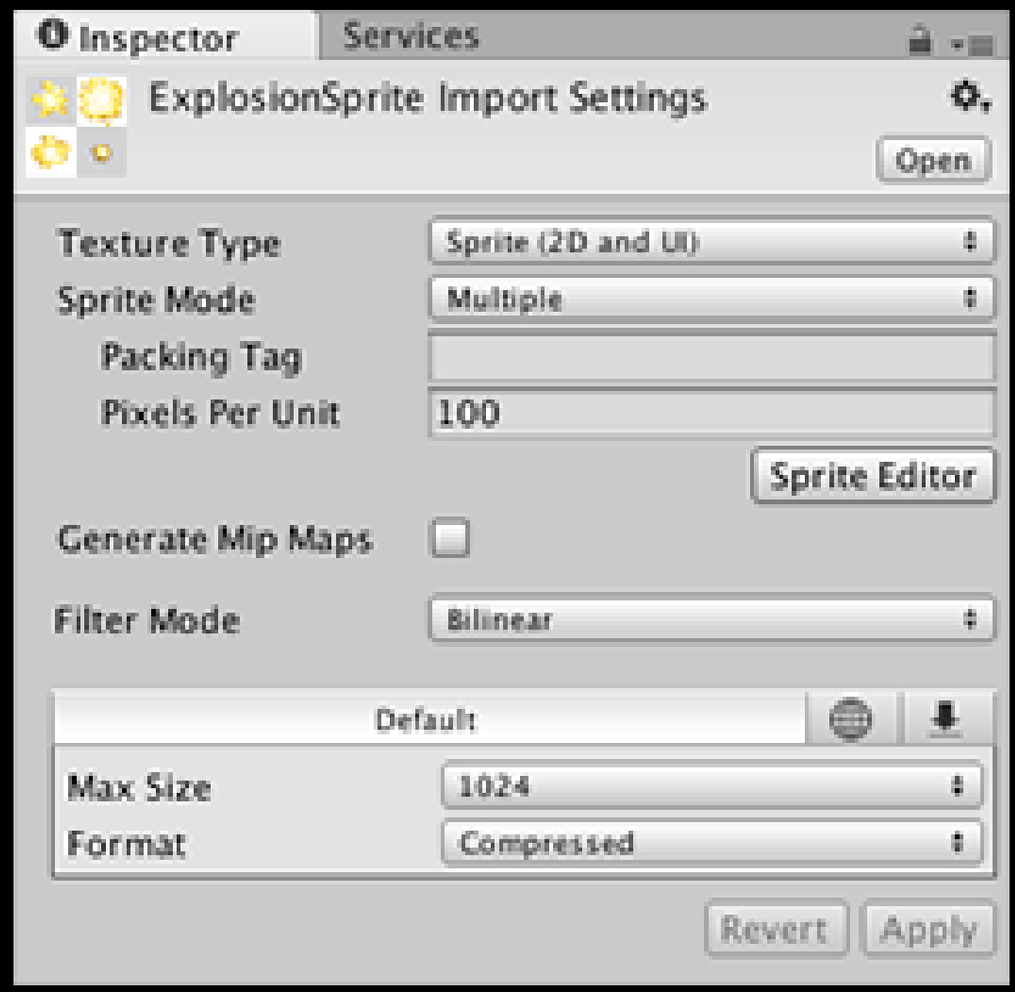
- 스프라이트 Sprite 는 2D 그래픽 애셋 이다
- Sprite Editor 를 통해 Sprite 를 편집 가능


Sprite Editor
Slice
Trim
Revert
Apply

Sprite Editor

- 스프라이트는 보통 연관된 여러개의 그래픽을 하나의 이미지에 동시에 담는다
- Sprite Editor 로 이 합쳐진 이미지들을 분리하여 가져올 수 있다

- Sprite Editor 실행법
  - 프로젝트 뷰에서 애셋 선택
  - Texture Import 인스펙터에서 Sprite Editor 를 클릭
    - Sprite Editor 윈도우가 실행됨
- 하나의 스프라이트에 여러 그래픽이 포함된 경우 Sprite Mode 를 Multiple 지정



Sprite Editor

- Sprite 를 자를려면 Slice 버튼
- Apply 와 Revert 로 작업물을 저장/되돌리기
- Trim: 영역의 공백을 잘라내기
- Automatic Slicing: 자동으로 자르기

Audio

Audio Mixer

- 오디오 믹서는 오디오 소스 들의 소리를 믹싱
- 다른 그룹의 출력을 효과의 입력으로 지정 가능
- Add 버튼을 눌러 효과 추가 가능

Console | Audio Mixer

Exposed Parameters (0)

Mixers +

- MainAudioMixer (Audio
- NewAudioMixer (Audio I

Groups +

- Master
  - Music
  - Reverb
  - Effects

Views +

- View

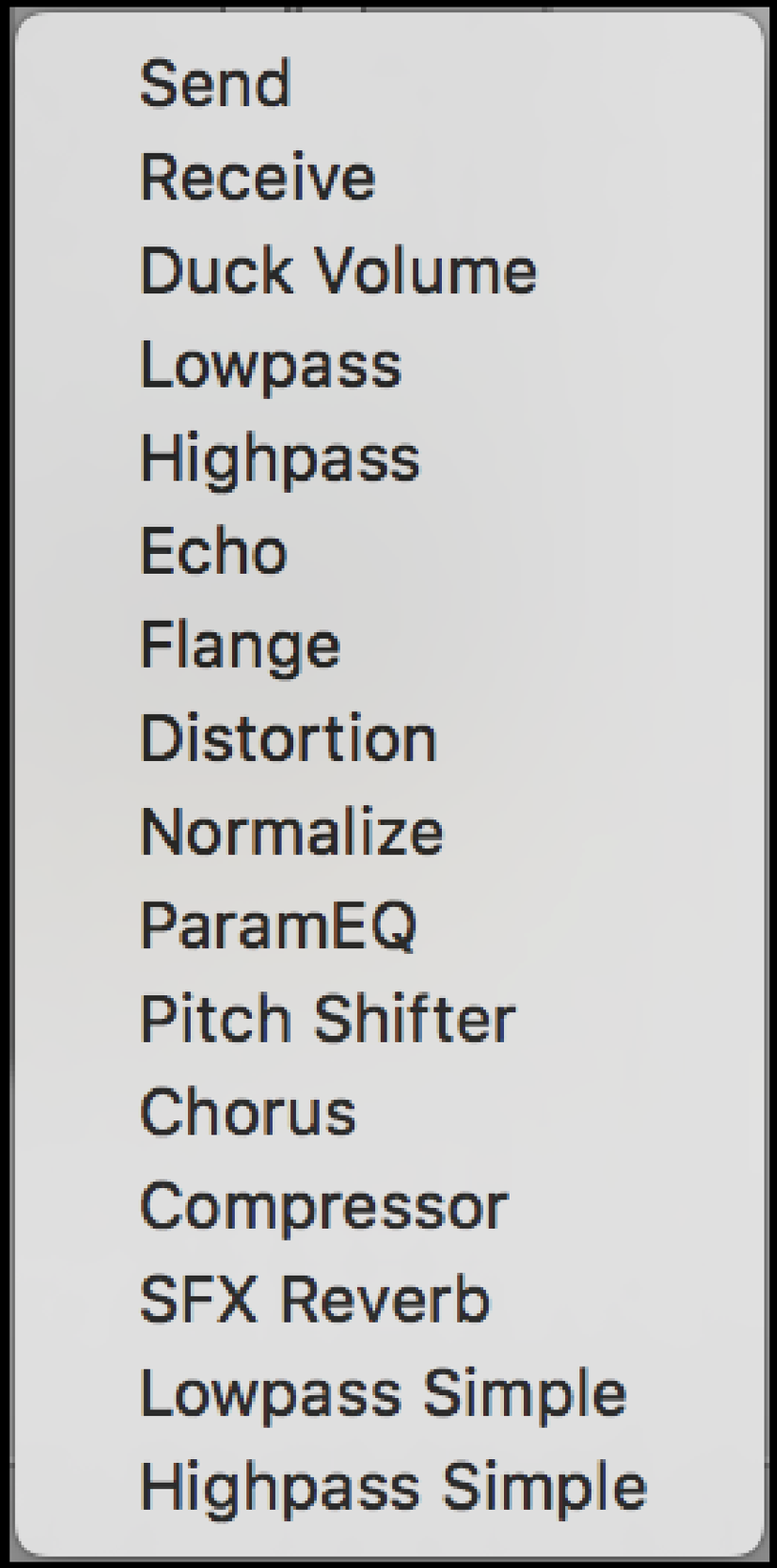
Snapshots +

- Snapshot ★

Master	Music	Reverb	Effects
20	20	20	20
0	0	0	0
-2			
-20	-20	-20	-20
-40	-40	-40	-40
-60	-60	-60	-60
-80	-80	-80	-80
-80.0 dB	-80.0 dB	-80.0 dB	-80.0 dB
S M B	S M B	S M B	S M B
Attenuation	Attenuation	Attenuation	Attenuation
	Lowpass	Receive	s Reverb\Receiv
	Compressor		
	Flange		
	Distortion		
	Echo		
	Highpass		
	s Reverb\Receiv		
Add..	Add..	Add..	Add..

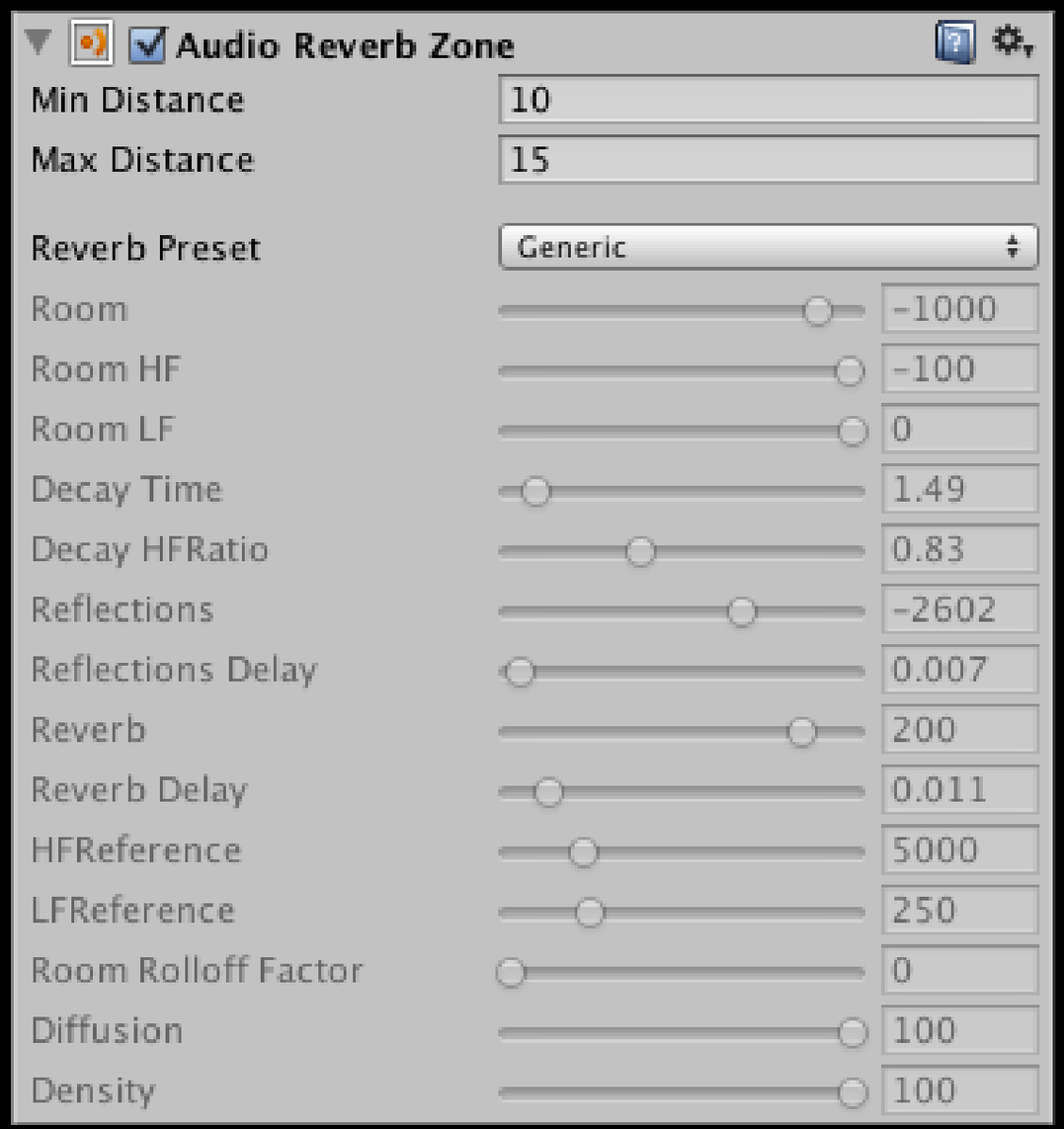
Audio Effects

https://docs.unity3d.com/Manual/class-AudioEffectMixer.html



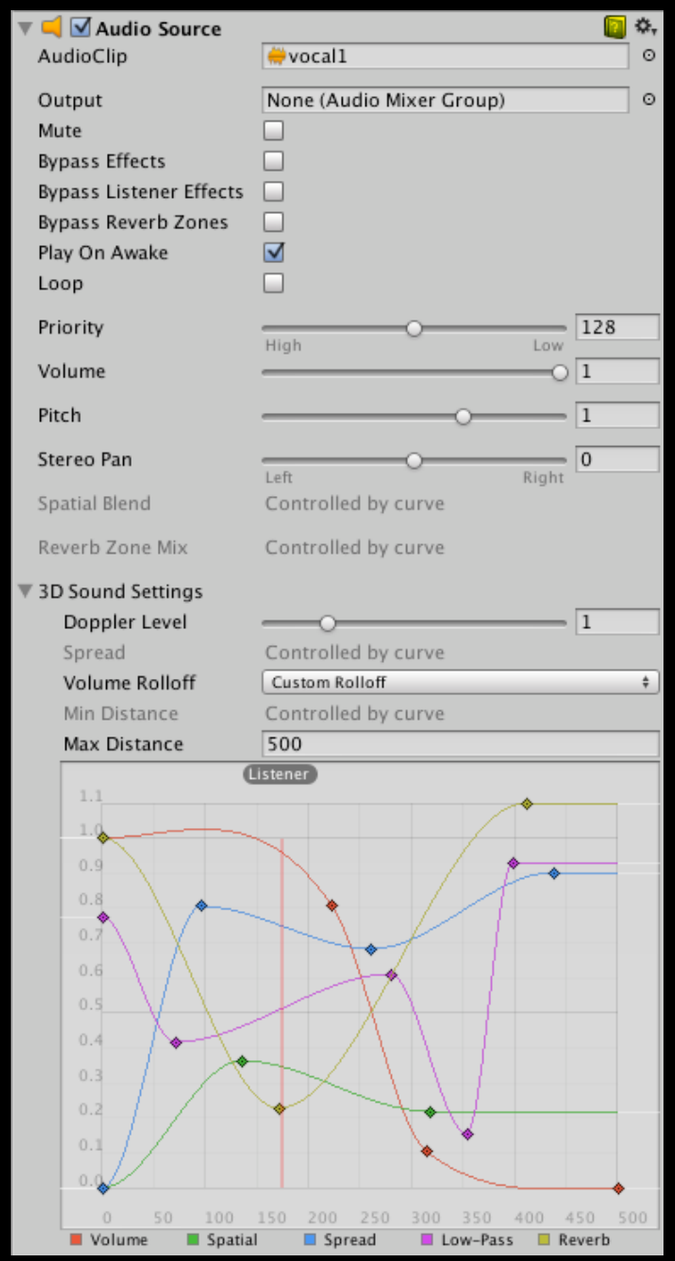
Audio Reverb Zone

- Audio Clip 을 재생할때 Listener 의 위치에 따라 음향을 왜곡
- 동굴등을 들어가고 나올때의 음향차를 구현하는 등
- Min Distance – 그라디언트(점진적)로 효과가 적용되는 반경
- Max Distance – 음향 왜곡이 최대 출력이로 적용되는 반경



Audio Source

- Audio Clip 은 재생할 클립
- Audio Listener
  - 오디오를 듣는 컴포넌트
  - Audio Source 와 혼동 하지 말것



Audio Source

- Audio Clip – 재생할 오디오 클립의 레퍼런스
  - 드래그 드롭
  - Select 버튼 사용
- Play On Awake – 게임이 시작될때 바로 재생
- Loop – 반복 재생
- Pitch – 음의 높 낮이
- Volume – 음의 크기

Audio Source

- 도플러 효과에 유의
  - 관찰자(듣는 사람) 입장에서 파동원(음원)의 상대적인 속도 차이에 의해 파동이(소리가) 달라지는 것

  - 소방차 소리가 가깝고 멀어짐에 따라 사이렌 소리가 달라지는 것이 대표적인 예

가 소방차가 정지한 경우 | 두 관찰자는 같은 높이의 소리를 듣는다.

나 소방차가 움직이는 경우 | 소방차의 앞쪽에 있는 관찰자는 더 높은 소리를 듣고, 뒤쪽에 있는 관찰자는 더 낮은 소리를 듣는다.

Editor Interface

Editor Customization

- 레이아웃
  - 유니티의 각 뷰는 이름을 잡아 끌어서 재배치 가능
  - 유티니의 가장 기초적인 윈도우(혹은 뷰 or 패널) 들
    - 씬, 게임, 하이라키(계층), 콘솔, 프로젝트, 인스펙터
  - 툴바: 유니티의 가장 필수적인 툴들 집합
    - 유일하게 재배치 불가능한 인터페이스

Toolbar

Pivot

Account

Layers

Default

Hierarchy

Create

All

Animation 3 Root

Cam rig pivot

Astrella Sitting

SHIP 1

SHIP 2

Hierarchy Window

floor

Cube 1

Large Cube

support_014

Jet Fan 1

prop_fanLarge_aperture_

prop_fanLarge_motor_0(

walls_050

Scene

Game

Shaded

2D

Gizmos

All

Persp

Scene View

Inspector

prop_fanLarge_aperture_

Static

Tag

Untagged

Layer

Default

Transform

Position

-13.277

Rotation

Scale

Inspector Window

Cast Shadows

Receive Shadows

Materials

Size

1

Element 0

prop_fan_large_a

Use Light Probes

Reflection Probes

Blend Probes

Anchor Override

None (Transform)

prop_fan_large_aperture_m

Shader

Legacy Shaders/Bumped Spe

Add Component

Project

Create

Favorites

Assets

Animations

Animators

Art

_Textures

Heavy Machinery Textures

Materials

Characters

Environment

OtherAnimation

Sky

Astrella Scene

Assets ▸ Art ▸ _Textures ▸ Heavy Machinery Textures ▸ Materials

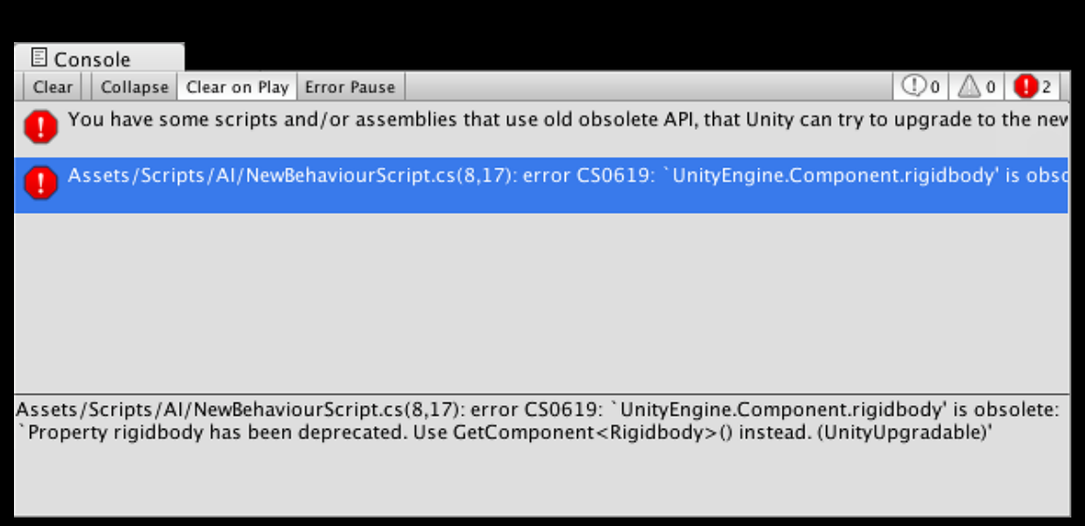
Project Window

Asset Store

- Window > Asset Store
- 미리 완성된 애셋을 구매하여 빠른 제작이 가능

Console

- Window > Console
- 에러와 경고, 기록 등의 메세지를 보여 주는 창
- Debug.Log 등으로 직접 메세지를 남길수 있음
- 메세지 종류
  - 회색 – 단순 기록이나 메세지
  - 노란색 – 경고나 권고 사항
  - 붉은색 – 런타임 에러나 코드 에러
- Collapse 버튼을 눌러 그룹화 하여 접을 수 있음

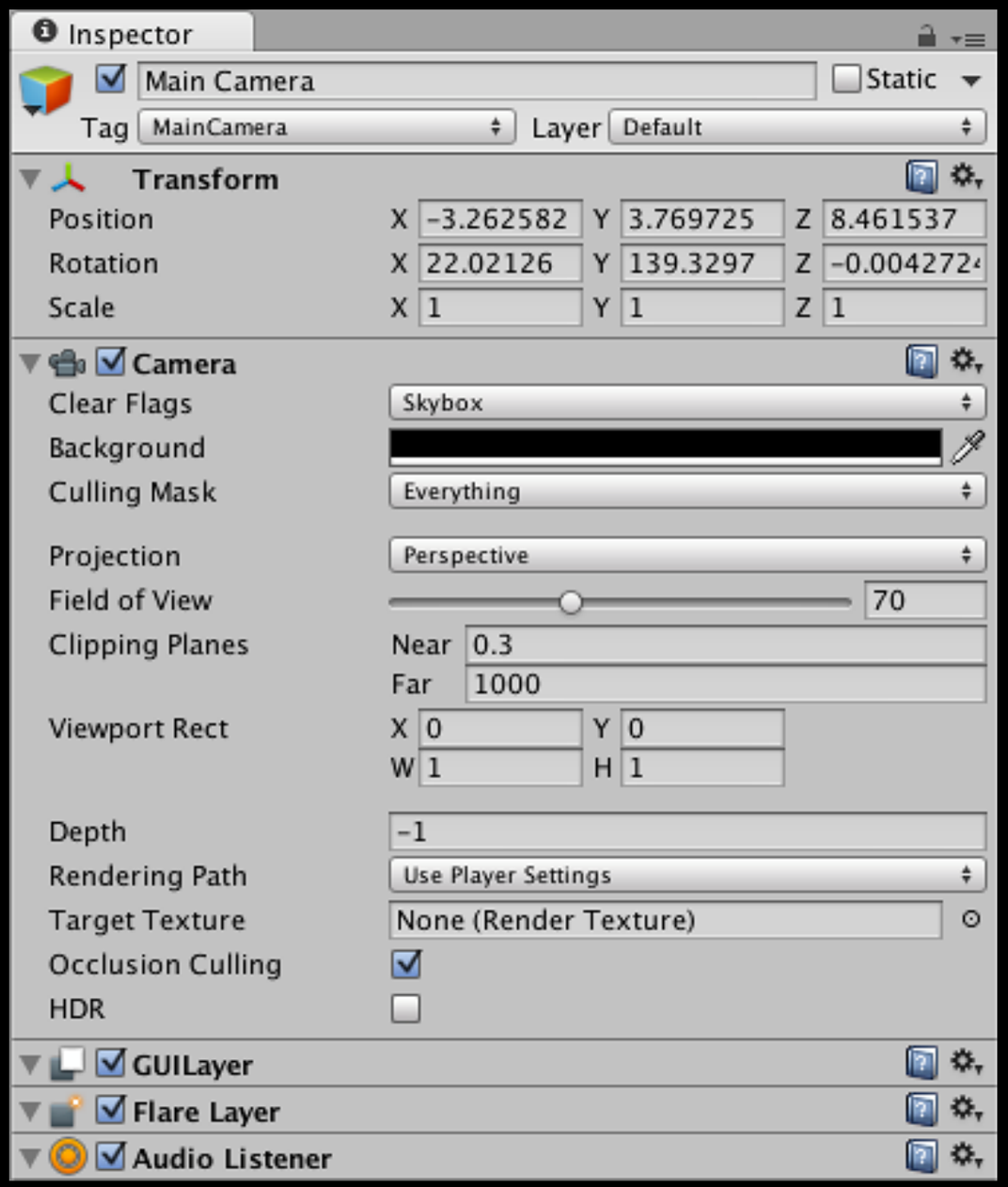


Hierarchy

- 현재 씬에 존재하는 모든 게임 오브젝트들이 리스트로 있는 곳
- 드래그-드롭을 통해 부모-자식 지정 가능
- Create 버튼으로 새 오브젝트 생성 가능
- 공란에 마우스 오른쪽으로 새 오브젝트 생성 가능
- 자식 오브젝트는 부모 오브젝트의 좌표계를 기준으로 위치함

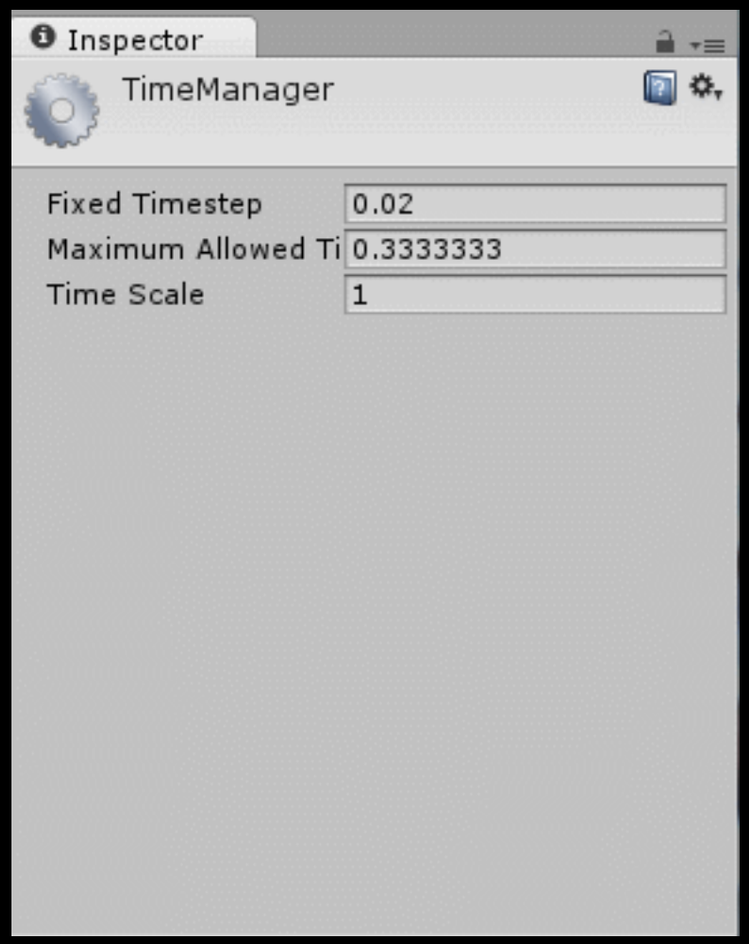
Inspector

- 인스펙터로 보는 것
  - 게임 오브젝트의 프로퍼티와 컴포넌트와 마테리얼의 모든 설정 값들
    - 애셋의 설정값
    - 기타 여러 에디터 설정 값들
- public 인 변수들은 보이게 된다



Inspector

- 컴포넌트의 값을 초기화
  - 해당 컴포넌트의 오른쪽 상단의 톱니 바퀴 버튼 > Reset 누르기
- 아이콘 할당
  - 게임 오브젝트를 씬에서 시각적으로 구분 할 수 있는 아이콘 할당 가능
  - 커스텀 아이콘도 가능

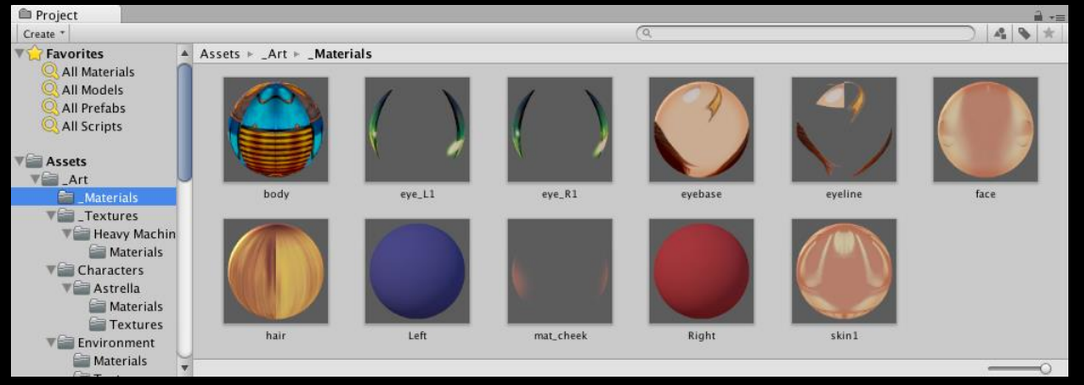


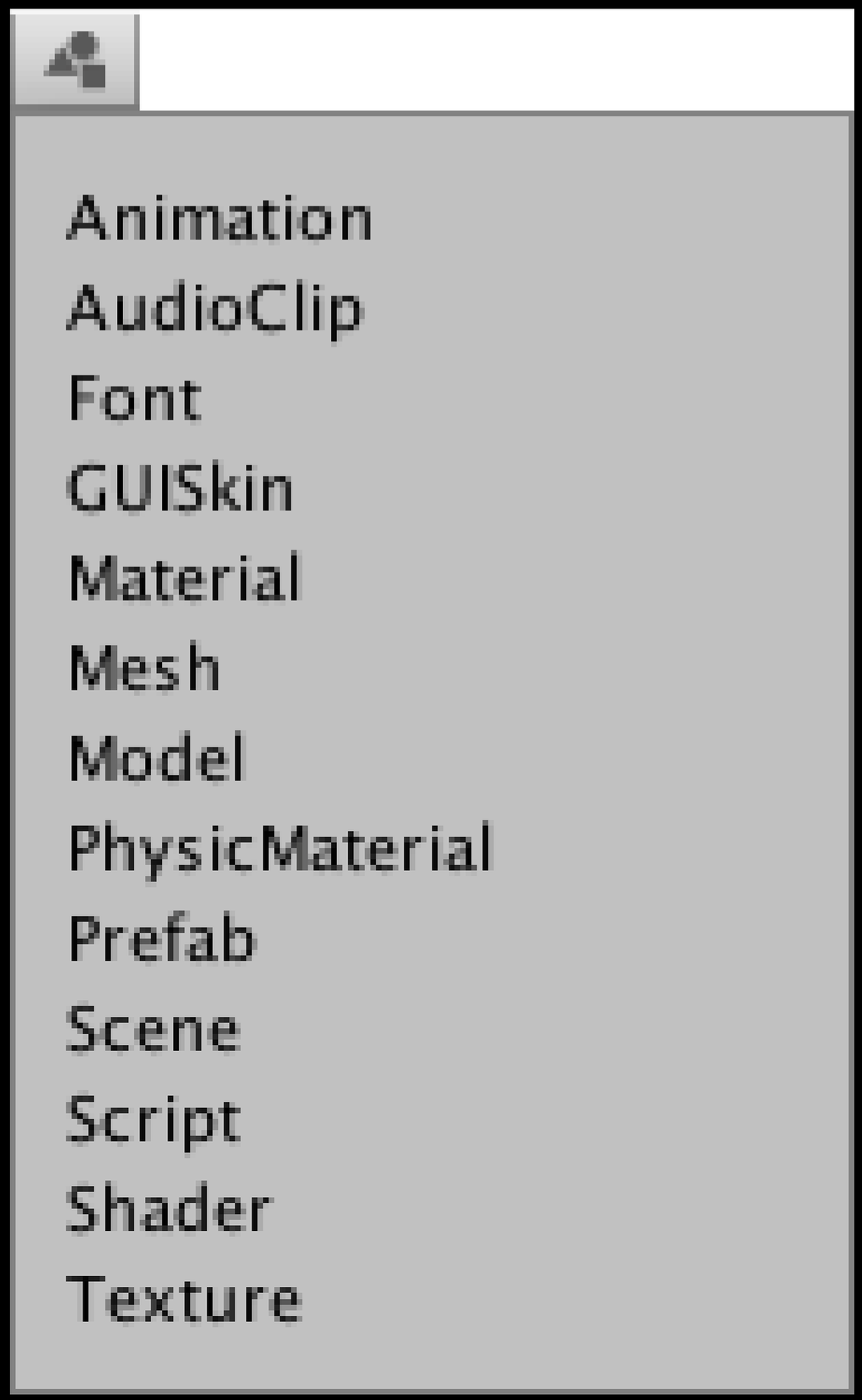
Inspector

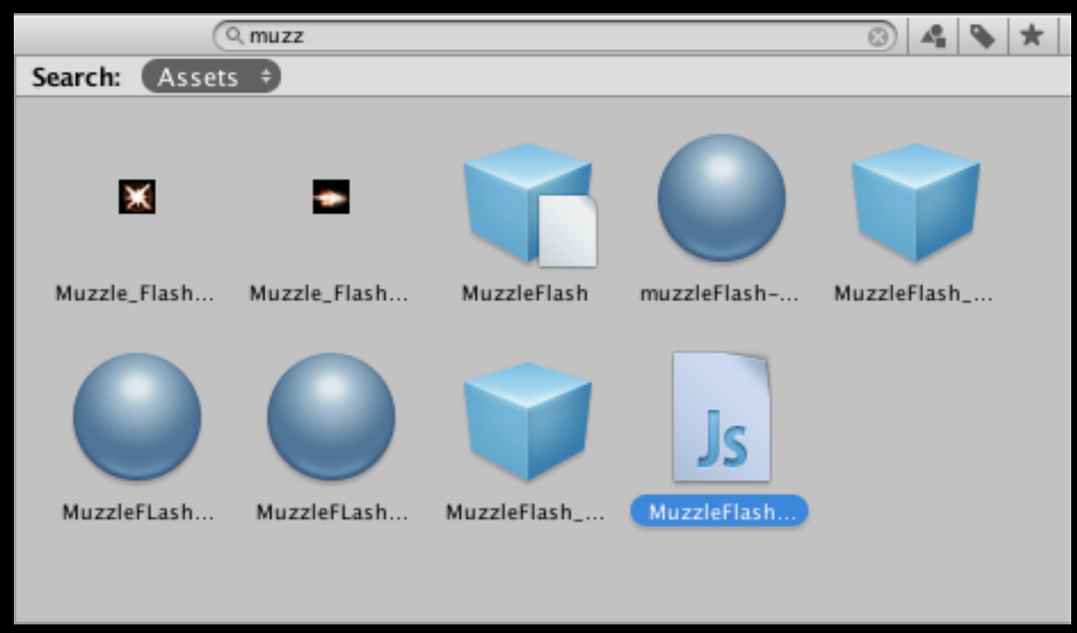
윈도우 추가 및 고정 (동영상)

Project

- 프로젝트에 있는 모든 애셋들을 본다
- 새로운 폴더 생성
  - 공란에 마우스 오른쪽
  - 좌측 상단에 Create 버튼
- 검색
  - 우측 상단에 검색 가능
  - Type 별로 정렬 검색 가능



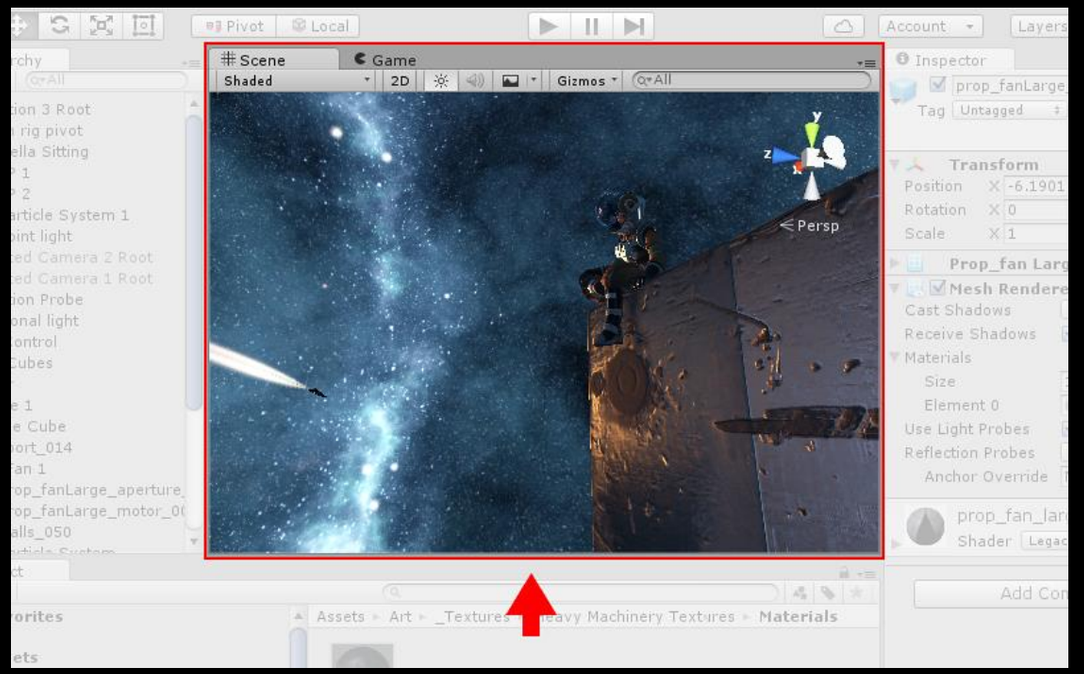

Animation
AudioClip
Font
GUISkin
Material
Mesh
Model
PhysicMaterial
Prefab
Scene
Script
Shader
Texture


muzz
Search:
Assets
Muzzle_Flash...
Muzzle_Flash...
MuzzleFlash
muzzleFlash-...
MuzzleFlash_...
MuzzleFLash...
MuzzleFLash...
MuzzleFlash_...
Js
MuzzleFlash...

Scene

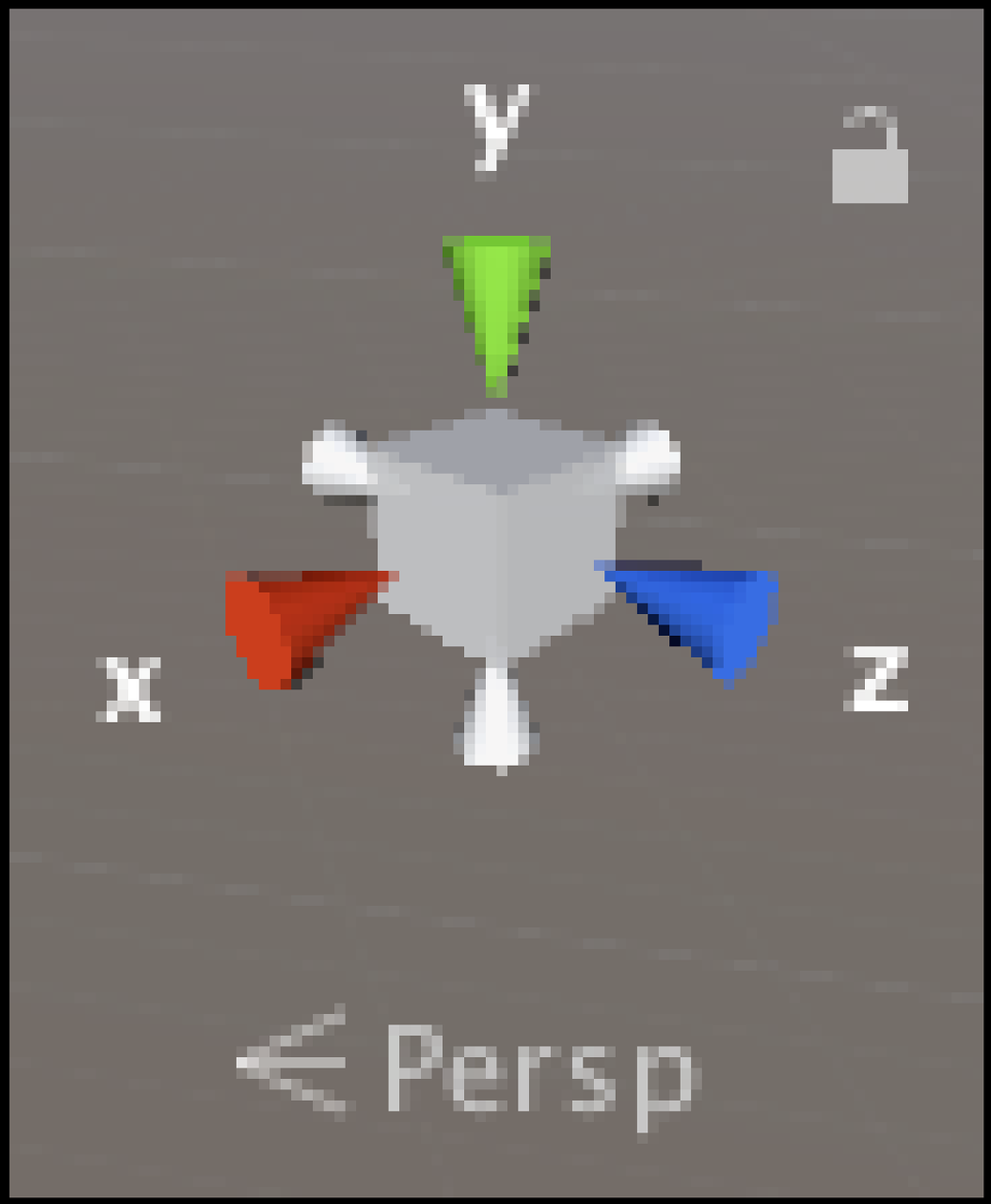
- 현재 만들고 있는 게임 월드를 편집하는 공간

- 캐릭터, 카메라, 라이트, 그외 모든 타입의 게임 오브젝트를 선택하고 위치를 편집 가능

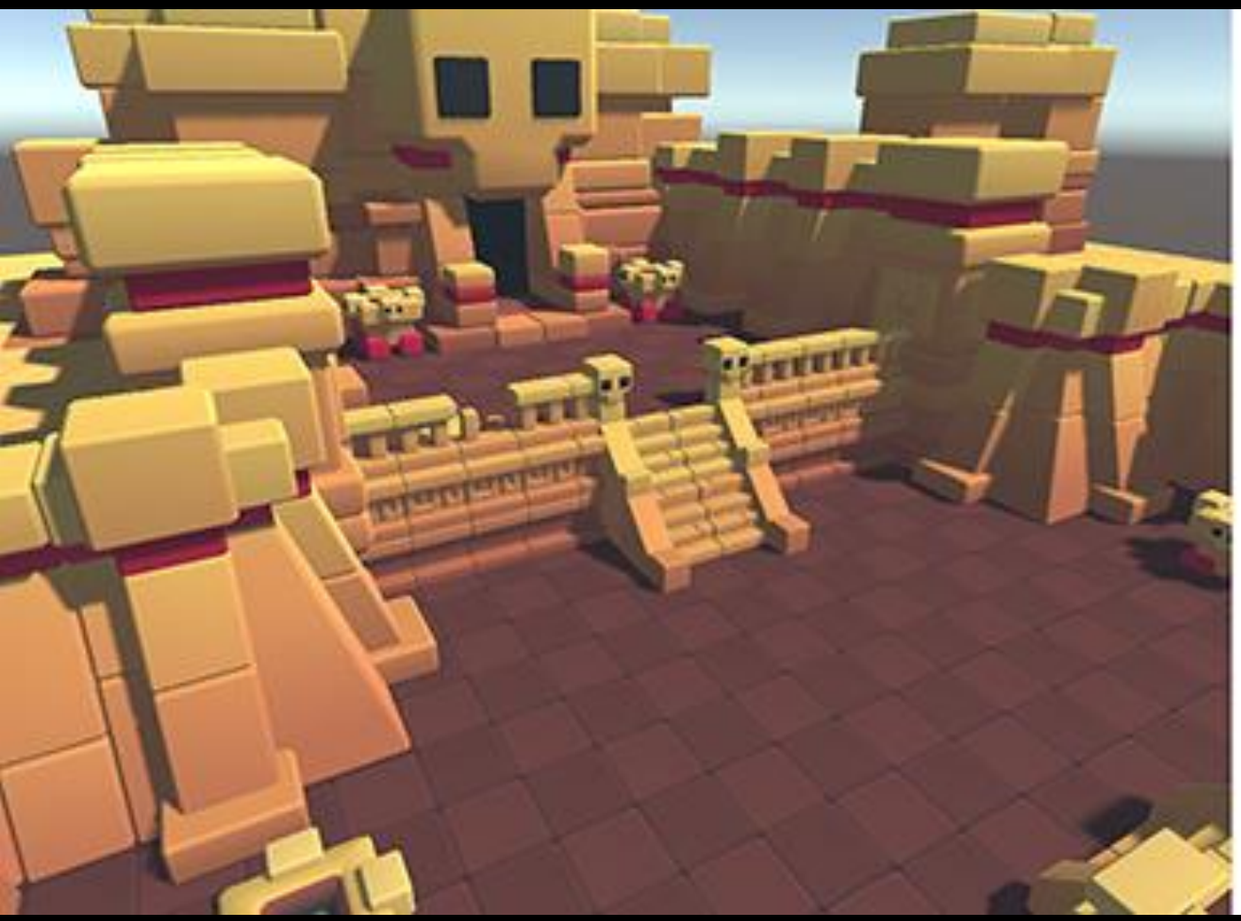


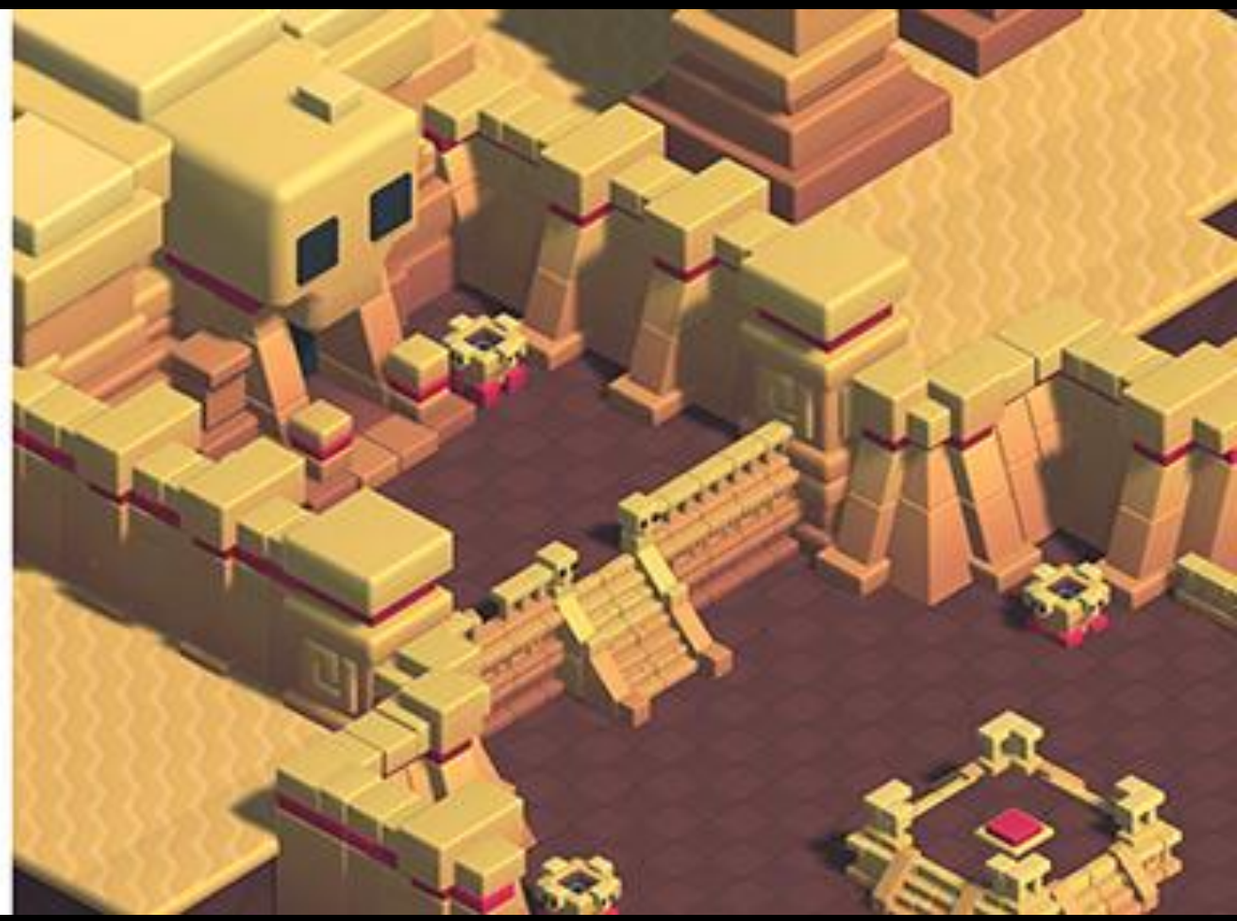
Gizmo

- 오른쪽 상단에 존재
- 현재 씬 뷰 카메라의 회전을 지정
- 뷰의 회전과 프로젝션(투영)모드를 바로 지정 가능
- X, Y, Z 컬러 물체를 누르면 해당 축을 기준으로 카메라가 바로 정렬
- 큐브를 누르면 투영 모드를 전환
  - Isometric(=Orthographic) 뷰
    - 한글로 등측뷰라 읽기도 함
  - Perpective 뷰



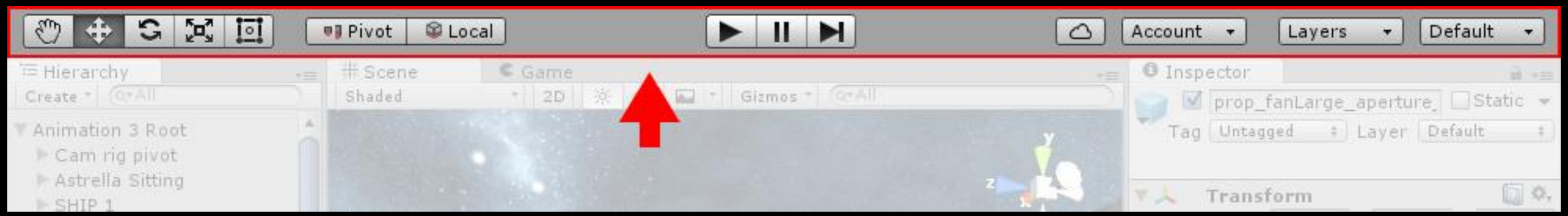
Perspective VS Isometric





Perspective VS Isometric

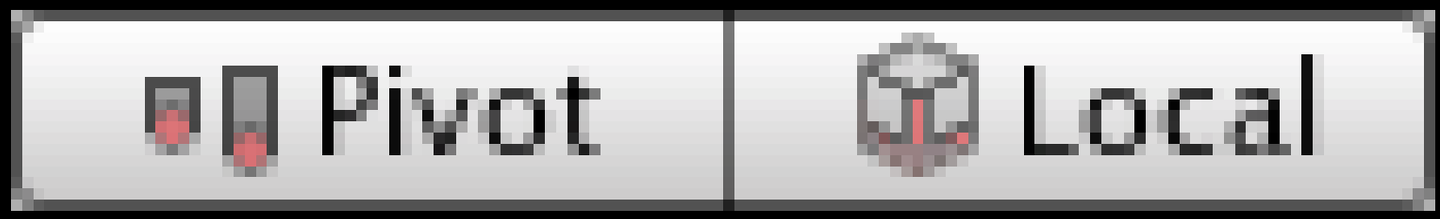
Toolbar


Pivot
Local
Account
Layers
Default
Hierarchy
Create
Animation 3 Root
Cam rig pivot
Astrella Sitting
SHIP 1
Scene
Game
Shaded
2D
Gizmos
Inspector
prop_fanLarge_aperture_
Static
Tag
Untagged
Layer
Default
Transform

Toolbar

- Transform Tools 변환 도구
  - 단축키 (순서대로) QWERT
  - 핸드툴
  - 평행이동
  - 회전
  - 스케일 (크기)
  - Rect 툴

Toolbar



- 트랜스폼 기즈모 토글
  - 피벗 | 센터: 씬상에 표시되는 오브젝트의 위치
    - 피벗: 오브젝트의 배치 기준점을 표시
    - 센터: 오브젝트의 중심을 표시
  - 로컬 or 글로벌: 오브젝트의 좌표계(회전축) 를 씬상에 어떻게 띄울지 결정
    - 로컬: 부모 좌표계와 스스로의 회전을 기준으로 평행 이동
    - 글로벌: 스스로가 얼마나 회전했는지 상관 없이,
      글로벌 좌표계를 기준으로 평행이동

Toolbar

- 시작
- 일시정지
- 다음 프레임으로 스킵

- 클라우드 버튼
- 유니티 서비스 윈도우 열기

유니티 에디터 편의기능

- 일인칭으로 씬을 돌아다니기
- Flythrough 모드
  - 가속(뛰기)키
- 씬 카메라 포커스
- 공전(Orbit) 모드
- 핸드 툴 없이 씬 카메라 움직이기

Employment

Preparedness

Collaboration Skills

- 비디오 게임 개발 업계에서 "비평" 의 의미란?

Employment Responsibilities

- NDA – Non-Disclosure Agreement (기밀 유지 협약)의 의미와 목적

- IP – Intellectual Property (지적 재산권) 의 의미

Game Art Principles

Character Design

- NPC – (Non-Player Characters) 의 최적화 방법

Concept Design

- 컬러 팔레트 – 아트에 의해 게임 분위기가 바뀌는 방법
- Look and Feel – 컨셉 아트의 목적

Environment Design

- Color Palette 컬러 파레트 – "일관된 색감"의 개념과 목적

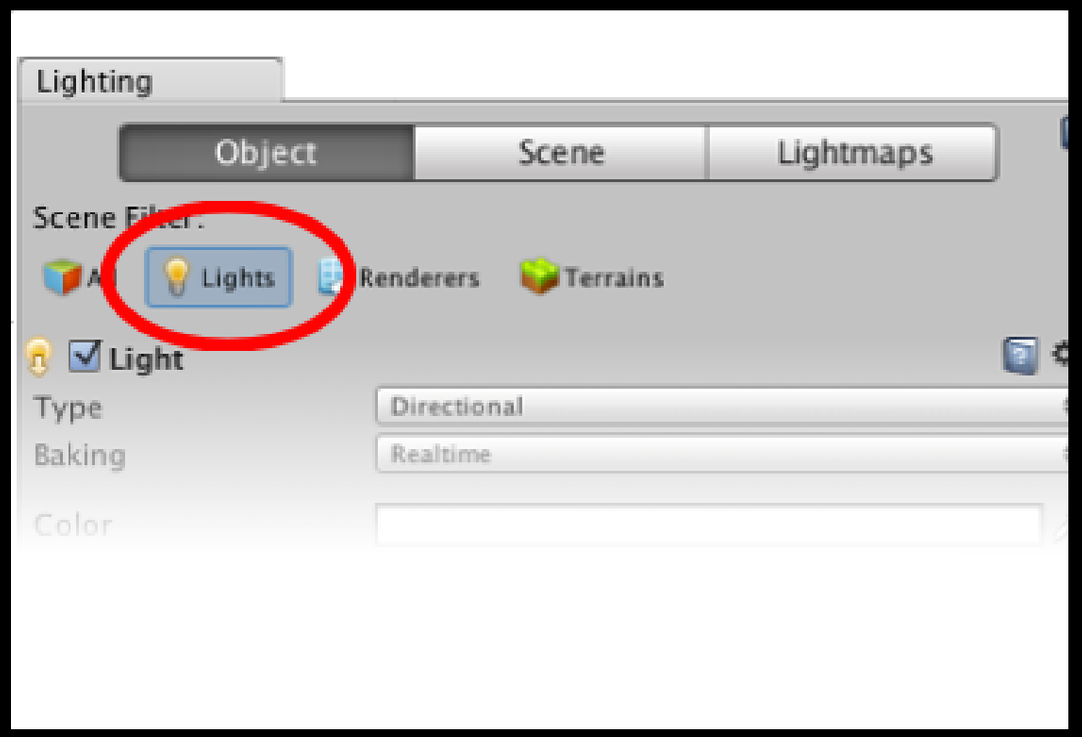
Lighting

Global Illumination (GI)

- GI - 전역 조명
  - 광원에서 직접 오는 빛 뿐만 아니라 다른 오브젝트에서 반사된 빛 까지 표면에 표현하는 시스템
  - 유니티에서 GI는 static 오브젝트에서만 가능
- Baked GI
  - Baked Lightmap 이라고도 함
  - 빌드 할때 오브젝트 위에을 굽기
  - 런타임 퍼포먼스가 더 낫다
- Precompute GI
  - 런타임 도중에 라이팅을 실시간으로 구음
  - 굽는 시간이 소요되지 않아 빌드 시간이 적다

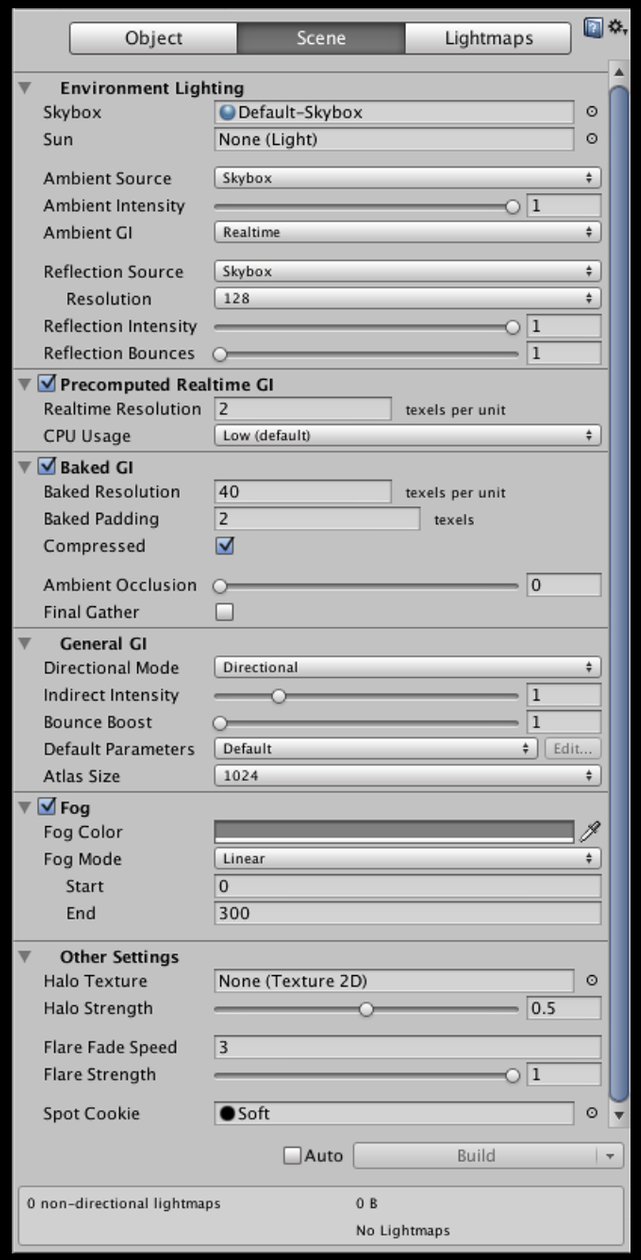
Light Setting

- Light 윈도우 열기
  - Window > Light > Settings
  - 5.5 와 5.6 이 진입 방법이 다름
    - 첨부된 이미지는 5.5 기준
- Object
  - 씬 상에 모든 Light 관련 오브젝트를 보여준다



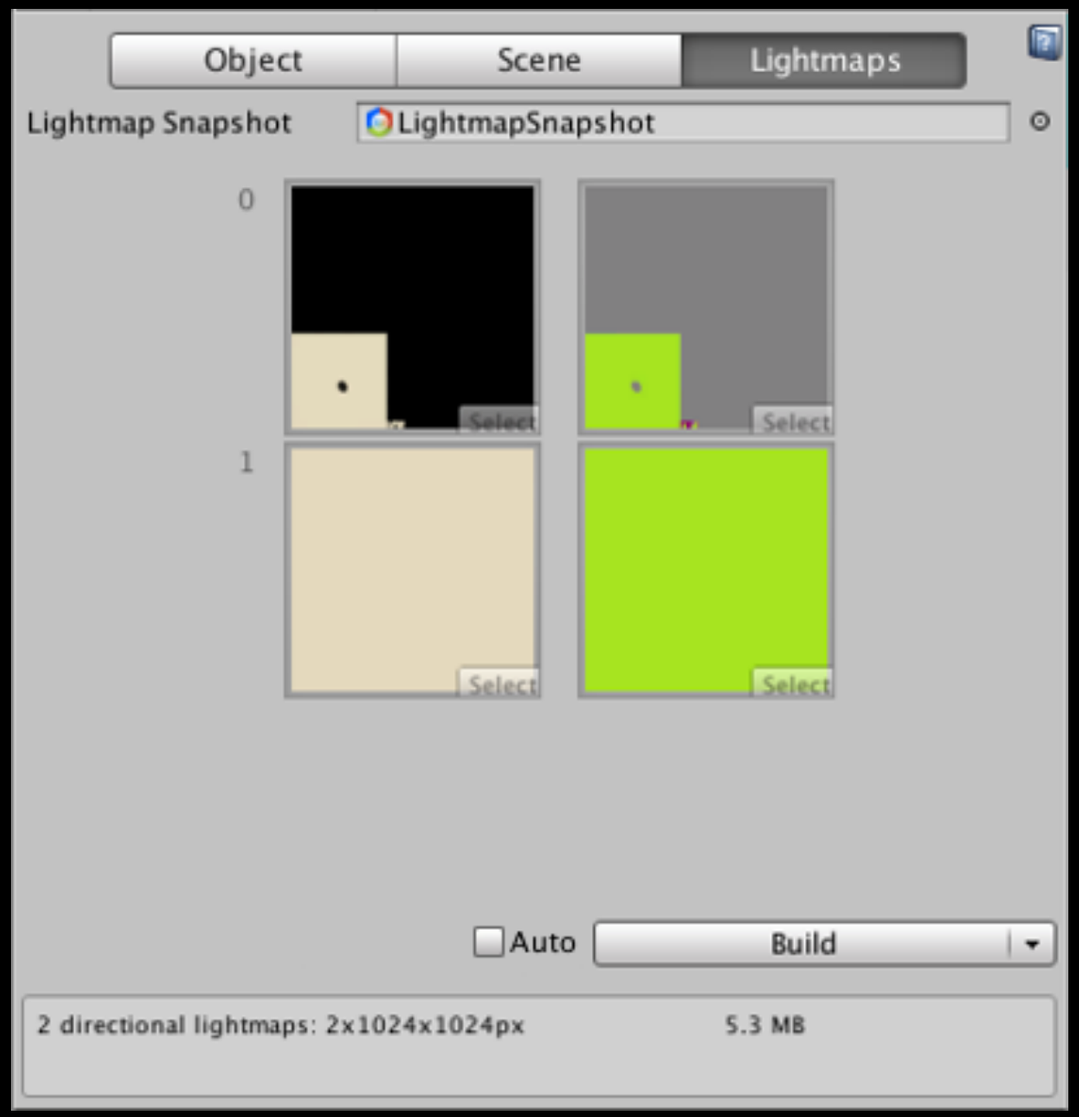
Light Setting

- Scene
  - 현재 씬 상의 GI를 편집
  - Precomputed Realtime GI
    - 활성화 옵션 및 해상도
- Baked GI
  - 활성화 옵션 및 해상도
- Ambient Source
  - GI 의 광원 (Skybox, Gradient, Color 지정가능)



Light Setting

- Lightmaps
  - GI 처리를 통해서 씬의 라이팅을 구운 라이트 맵 애셋을 지정하는 곳
- 굽기
  - 라이팅을 오브젝트 표면 위에 미리 굽는 처리
    - 런타임 퍼포먼스를 아낀다
  - 라이팅 윈도우의 하단에 위치
  - Auto – 자동으로 백그라운드에서 굽기
  - Build – 수동으로 굽기



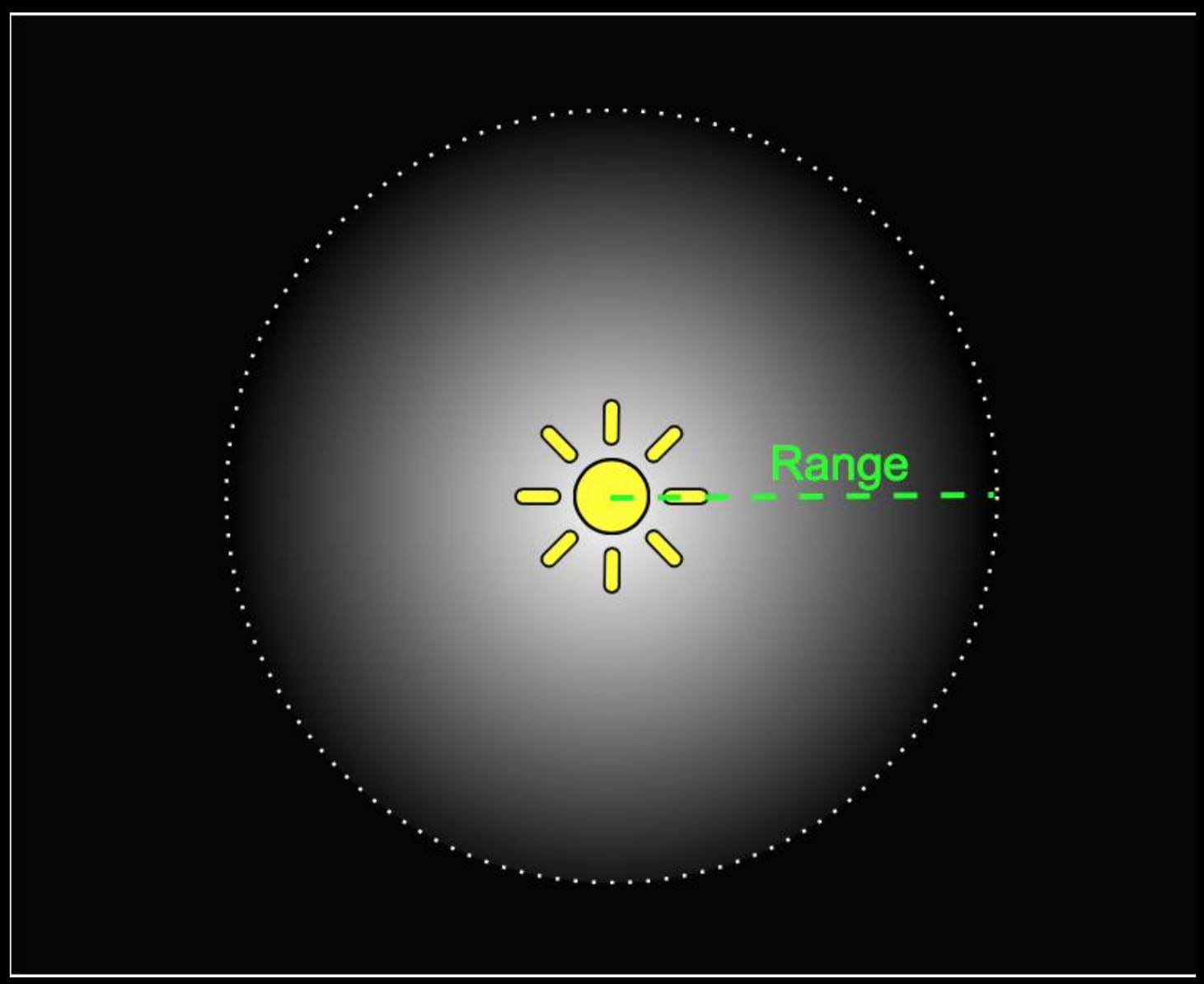
Shadow Type

- No Shadow
- Hard Shadow - 직사 그림자
- Soft Shadow - 부드러운 그림자

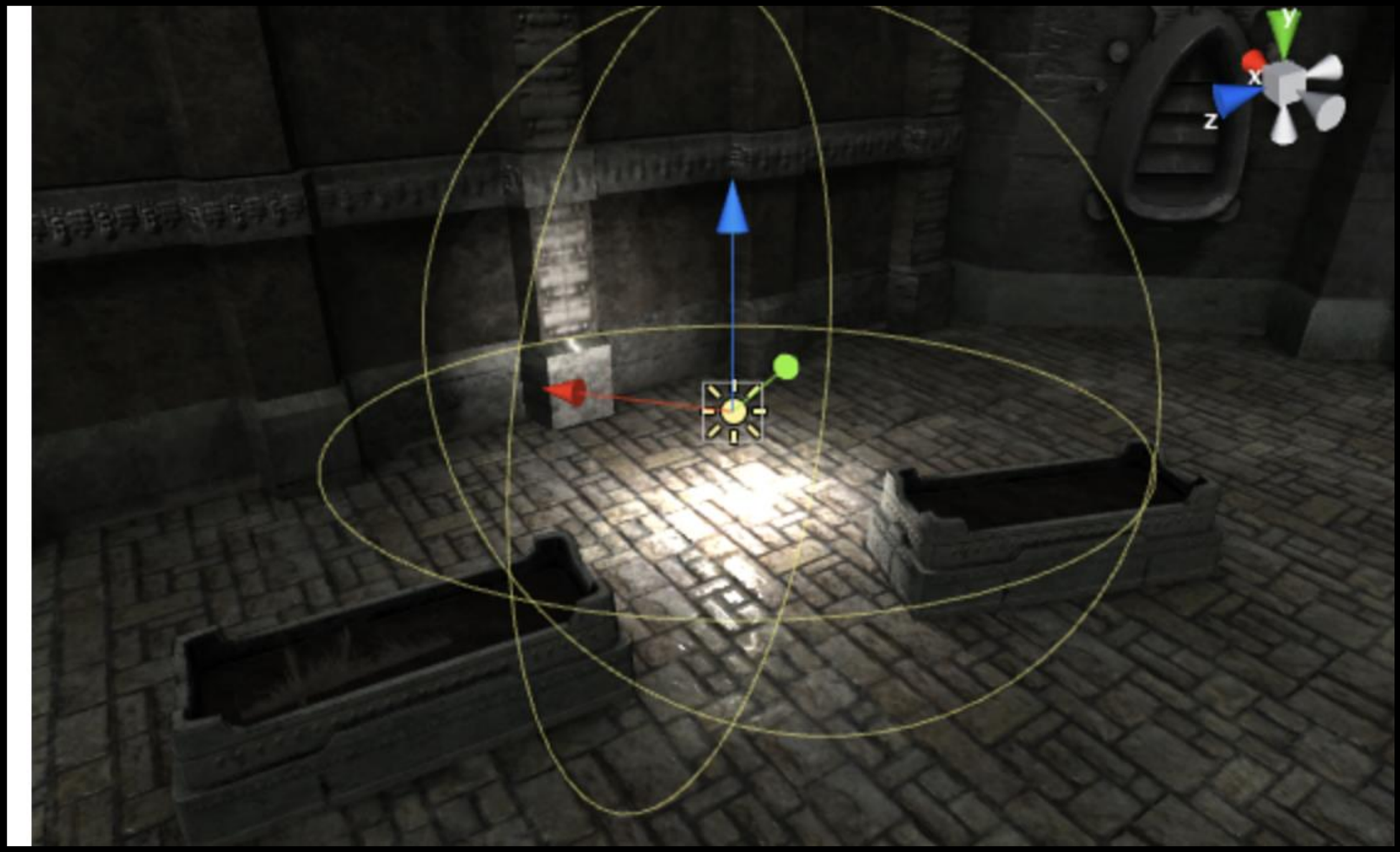
- Strength - 그림자의 세기
- Resolution: 그림자의 해상도(품질)
- Bias: 그림자의 에러 보정
- Softness: 그림자를 부드럽게할 가장자리 영역 크기

Lighting Type

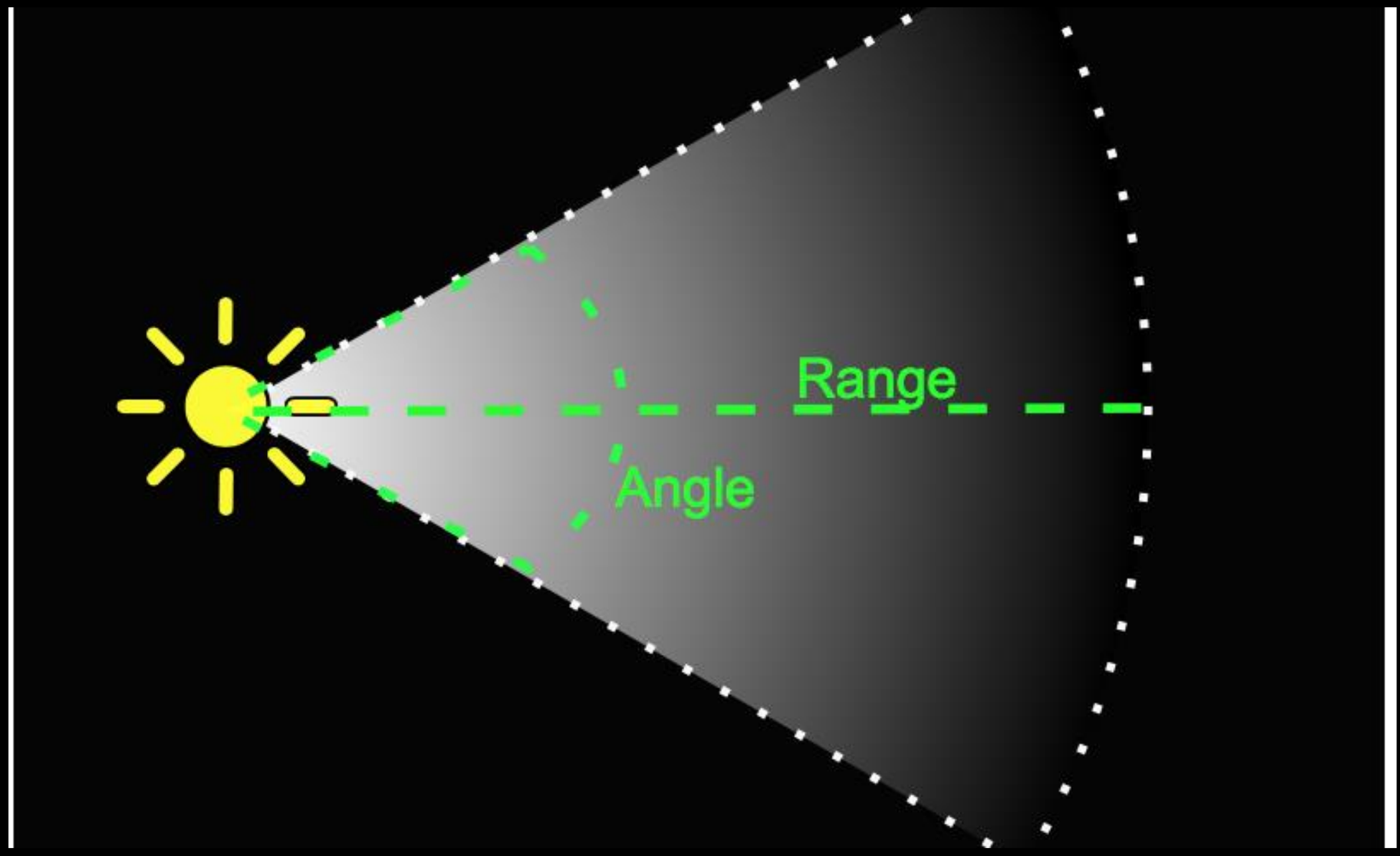
- Spot – 원뿔로 퍼지는 가로등
- Directional – 위치 상관없이 전체에 적용되는 직사광
- Point – 모든 방향으로 구를 그리며 퍼짐
- Area – 2차원 사각형을 기준으로 라이트 적용



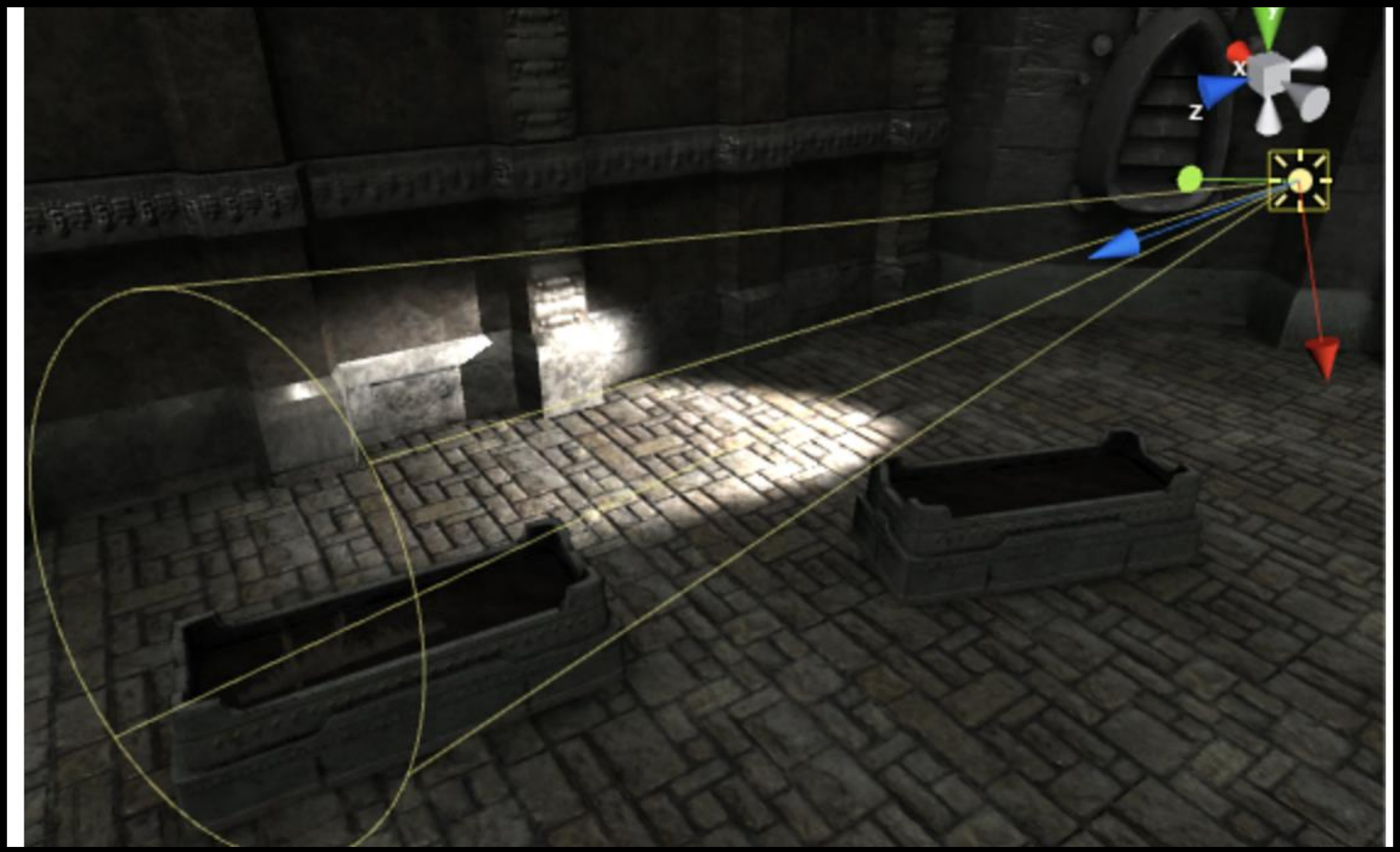
Point



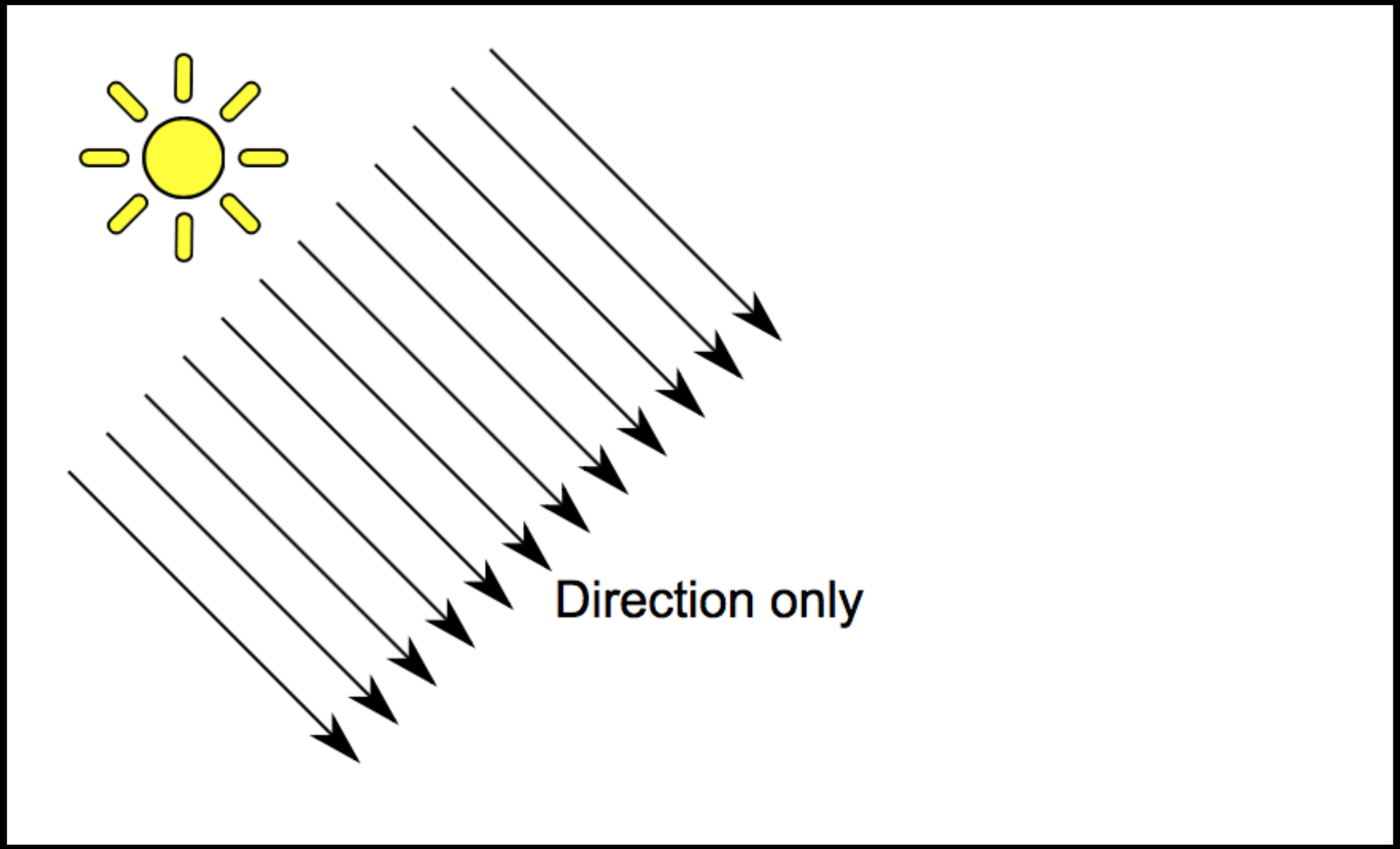
Point



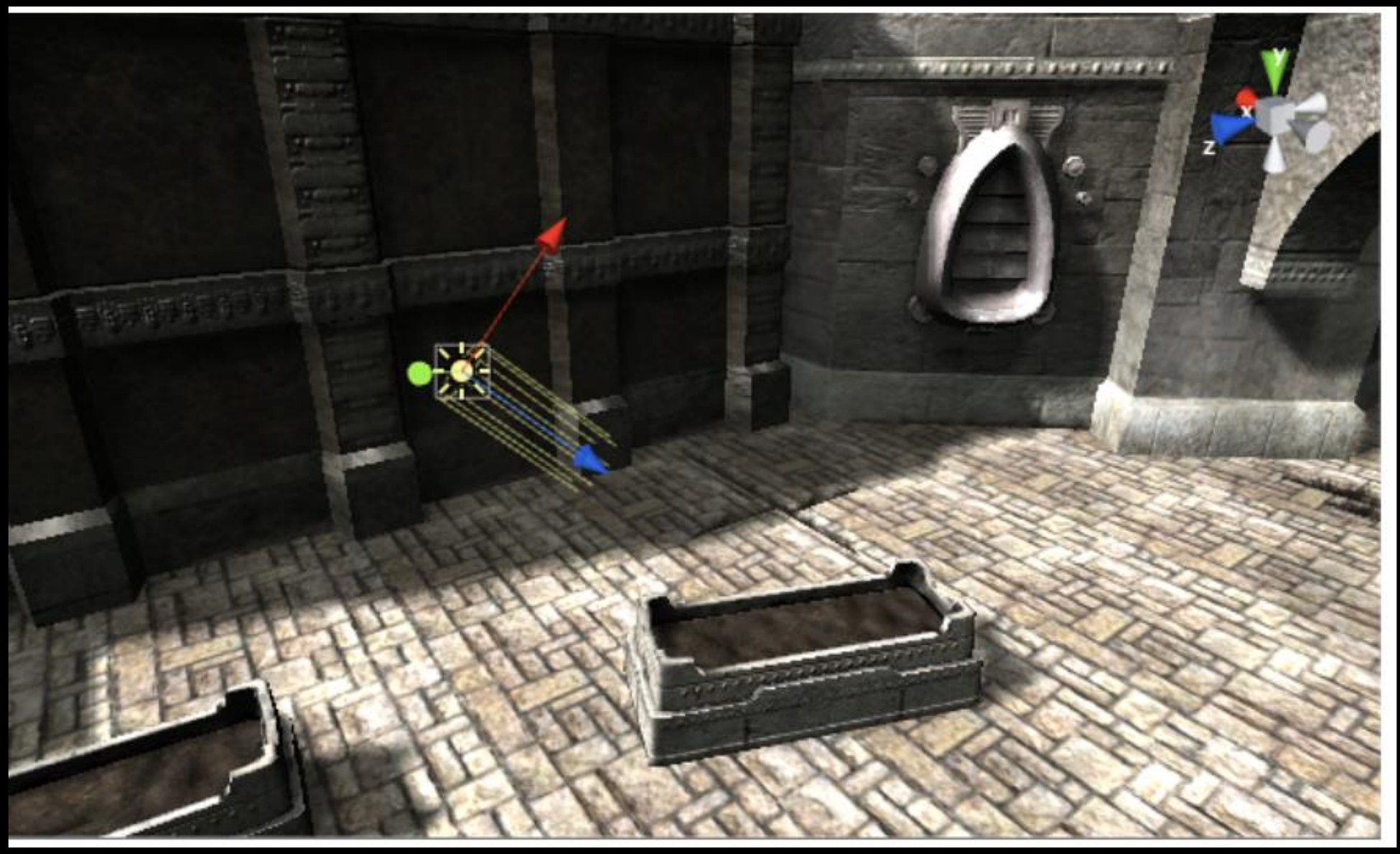
Spot



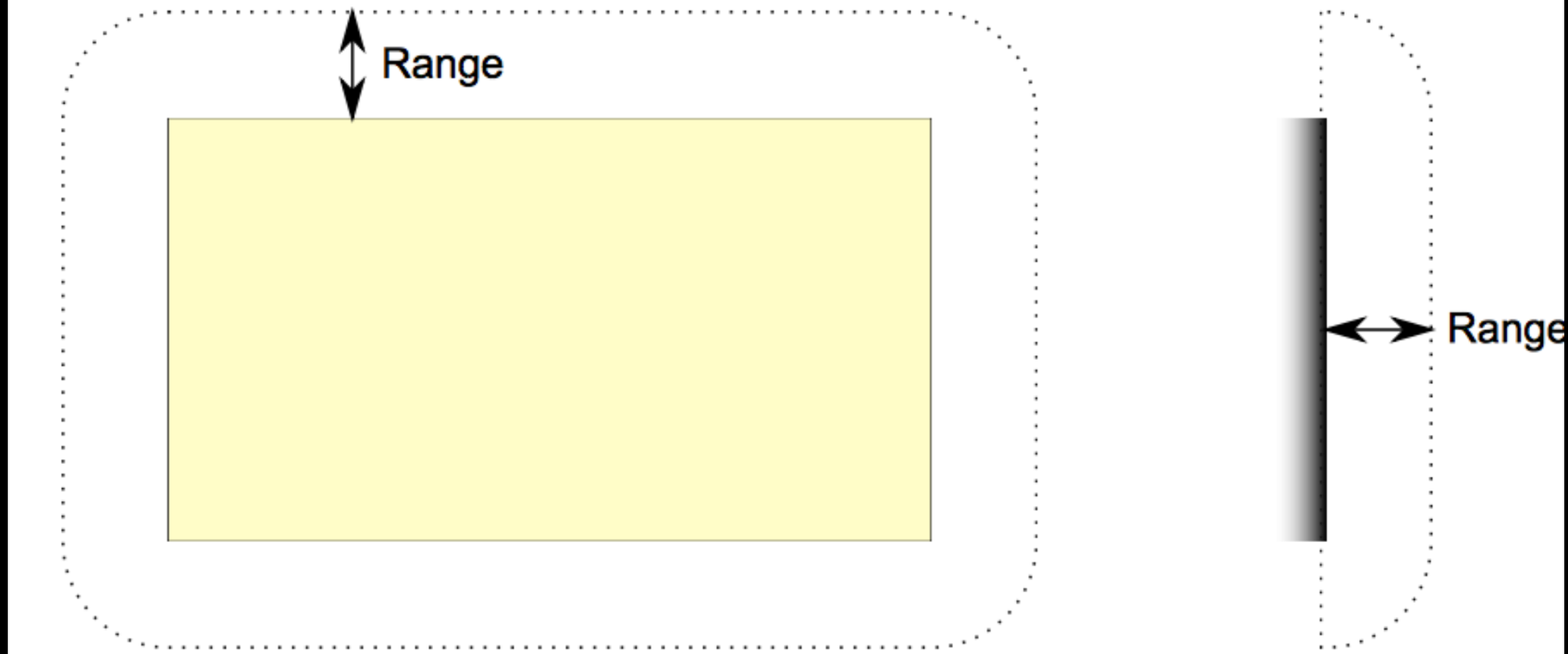
Spot



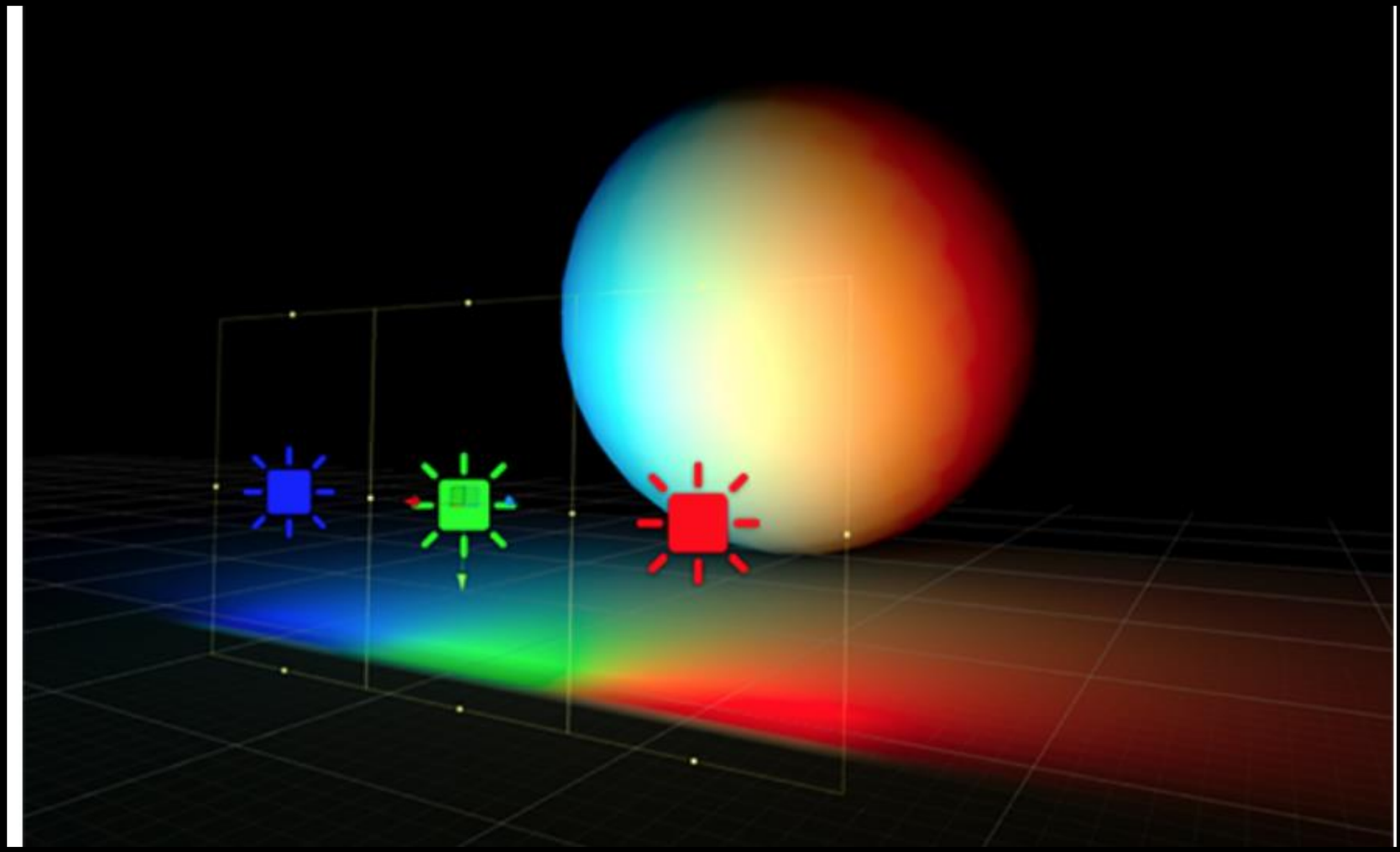
Directional



Directional



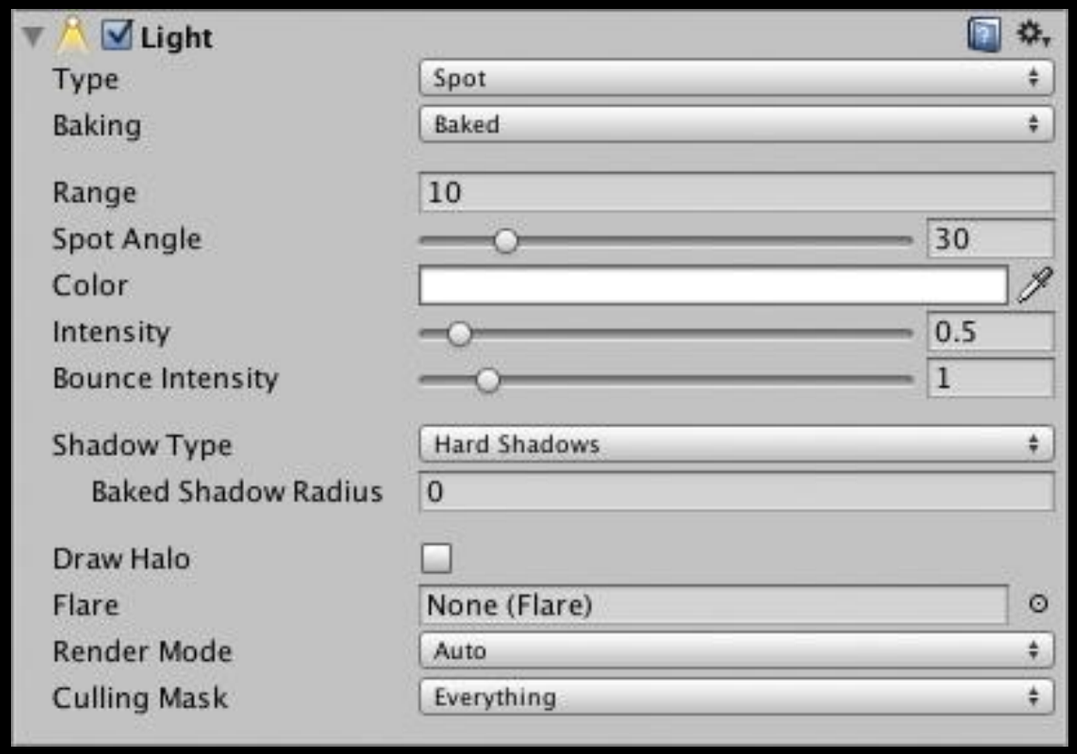
Area



Area

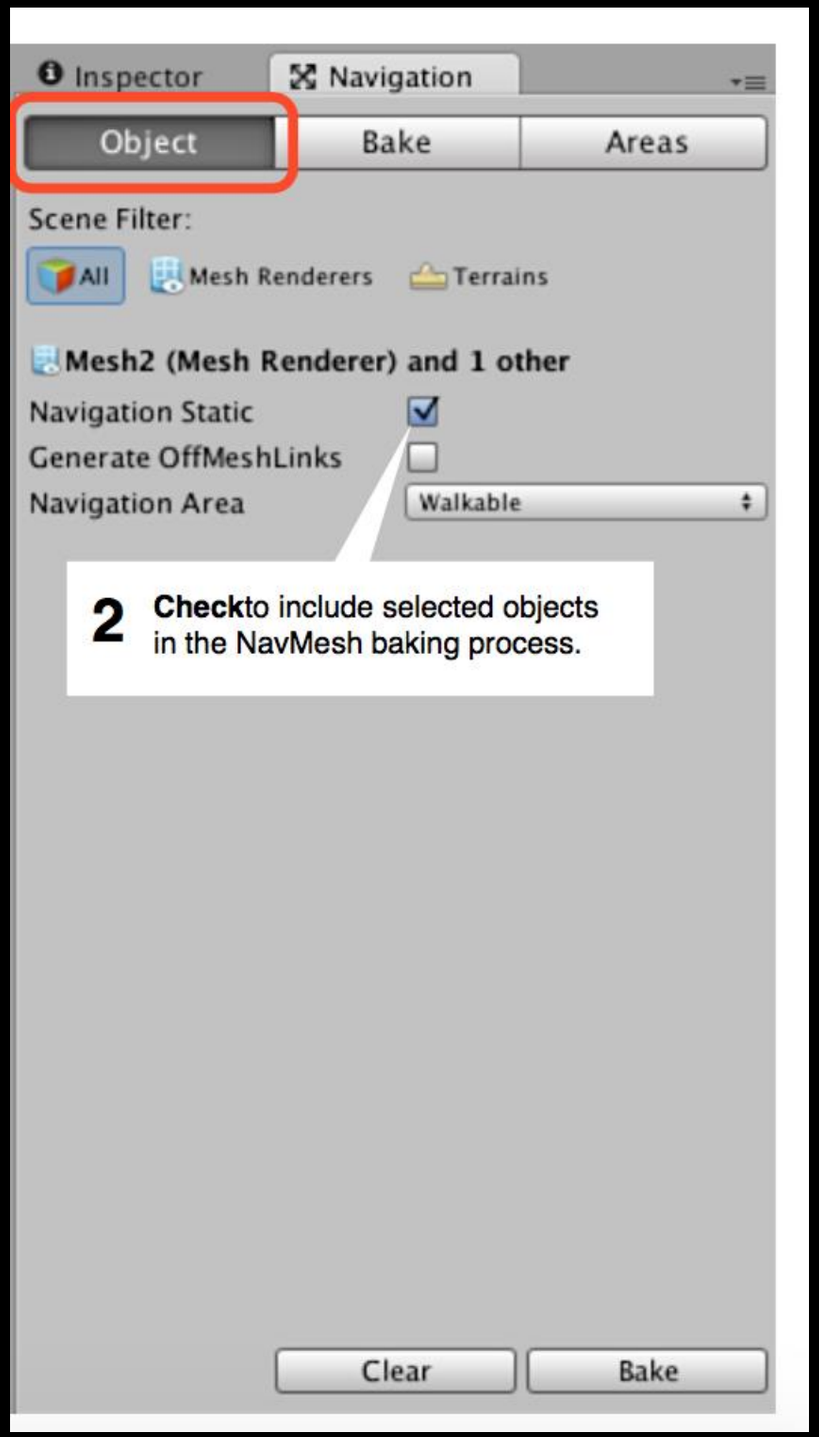
Light Property

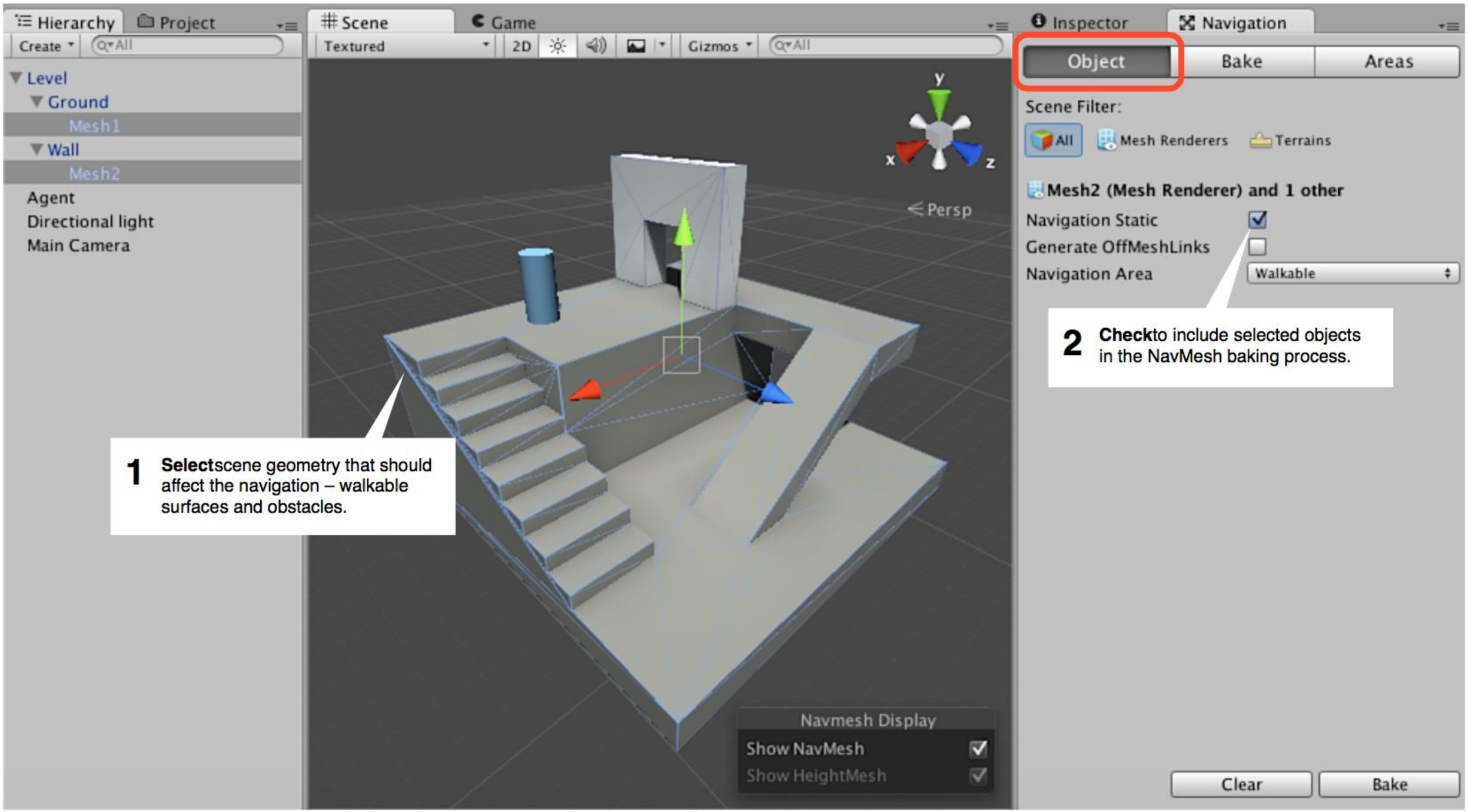
- Intensity: 빛의 강도
- Culling Mask: 빛의 영향을 받는 물체를 필터링 하여 지정 (레이어 마스크)



Navigation System

- Navigation System
  - Window > Navigation 으로 윈도우 열기
  - 콜라이더가 붙은 오브젝트를 Navigation Static 으로 지정후 Bake
    - 런타임 도중 수정 불가
- 장애물
  - Nav Mesh Obstacle 컴포넌트 추가
  - Navigation Static 으로 지정
- AI
  - Nav Mesh Agent 컴포넌트 추가

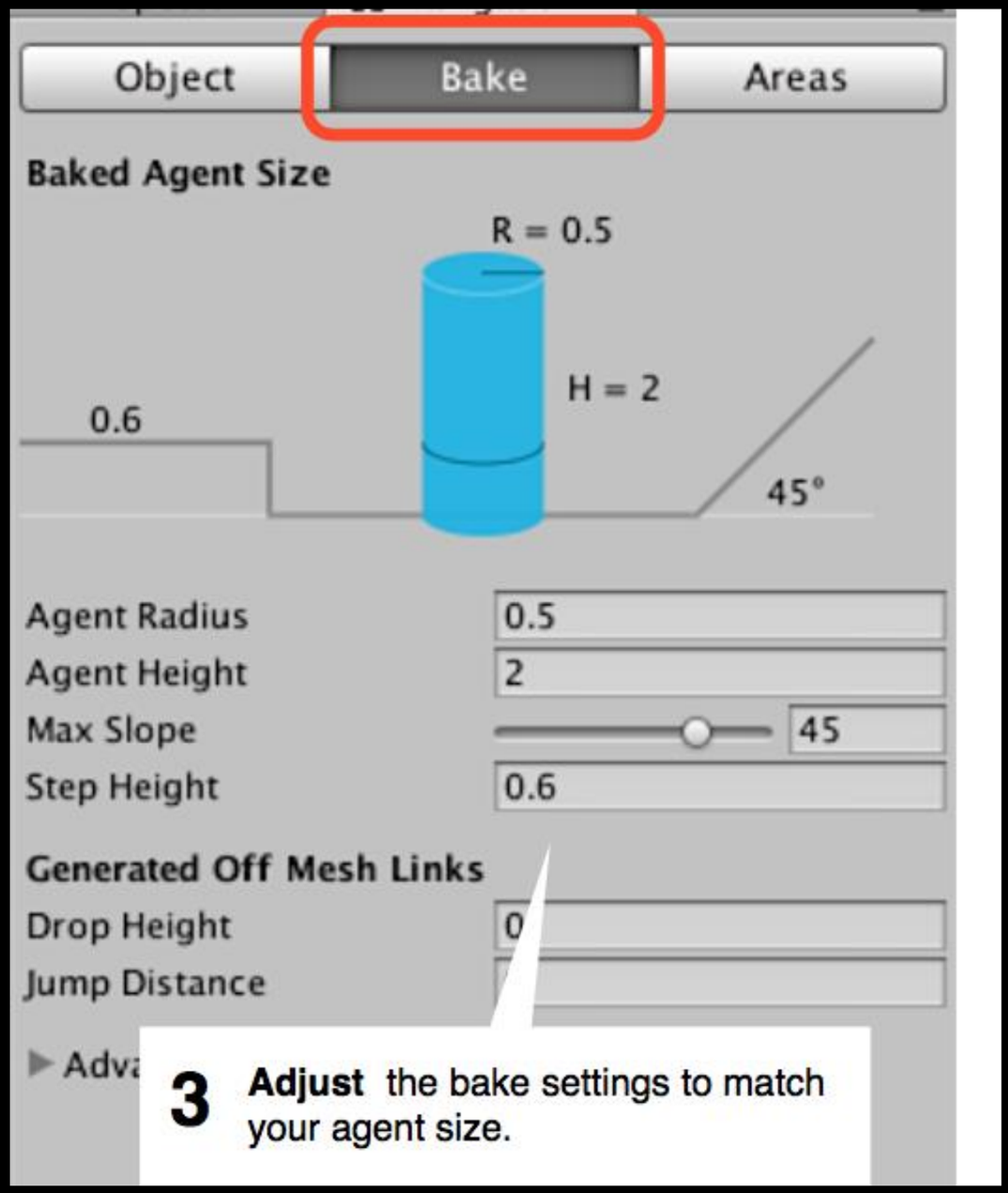



Hierarchy
Project
Create
All
Level
Ground
Mesh1
Wall
Mesh2
Agent
Directional light
Main Camera
Scene
Game
Textured
2D
Gizmos
All
y
x
z
Persp
Inspector
Navigation
Object
Bake
Areas
Scene Filter:
All
Mesh Renderers
Terrains
Mesh2 (Mesh Renderer) and 1 other
Navigation Static
Generate OffMeshLinks
Navigation Area
Walkable
2 Checkto include selected objects in the NavMesh baking process.
1 Selectscene geometry that should affect the navigation – walkable surfaces and obstacles.
Navmesh Display
Show NavMesh
Show HeightMesh
Clear
Bake

Hierarchy
Project
Create
All
Level
Ground
Mesh1
Wall
Mesh2
Agent
Directional light
Main Camera
Scene
Game
Textured
2D
Gizmos
All
y
x
z
Persp
Heightdefines how low spaces the agent can reach.
Step Heightdefines how high obstructions the agent can step on.
Max Slopedefines how steep ramps the agent walk up.
Radiusdefines how close the agent center can get to a wall or a ledge.
Navmesh Display
Show NavMesh
Show HeightMesh
Inspector
Navigation
Object
Bake
Areas
Baked Agent Size
R = 0.5
H = 2
0.6
45°
Agent Radius 0.5
Agent Height 2
Max Slope 45
Step Height 0.6
Generated Off Mesh Links
Drop Height 0
Jump Distance
Adva
3 **Adjust** the bake settings to match your agent size.
4 **Click bake**to build the navmesh.
Clear
Bake

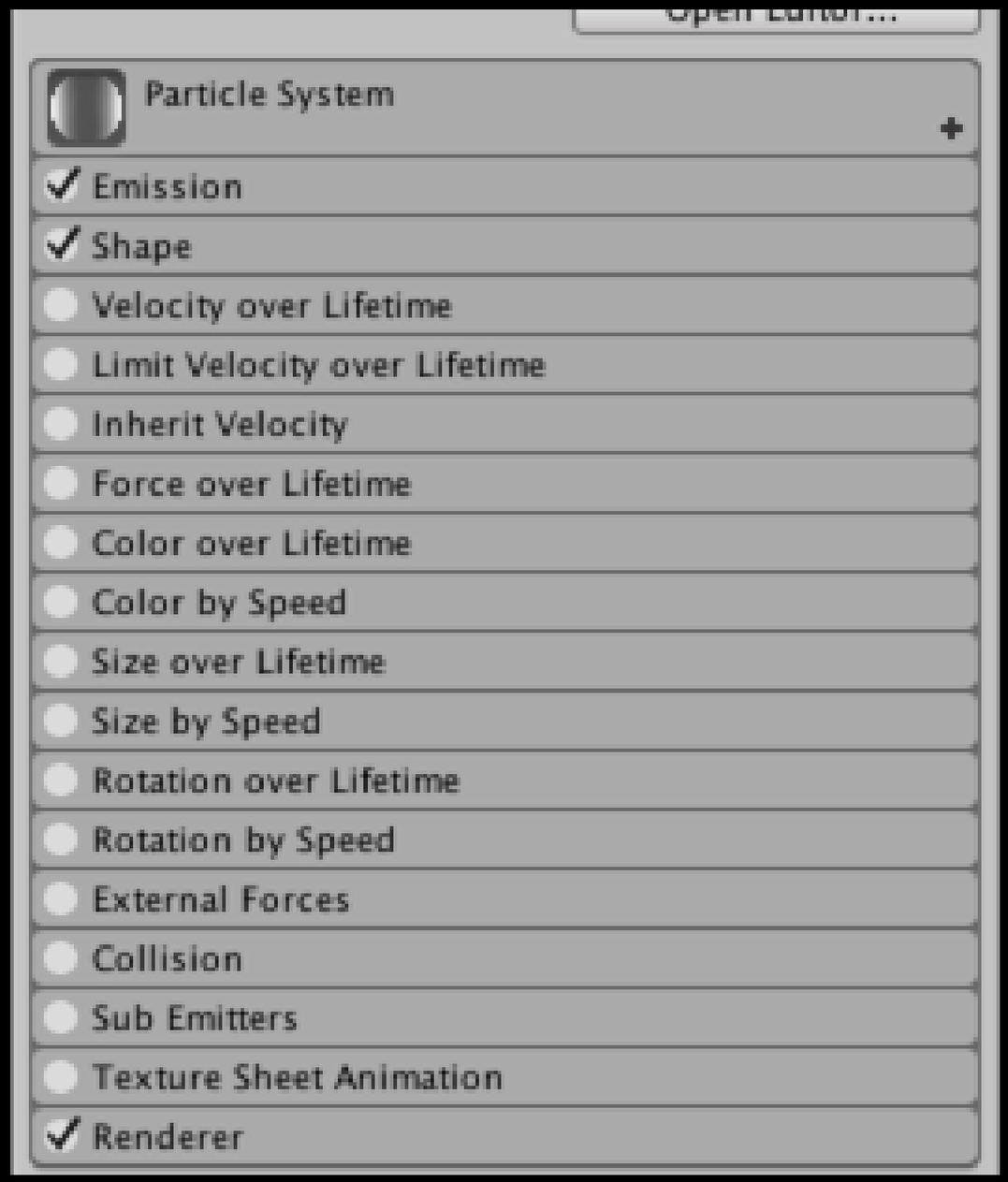
Baked Agent

- Radius 를 키울 수록 뚱뚱해져서 다닐수 있는 영역이 한정됨
- Radius 를 줄이면 좁은 곳도 구워지게 됨
- Max Slope 을 늘리면 이동 가능한 경사가 커짐



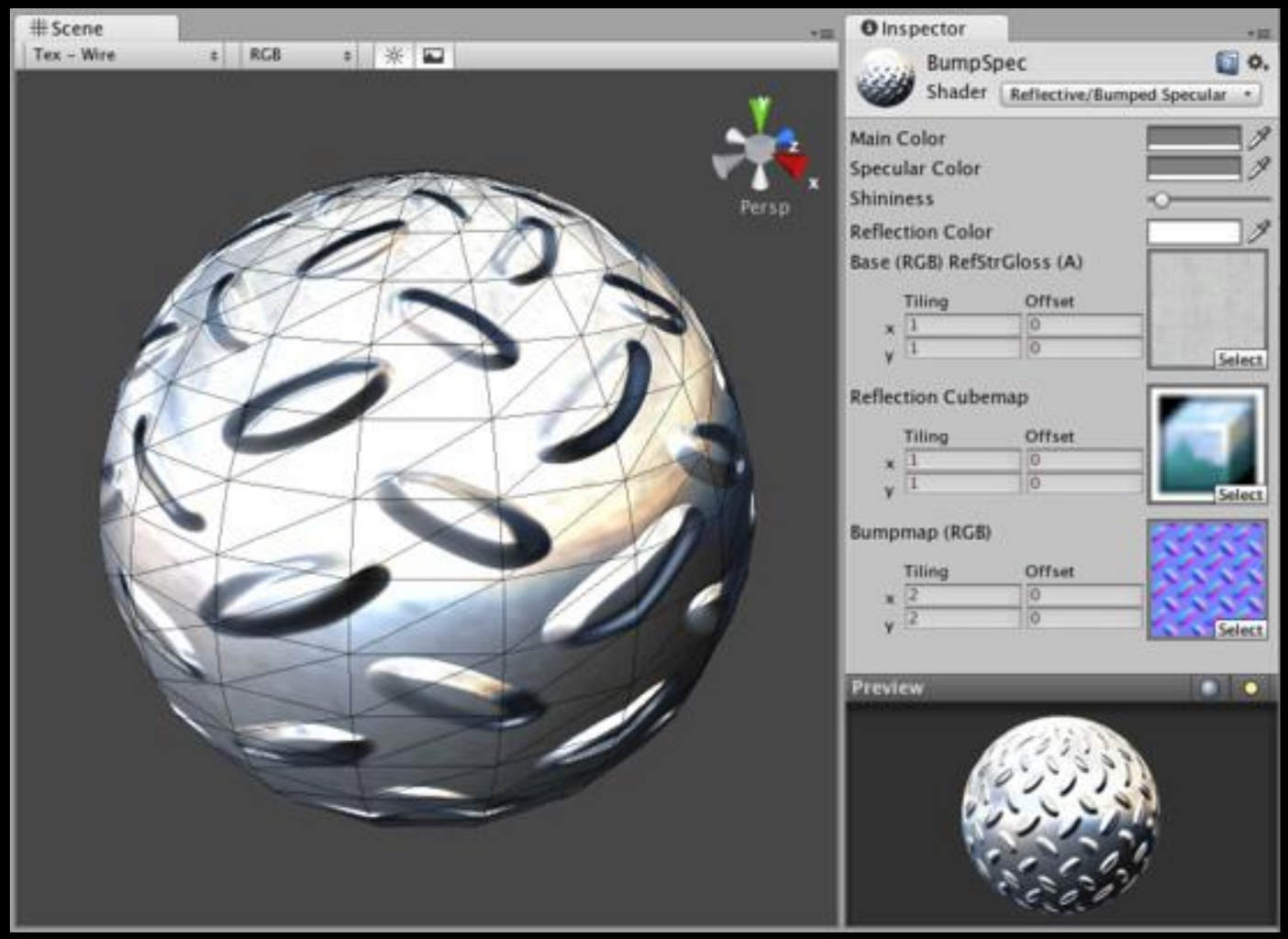
Particle System

- 2D Sprite 이미지를 생성해서 뿌림
- Emission
  - Rate Over Time : 시간 비례
  - Rate over Distance 거리 비례
- Shape
  - 파티클이 생성되는 영역 형태
  - Cone, Box, Sphere
- Velocity over Time: 속도



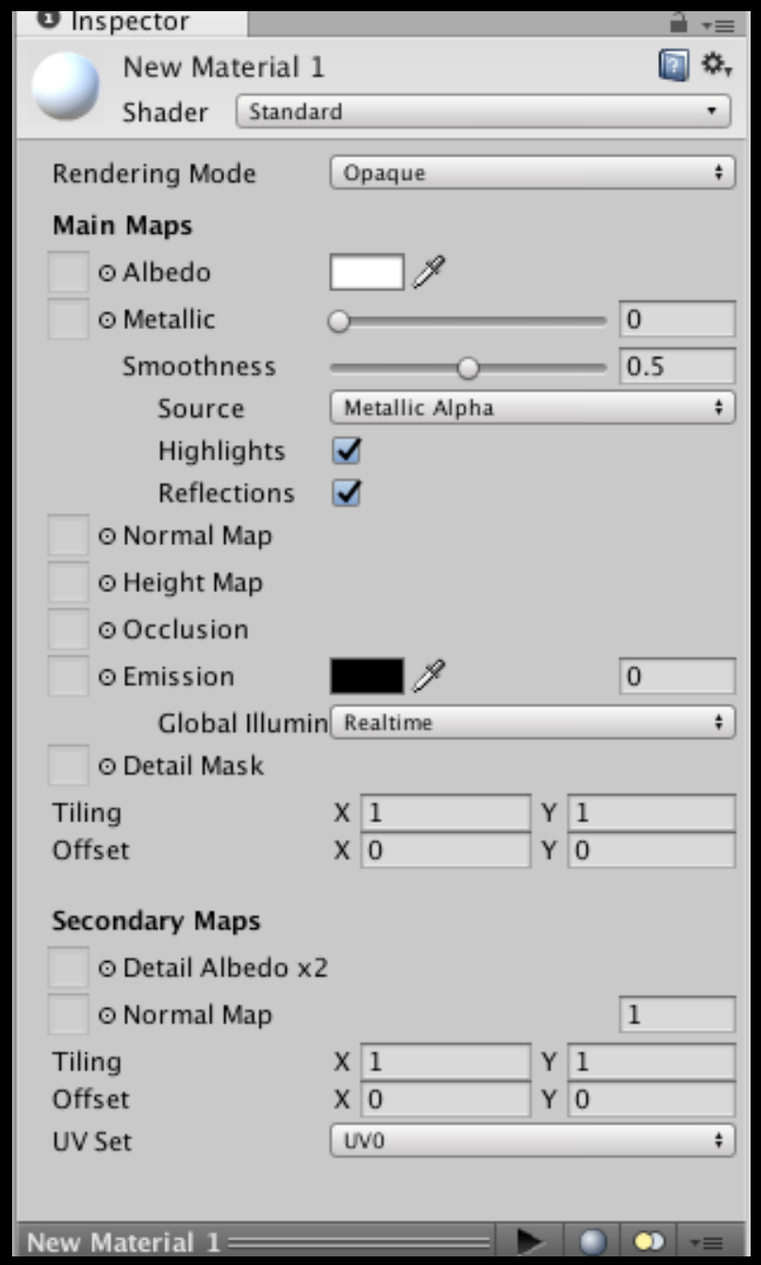
Material

- 쉽게 말해,
  텍스쳐 + 쉐이더
- 오브젝트의 각 픽셀의 색깔을 결정해주는 애셋



Material 필드

- 쉐이더
  - 빛에 의해 픽셀이 어떻게 보일 것인가
- 텍스쳐
  - 색을 결정하는 이미지
- Standard 쉐이더 기준으로 설명
- Albedo – 기본 색을 지정
- Metallic – 물체가 얼마나 금속성 표면을 가지는지



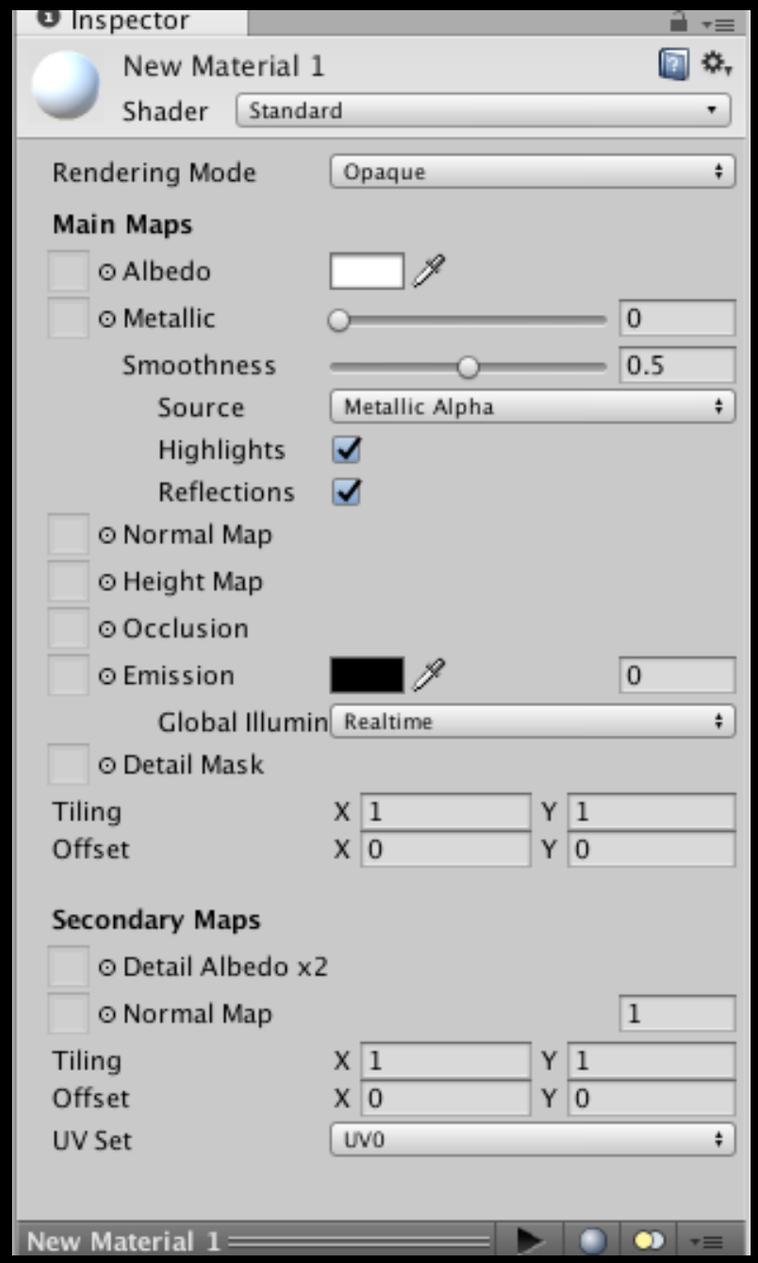
Ambient

Ambient
+ Diffuse

Ambient
+ Diffuse
+ Specular

Material 필드

- Smoothness
  - 표면이 부드러운 정도 = 반사의 정도
  - 값을 높이면
    - 하이라이트 증가
    - 거울 처럼 주변이 비침
- Specular: 직사광
  - 표면에 수직으로 들어와서 반사되는 빛
  - 자동으로 처리되나 수동으로 지정 가능



- 노말맵 등 디테일을 추가해주는 텍스쳐들의 특징:
  - 실시간 연산이 아닌 눈속임
    - 물리적인 연산은 거의 하지 않음
  - 할당하는 맵 텍스쳐는 마스킹을 위해 사용
    - 미리 텍스쳐 이미지에 새겨진 형상에 따라 효과가 적용되는 것
    - 아티스트가 미리 3D 프로그램 등 에서 랜더하여 뽑아냄
- 결론은 아무리 이뻐도 이미테이션(잘 따라한 가짜)

- 노말맵
  - 표면의 디테일을 추가 (적은 비용)
  - 원본인 하이 폴리곤에서의 빛의 반사를 흉내낸 것
    - 원본의 각 꼭지점의 수직(=노말) 벡터를 저장
  - 폴리곤이 적어도, 폴리곤이 많았을때의 빛효과를 흉내 내서 눈의 착각을 줌
- 가까이서 기울여 보면 티남

- 하이트맵
  - 노말맵의 고급 버전
- 노말맵은 빛 효과를 흉내, 하이트맵은 깊이를 흉내
- 기울여서 봐도 깊이가 있는 것 처럼 보임
- 물론 가짜

- 오쿨루전 맵
  - 어떤 부분이 그림자를 약하고 강하게 받을 것인지 지정한 맵
- 간접광들에 의한 그림자 효과가 실시간으로 나는 것 처럼 꾸밈

- 에미션
  - 빛을 내는 것 처럼 꾸밈
- 에미션 마스크
  - 빛이 나는 곳과 나지 말아야 할 곳을 지정한 맵

- 디테일 마스크
  - 디테일을 위한 부가적인 텍스쳐
  - 낮은 해상도의 텍스쳐를 추가로 입혀주는 것
  - 예) 기본 피부 텍스쳐 위에 얼룩 패턴 등을 추가 텍스쳐로 할당

Rendering Mode 랜더 모드

- Opaque
  - 투명도 없음
  - 속이 꽉차있는 일반적인 물체
    - 도자기, 플라스틱 장난감
- Transparent
  - 투명도 있음
  - 불투명한 부분이 있는 일반적인 물체
  - 얼룩이 묻은 유리 창문, 반투명한 얼음

Rendering Mode 랜더 모드

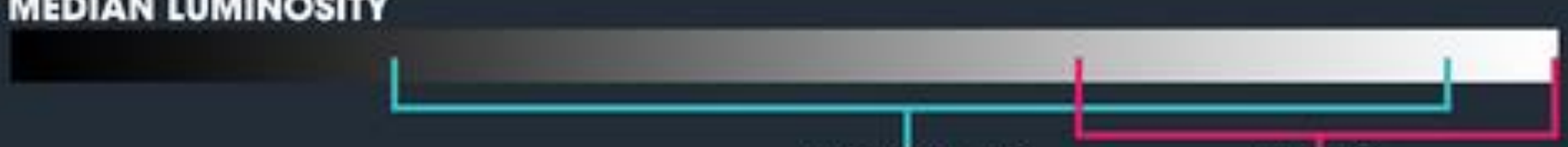
- Cutout
  - 알파값은 적용되나 그 정도를 지정할 수 없음
  - 완전 투명 혹은 완전 불투명만 가능
    - 벌레먹은 잎
- Fade
  - 일괄적으로 투명도 지정
  - 빛의 반사광 까지 모두 투명도가 적용됨
  - 주로 홀로그램을 표현할때 사용

SHADER CALIBRATION SCENE
METALLIC VALUE CHARTS

ALBEDO RGB

ALBEDO DEFINES THE **OVERALL COLOUR** OF AN OBJECT
VALUES USUALLY MATCH THE PERCEIVED COLOUR OF AN OBJECT

MEDIAN LUMINOSITY



NON-METAL sRGB RANGE **50-243**

METAL sRGB RANGE **186-255**

NON-METAL EXAMPLE VALUES

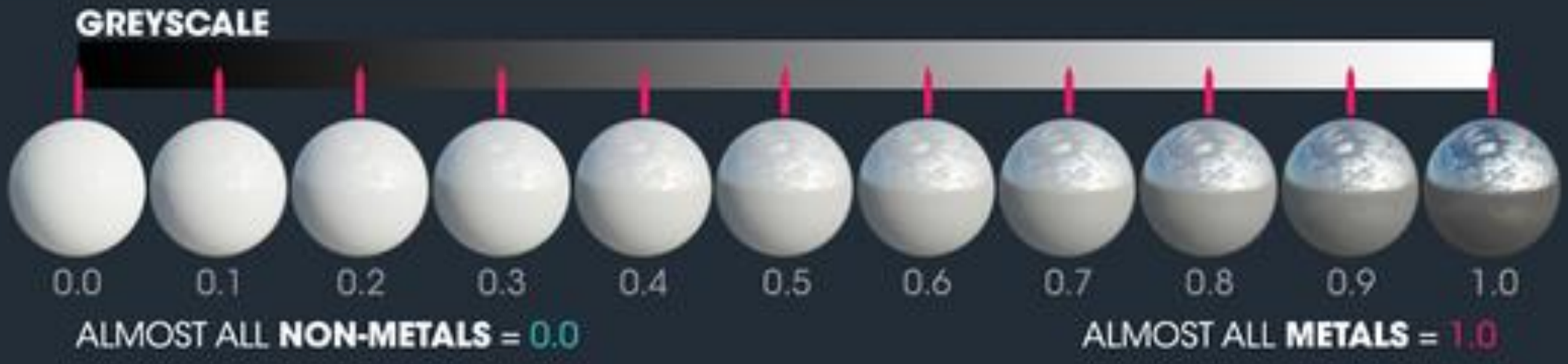
COAL RUBBER MUD GRASS BRICK WOOD CONCRETE

METAL EXAMPLE VALUES

GOLD BRASS COPPER IRON PLATINUM ALUMINIUM SILVER

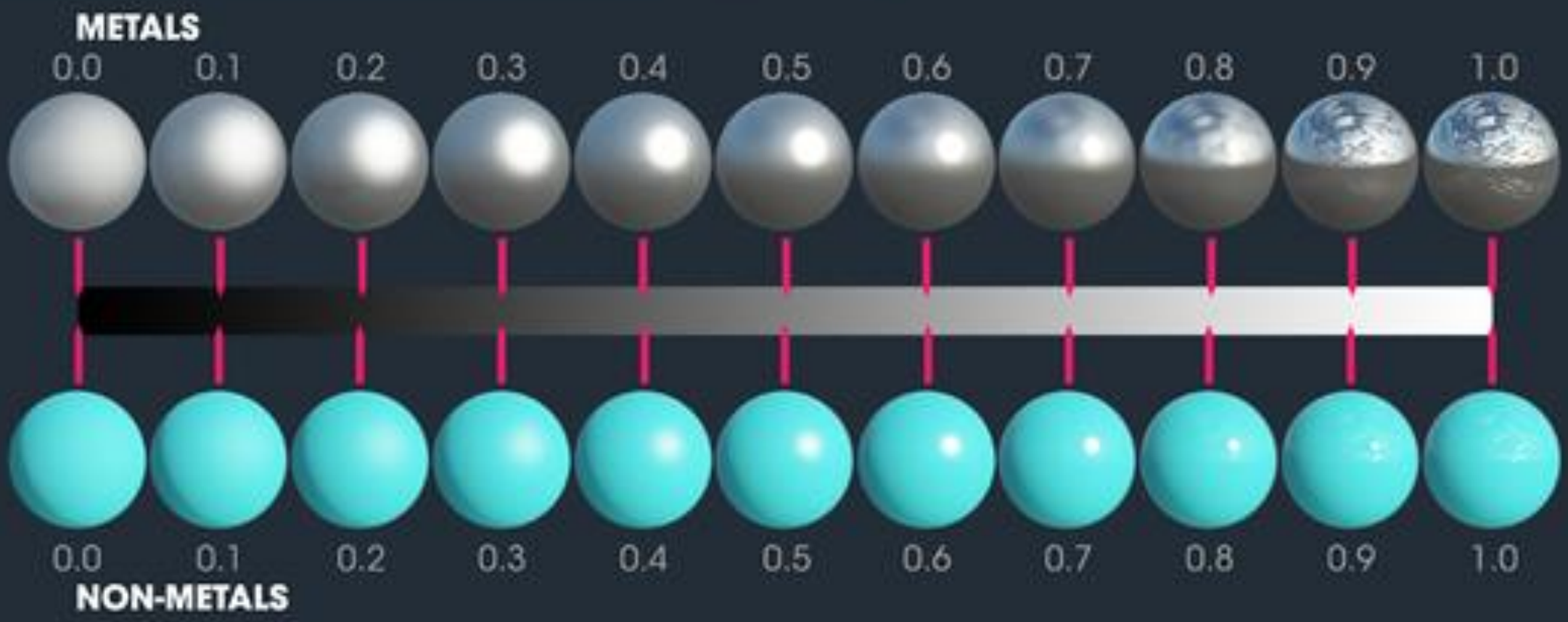
METALLIC R

METALLIC DEFINES WHETHER A SURFACE APPEARS TO BE **METAL** OR **NON-METAL**
WHILST PURE SURFACES WILL BE EITHER 0.0 **OR** 1.0, BEAR IN MIND FEW PURE, CLEAN, UNWEATHERED MATERIALS EXIST IN REAL LIFE
WHEN **TEXTURING** A METALLIC MAP, THIS VALUE WILL ALWAYS BE **GREYSCALE** AND IS STORED IN THE **R CHANNEL** OF AN RGB FILE



SMOOTHNESS A

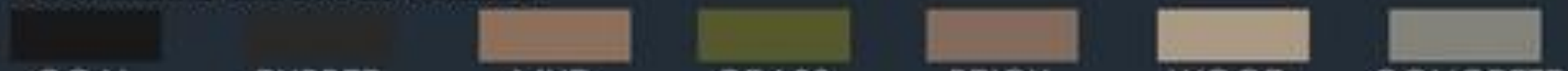
SMOOTHNESS DEFINES THE PERCEIVED **GLOSSINESS** OR **ROUGHNESS** OF A SURFACE
FOR TEXTURES, THIS IS STORED AS THE ALPHA CHANNEL OF THE **METALLIC MAP**



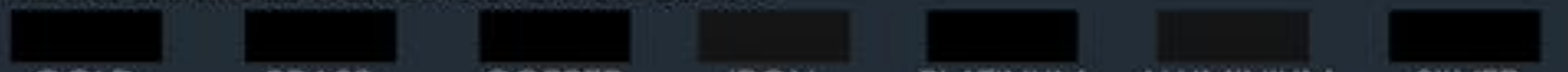
ALBEDO RGB

ALBEDO DEFINES THE **DIFFUSE REFLECTIVITY** OF A SURFACE
FOR **NON-METALS** THIS WILL USUALLY BE THE PERCEIVED COLOUR OF THE OBJECT. FOR **METALS** THIS IS USUALLY BLACK

NON-METAL EXAMPLE VALUES



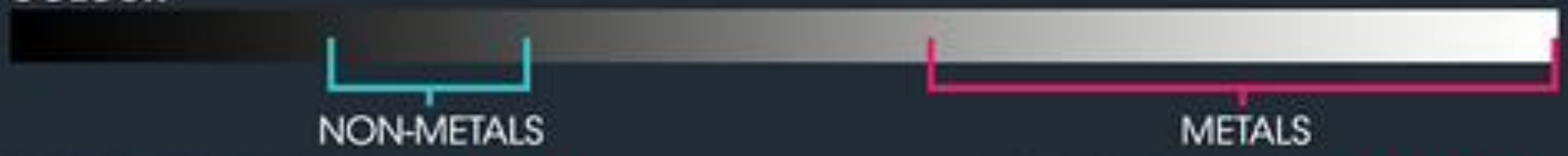
METAL EXAMPLE VALUES GENERALLY BLACK



SPECULAR RGB

SPECULAR DEFINES THE **SPECULAR REFLECTIVITY** OF A SURFACE
NON-METALS ARE USUALLY A DARK GREY OF ~55 in sRGB. FOR **METALS**, THIS IS USUALLY THE PERCEIVED COLOUR OF THE METAL

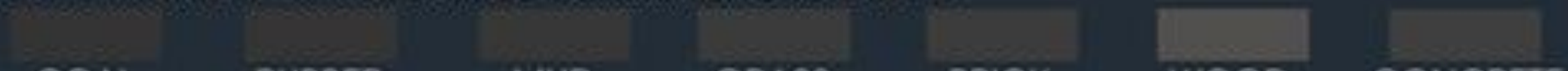
COLOUR



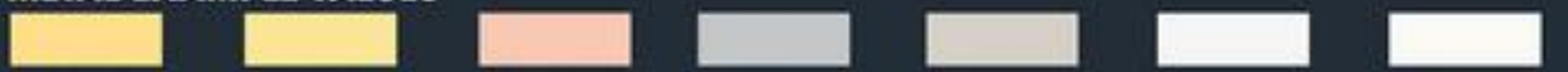
NON-METAL sRGB RANGE **40-75**

METAL sRGB RANGE **155-255**

NON-METAL EXAMPLE VALUES GENERALLY GREYSCALE

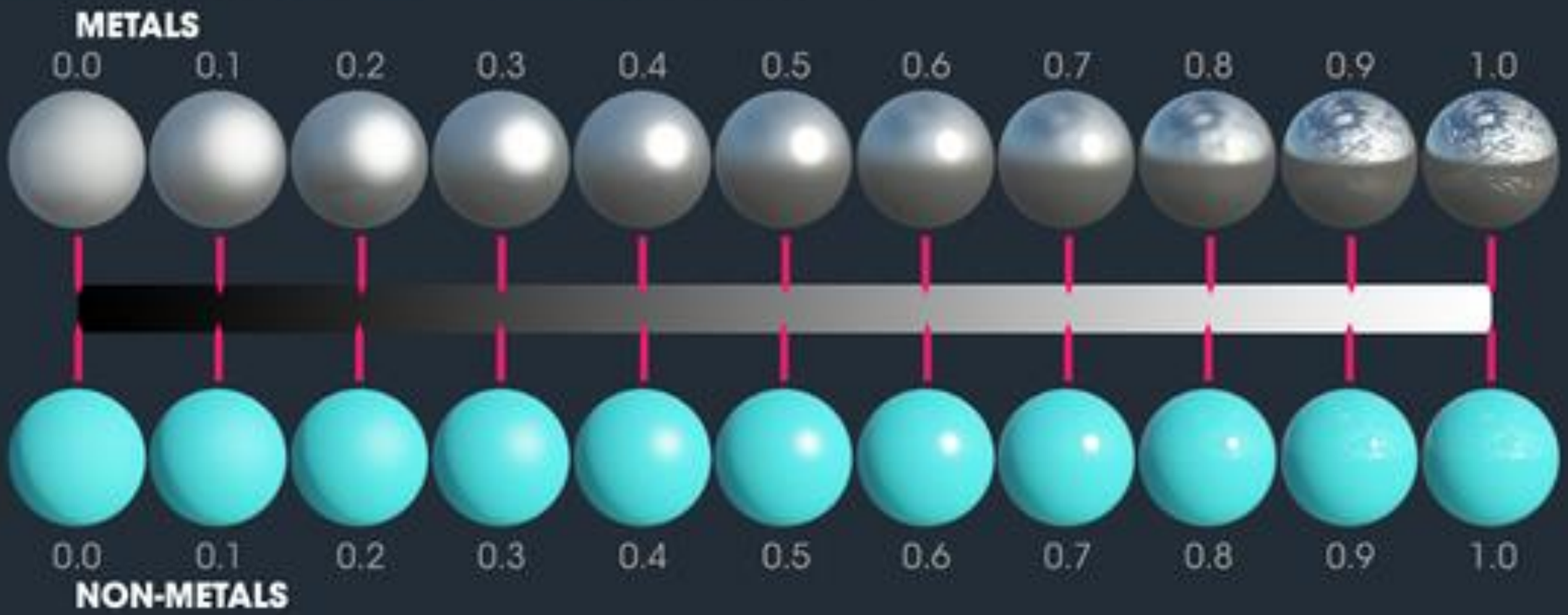


METAL EXAMPLE VALUES



SMOOTHNESS A

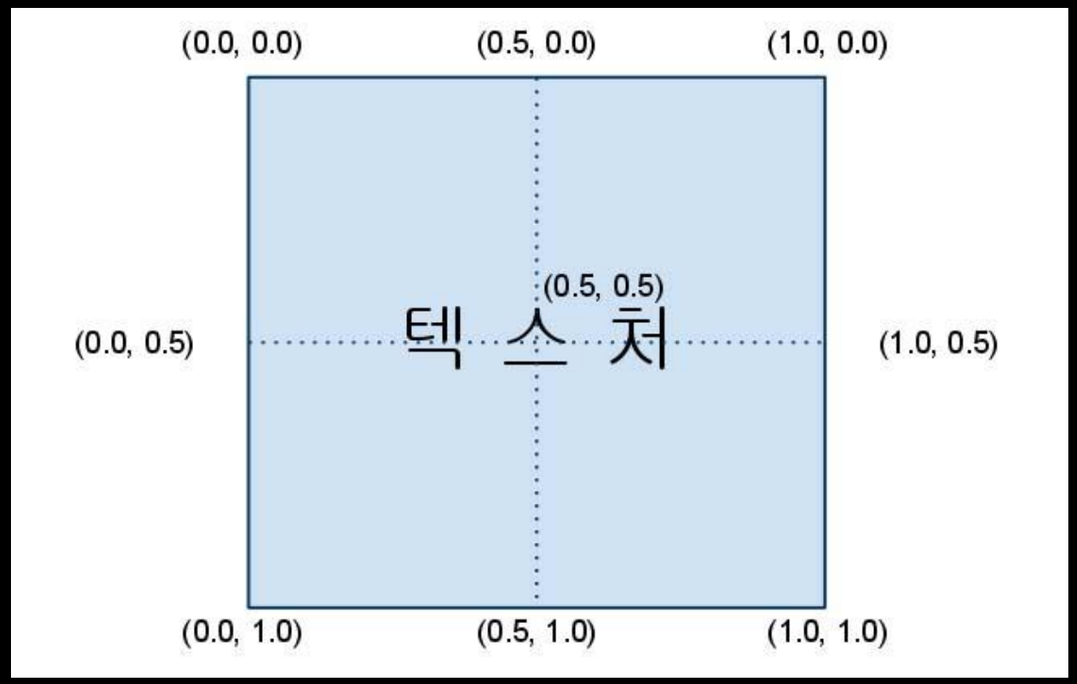
SMOOTHNESS DEFINES THE PERCEIVED **GLOSSINESS** OR **ROUGHNESS** OF A SURFACE
FOR TEXTURES, THIS IS STORED AS THE ALPHA CHANNEL OF THE **SPECULAR MAP**

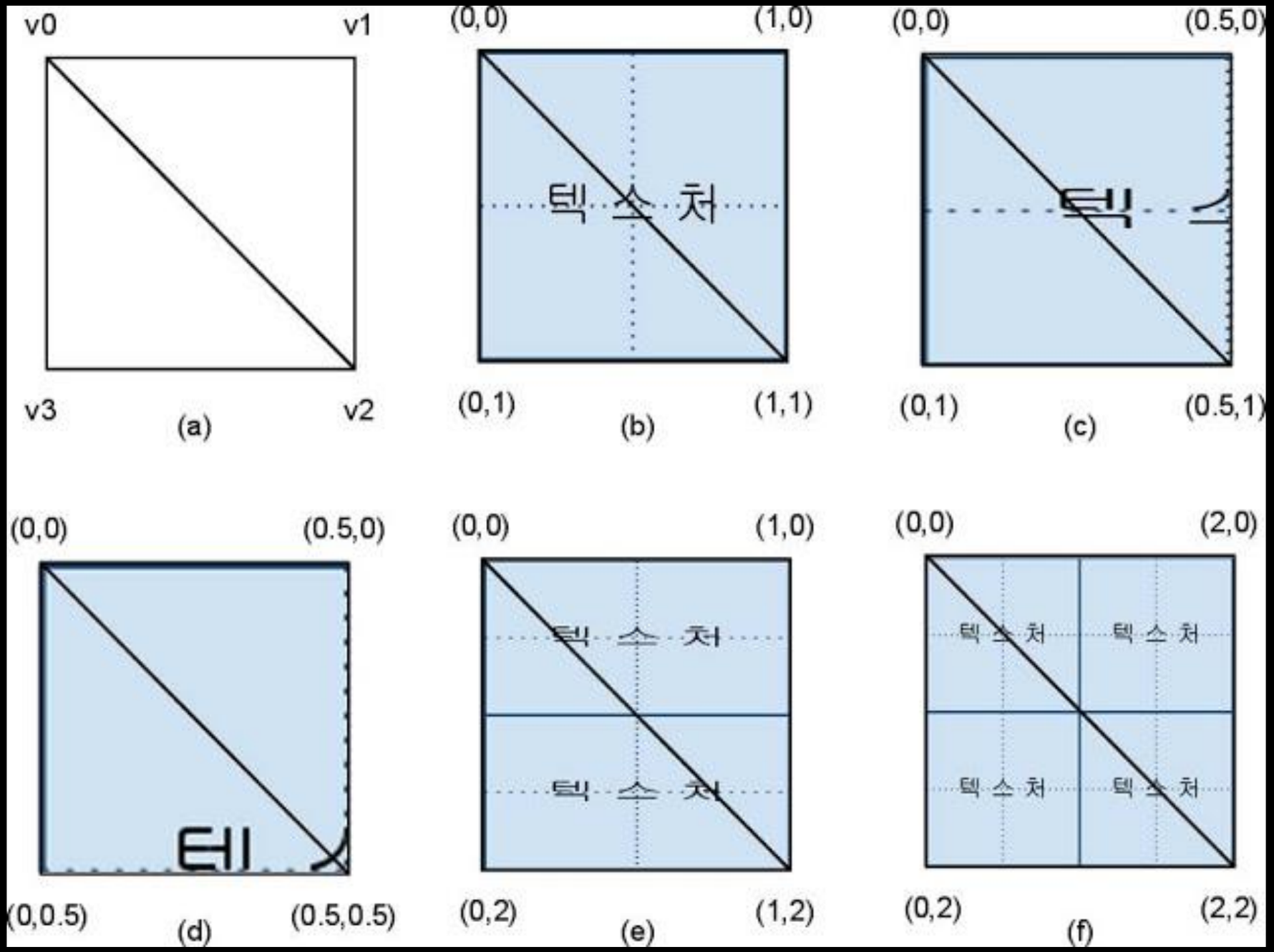


타일링 & 오프셋

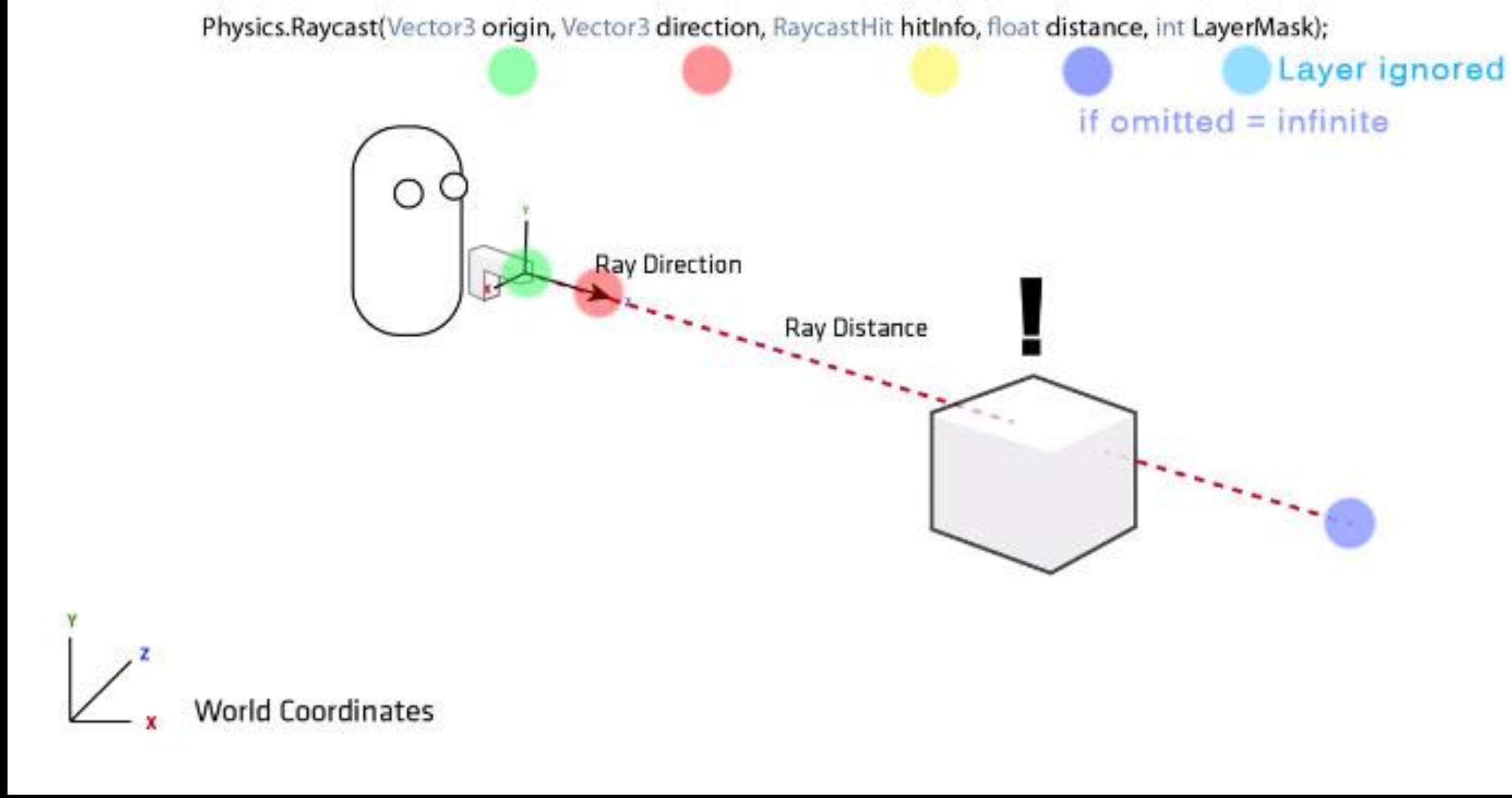
UV

- UV 란 텍스쳐의 좌표계
- UV Map 은 2차원 전개도, 접으면 X, Y, Z 3차원에 대응
- UV 란 명칭은 그냥 X, Y 와 혼동을 피할려고…



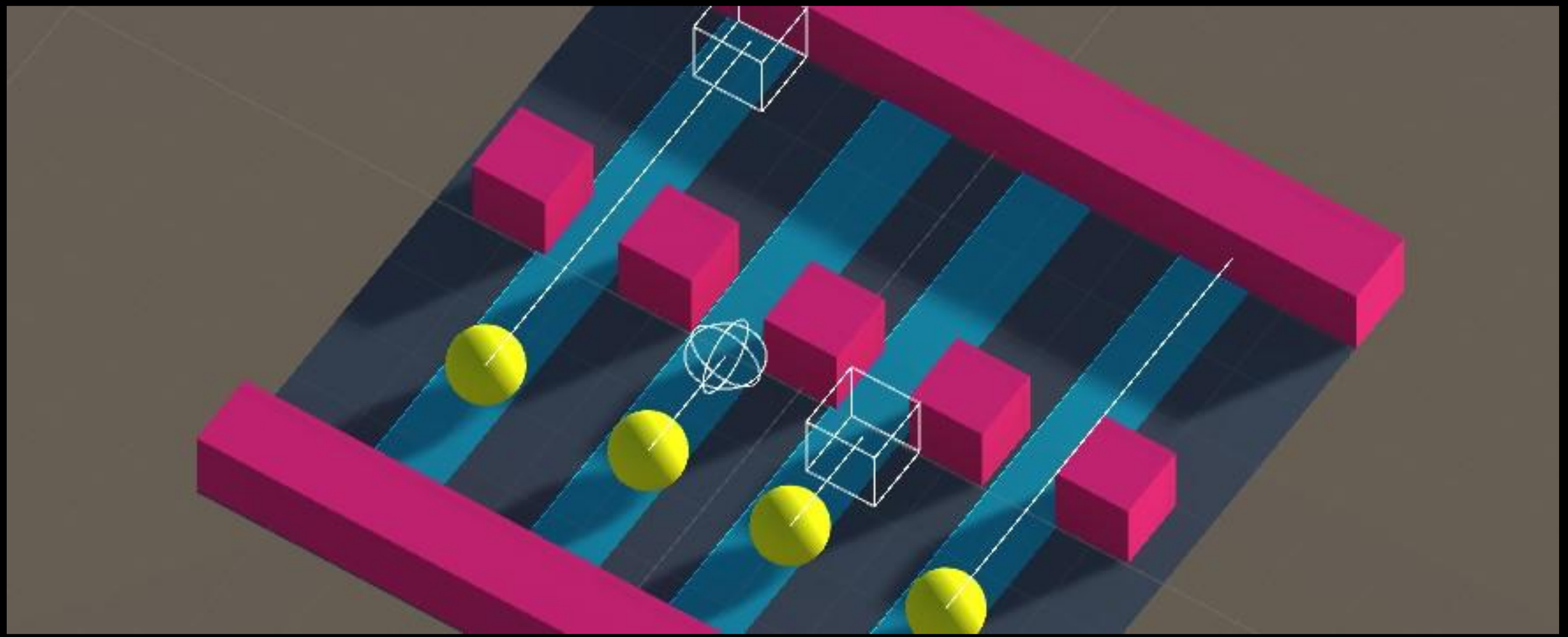

v0
v1
v3
v2
(a)
(0,0)
(1,0)
텍 스 처
(0,1)
(1,1)
(b)
(0,0)
(0.5,0)
텍
(0,1)
(0.5,1)
(c)
(0,0)
(0.5,0)
텍
(0,0.5)
(0.5,0.5)
(d)
(0,0)
(1,0)
텍 스 처
텍 스 처
(0,2)
(1,2)
(e)
(0,0)
(2,0)
텍 스 처
텍 스 처
텍 스 처
텍 스 처
(0,2)
(2,2)
(f)

Ray Casting



Physics.Raycast (rayOrigin, direction, out hit, maxDistance)

이외의 여러가지 오버로딩 형태를 익혀둘것

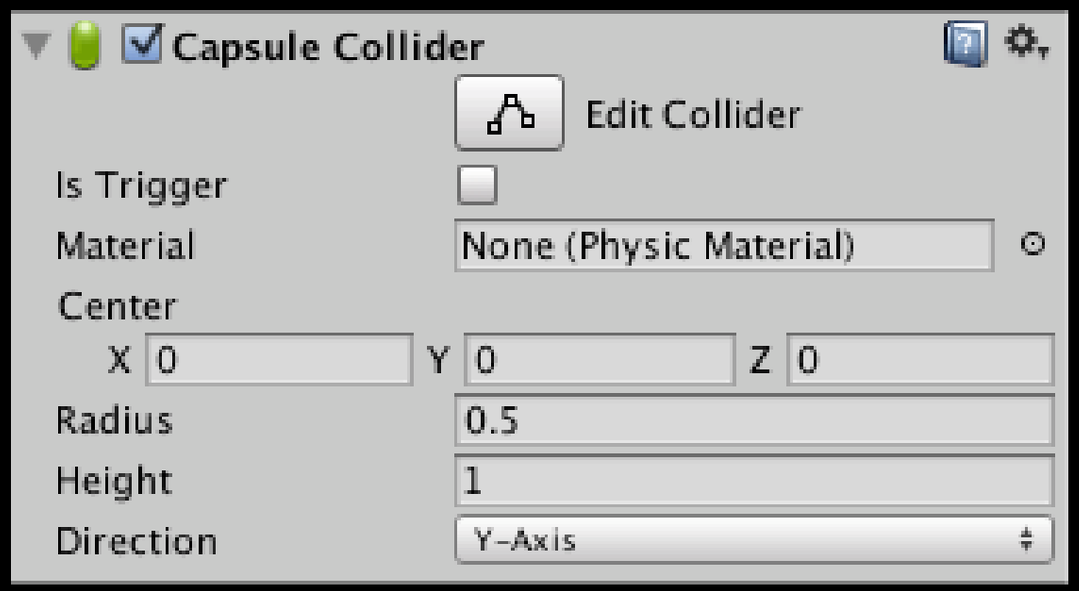


Physics.RayCast
Physics.LineCast
Physics.SphereCast

등의 다양한 형태의 캐스팅들 또한 알것

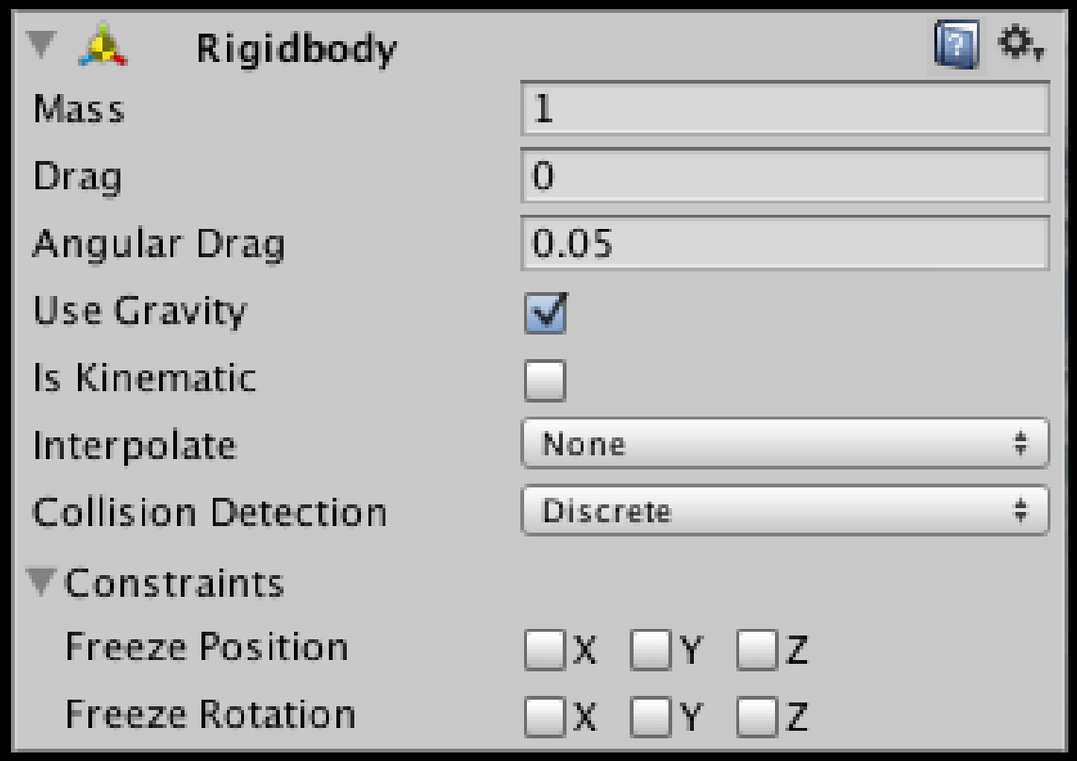
Capsule Collider

- 캡슐 형태의 콜라이더
- Is Trigger
  - 다른 물체가 통과 가능
- Center 콜라이더의 상대적 위치
- Radius 반지름
- Height 높이
- Direction 캡슐의 높이의 방향
  - = 캡슐을 세우는 방향



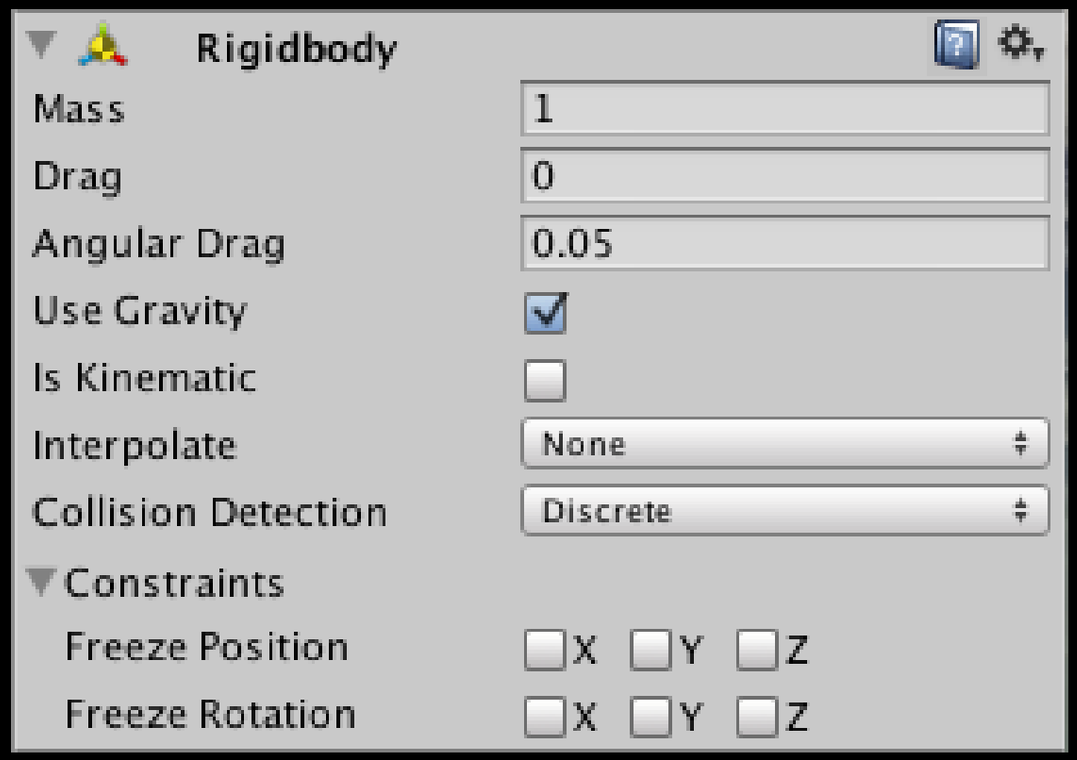
Rigidbody

- 오브젝트가 물리 엔진의 지배를 받게 함
- Use Gravity 중력 영향 받기
- Mass 질량
- Is Kinematic 외부의 힘 영향 받지 않기
- Drag 공기의 저항(마찰력)
- Angular Drag 회전에 대한 공기의 저항(마찰력)
- Constraints
  - 원하는 방향, 회전에 대해서 움직이지 못하게 잠금을 걸기



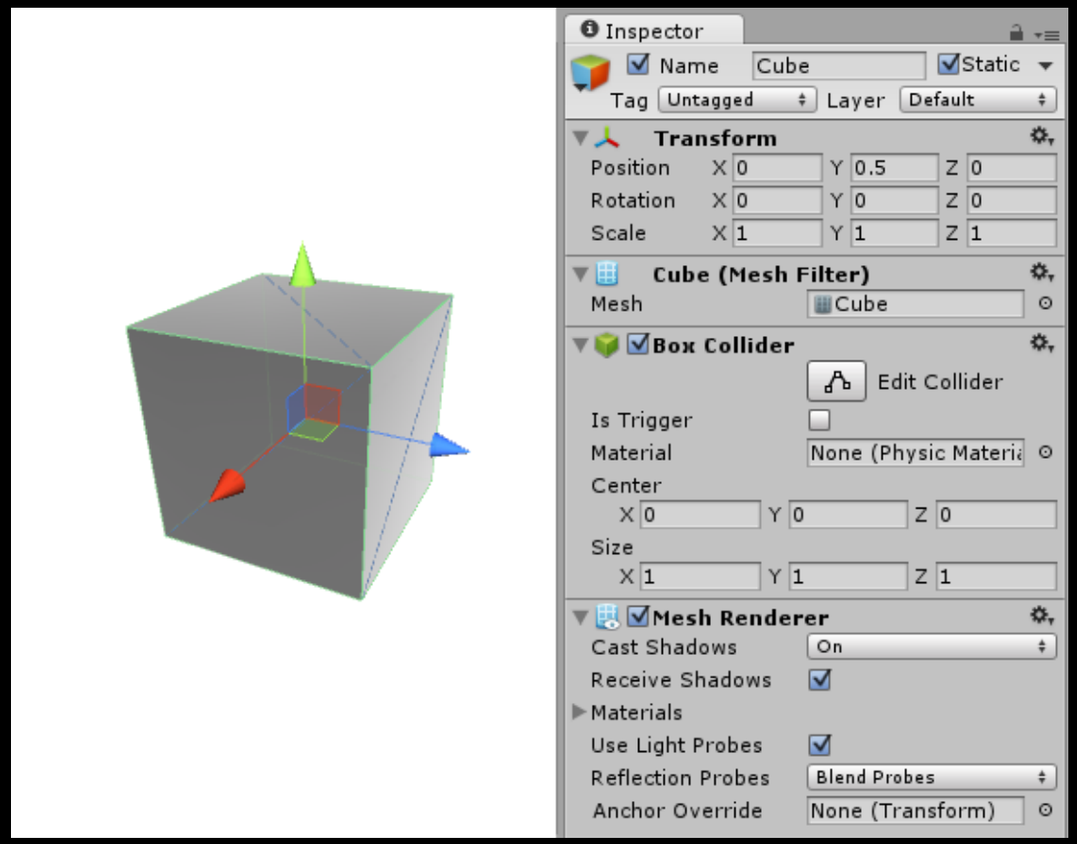
Rigidbody

- 리기드 바디는 강체라고도 부름
- 속도에 접근
  - rigidbody.velocity
- 힘을 추가
  - rigidbody.AddForce(new Vector3(x,y,z))



GameObject

- 게임 오브젝트는 컴포넌트를 쥐고 있는 홀더
  - 트랜스폼은 무조건 가진다
- 태그 – 게임 오브젝트를 구분-관리 하는 방법 중 하나
- 레이어 – 콜라이더 등 물리를 위한 태그
  - 태그와 달리 레이어는 레이어마스크를 통해 여러개를 동시에 필터링 가능
- 게임 오브젝트 좌측 상단의 체크 박스로 활성-비활성화 가능
- 컴포넌트 좌측 상단의 체크 박스로 개별 활성-비활성화 가능
- 컴포넌트를 추가하는 방법 – Add Component



유니티 UI (UGUI)

- 모든 UI 요소는 Canvas 안에 있어야 함
- 처음 UI 요소를 만들면 Canvas 와 EventSystem 이 자동으로 생성됨
- Canvas 의 모드를 Screen Overlay 로 할시 게임 화면 기준, Word Space 기준으로 할시 월드 좌표계 기준으로 배치
- Rect 툴을 사용해서 UI 요소를 편집 가능

유니티 UI (UGUI)

- 캔버스 아래 UI 요소는 RectTransform 과 아래를 가짐
  - 피벗: 위치를 찍는 기준점
  - 위치: UI를 배치하는 장소
  - 앵커: 배치할때 0,0 좌표가 시작되는 기준점
- 앵커 프리셋: RectTransform 좌측의 버튼
  - 많이 쓰이는 앵커-위치-피벗 들을 모아둔 프리셋이다
  - 그냥 클릭시 앵커를 교체한다
  - Alt 키를 누를시 위치도 같이 바꾼다
  - Shift 키를 누를시 피벗도 같이 바꾼다
  - Stretch 를 어떻게 적용하는지 반드시 알아둘것

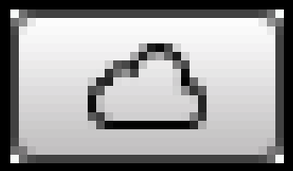
유니티 UI (UGUI)

- Button 컴포넌트
  - intractable: 상호작용 가능한 모든 UGUI 컴포넌트들이 가지고 있는데, 체크하면 상호작용 가능, 체크 해제시 상호작용 불가능
  - transition: 유저가 상호작용할때(클릭하거나 마우스를 호버하거나 할때) UI가 어떻게 바뀌는지 지정
  - OnClick: 버튼이 클릭됐을때의 이벤트

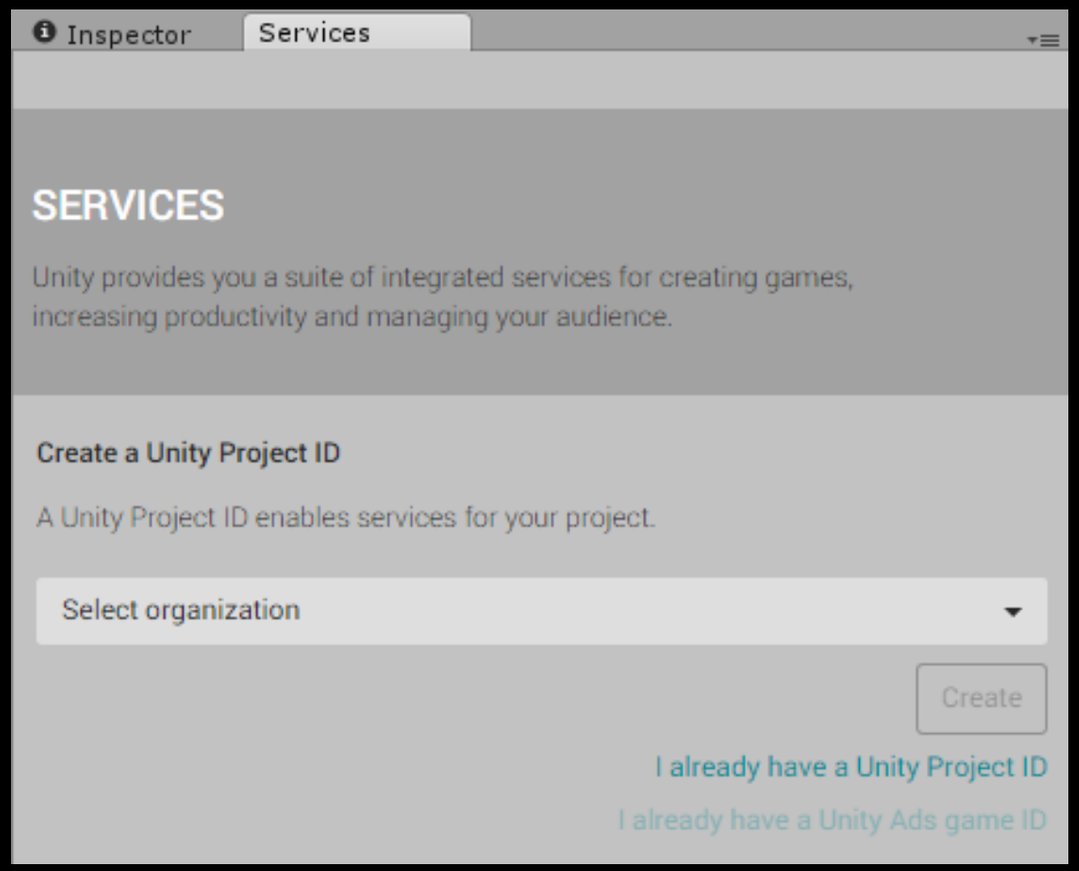
유니티 UI (UGUI)

- 슬라이더
  - 슬라이더 오브젝트에서 실제로 그림을 그리는 부분은 Image 컴포넌트 임에 유의
  - 슬라이더는 지정된 value 만큼, fill 로 지정된 rect 의 사이즈를 늘이고 줄여서 슬라이더를 채우는 것 처럼 보임

Unity Service



- 구름 버튼을 누르면 유니티 서비스 윈도우로 진입
- 반드시 Organization 과 Project Name 을 명시해야함
- 여기서 Unity Ads 등의 서비스 가능
- Ads: 광고 수익 창출
- Analytics: 플레이어 패턴 분석
- Could Build: 클라우드 빌드
- Collaborator: 협업 동기화 기능 (베타)



모듈	주제	하위 주제	인증 목적
프로그래밍	카메라 API	ScreenPointToRay	기존 코드의 목적 인식
	게임객체	컴포넌트	코드의 특정 라인 마무리
	메소드/기능	지정 및 사용	메소드 목적 설명 결과로 메소드 구별
	모노비헤이비어(MonoBehavior) API	Awake	결과로 메소드 구별 필요한 결과로 메소드 인식
		FixedUpdate	특정 메소드의 효과 평가
	객체 지향 프로그래밍	객체	제공된 코드의 클래스 정의 인식
	쿼터니언	사용	쿼터니언 설명
	타임	델타타임	델타타임 설명
	Unity 인터페이스	파일 관리	코드 내에서 퍼블릭 변수 구별 새로운 스크립트 생성
	변수	부동(Floating) 소수점	코드 내에서 변수 인식 및 대체
		정수	코드 내에서 변수 인식 및 대체
		Vector3	변수 정의

프로그래밍 파트

카메라 API

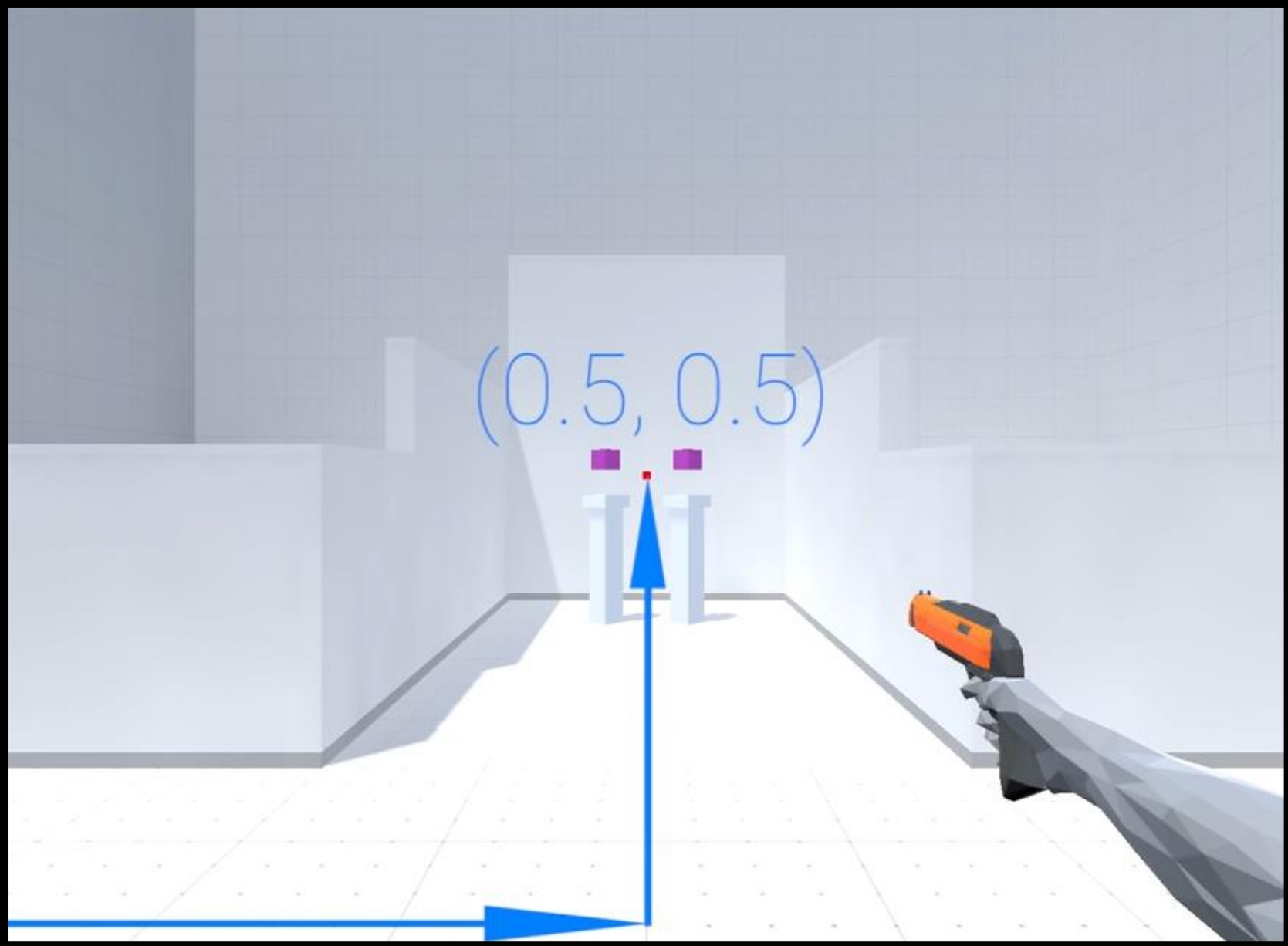
- 현재 Main 태그를 가진 카메라로 접근하는 코드
  - Camera.main

ScreenPointToRay

- ScreenPointToRay(Vector2 point)
  - 카메라 화면 상의 좌표를 주면, 그곳으로 향하는 광선을 생성

ScreenPointToRay

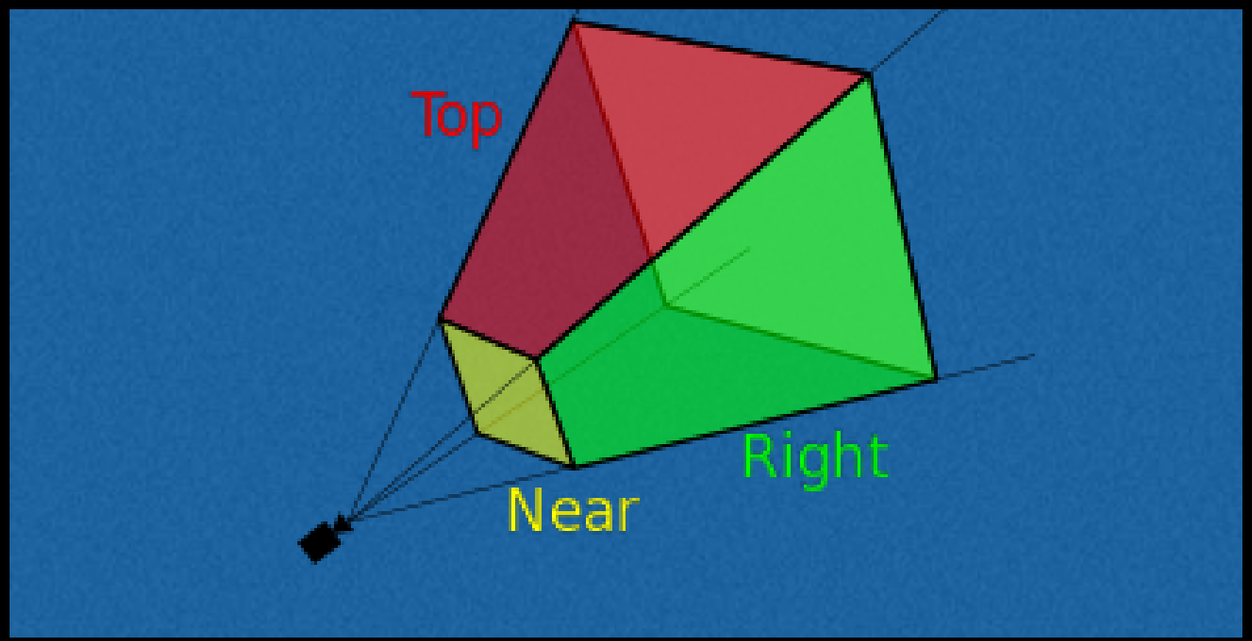
- 즉 x = 0.5, y = 0.5 일시,
  플레이어가 보는 화면의 정 중앙을 말함



ViewportToRay

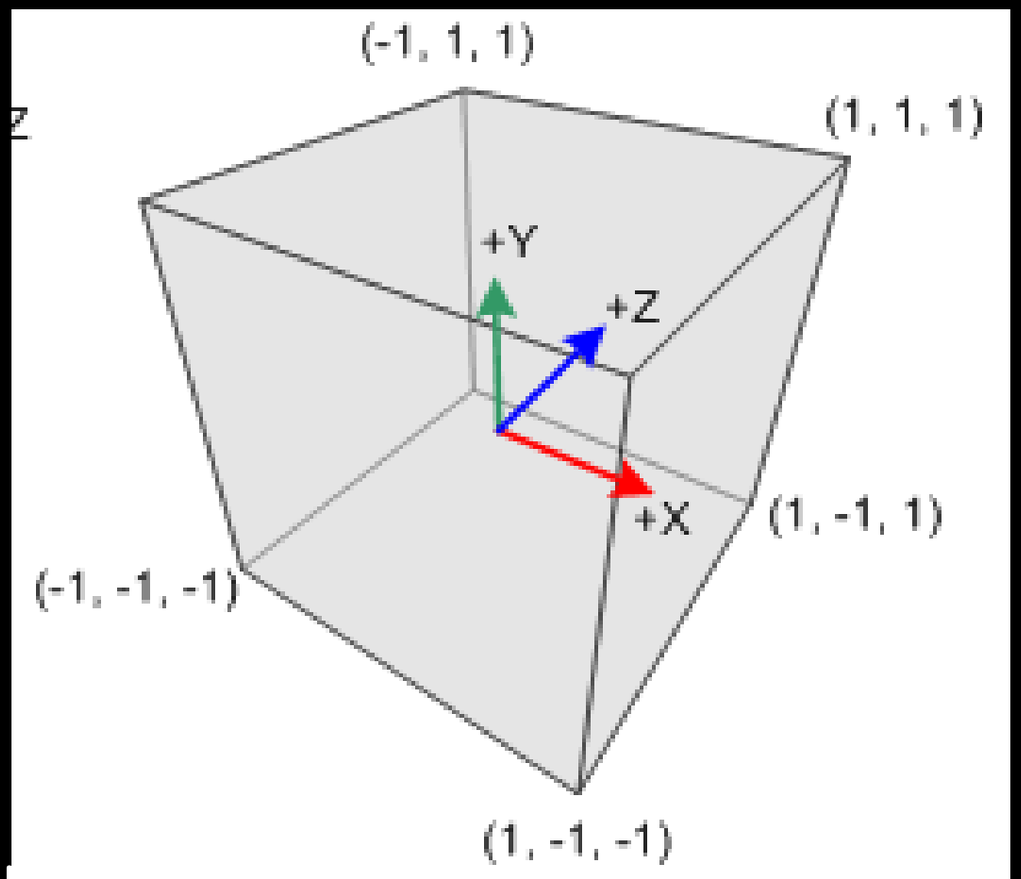
- ViewportToRay(Vector3 point)
  - 카메라 뷰포트 상의 좌표를 주면, 그곳으로 향하는 광선을 생성
  - 프러스텀 상 좌표는 무슨 의미?

ScreenPointToRay



- 프러스텀 상의 좌표 = 카메라가 보는 세상을 기준으로 한 좌표계
  - 카메라는 모든 세상을 가로 세로 높이가 1인 큐브로 변환해서 화면으로 출력함

ViewportToRay



- 프러스텀 상의 x,y,z 는...
  - x,y 는 화면의 가로 세로 위치
    - (0,0)은 화면 좌측 하단, (1,1)은 우측 상단
  - z 는 화면의 깊이

ViewportToRay

- 결론
  - Camera.main.ViewportToRay(new Vector3(0.5f,0.5f,0.5f));
  - 이 코드의 의미는, 메인 카메라의 위치에서 화면상의 정중앙 (깊이는 0.5)으로 향하는 레이를 생성하는 것
    - 마지막 z 값은 카메라의 깊이
      - 0보다 크면 어떤 값이든 앞쪽 방향
  - 이 코드로 화면의 정중앙 앞쪽에 있는 물체를 검사 가능

컴포넌트

- 특정 컴포넌트를 코드로 동적 추가할시
- gameObject.AddComponent<Type>();

Monobehaviour

- Monobehaviour 는 컴포넌트로서 게임 오브젝트에 부착 되기 위한 베이스 클래스
- 유니티의 메세지를 받을 수 있게됨
- 초기화를 목적으로, 게임이 시작되거나 오브젝트가 생성될때 가장 먼저 한번 실행되는 메세지
    - OnEnable, Awake, Start 순번으로 실행
        - OnEnable 은 반복 실행 가능

Monobehaviour

- Update – 새로운 화면을 랜더할때, 즉 초당 프레임에 맞추어 실행
- FixedUpdate – 화면이 갱신되는 주기와 상관없이 일정한 주기로 실행. 주로 정확하게 계산해야 하는 물리 관련 코드를 삽입

쿼터니언

- 물체의 회전은 Vector3 로는 온전하게 나타낼수 없음
  - 짐벌락 현상 때문에 x,y,z 만으로는 90도 회전을 나타낼 수 없음
  - 따라서 x,y,z 이외에 w 라는 값을 하나 더 가지는 쿼터니언을 통해 회전을 나타냄
- 인스펙터에서는
  - 편의상 Vector3 로 Transform 의 rotation 을 편집
- 하지만 Transform.rotation 은 사실 Quaternion 타입

쿼터니언

- 물체에 x=30,y=60,z=90 도 회전을 주는 코드 예시

  - transform.rotation = Quaternion.Euler(new Vector3(30f,60f,90f));

    - Euler(Vector3)는 Vector3 로 회전을 넣으면, 알아서 Quaternion 으로 변환해주는 함수

- Euler (오일러)는 특정한 수식이 아니라, Vector3 로 회전을 나타내게 한 수학자 이름

타임

- Time.deltaTime
  - 현재 프레임과 이전 프레임 사이의 시간차
  - 현재 Update 함수와 이전 Update 함수 사이의 시간차
  - 즉 1초에 화면을 60번 그리는 게임은 (렉이 없는 경우) Time.deltaTime 은 1/60 초

변수

- 매우 기초적인 변수 타입별 서식
  - Vector3: new Vector3()
    - Vector2 에서의 캐스팅
  - float: 끝에 f
  - int: 정수

인스턴스화

- 유니티에서 게임 오브젝트는 Instantiate 함수로 찍어냄
  - 예외적으로 빈 게임 오브젝트는 new 키워드 사용가능
  - Instantiate 함수는 원본 오브젝트를 주면 씬 상에 복제본을 생성
    - 원본을 원점에 찍어낼때
      - **Instantiate**(Object **original**);
    - 원본을 위치와 회전을 지정해서 찍어낼때
      - **Instantiate**(Object **original**, Vector3 **position**, Quaternion **rotation**);

인스턴스화

- 유니티 5.3 이전의 버전
  - 변수에 레퍼런스를 저장할시,
    Instantiate 에서 as 키워드로 명시적으로 캐스팅해야함
- 최신버전: 알아서 타입을 추측하므로 캐스팅 안해도됨
- 이전버전
  - Rigidbody clone = Instantiate(projectile, transform.position, transform.rotation) as Rigidbody;
- 최신버전
  - Rigidbody clone = Instantiate(projectile, transform.position, transform.rotation)

충돌 체크 함수

- OnCollisionEnter/Exit/Stay(Collision other)
  - 일반 충돌체와 충돌할때 발동
- OnTriggerEnter/Exit/Stay(Collider other)
  - 트리거인 충돌체와 충돌할때 발동
- 모두 2D 버전이 있음

충돌 체크 함수

- void OnCollisionEnter/Exit/Stay(Collision other)
  {
  if(other.collider.gameObject.tag == "Player")
  {
  // 충돌한 상대방의 태그를 검사
  }
  }

- Collision 변수는 충돌한 상대방을 포함하여, 충돌각, 충돌 지점 등의 정보까지 포함한 것이 특징

충돌 체크 함수

- void OnTriggerEnter/Exit/Stay(Collider other)
  {
      if(other.gameObject.tag == "Player")
      {
          // 충돌한 상대방의 태그를 검사
      }
  }

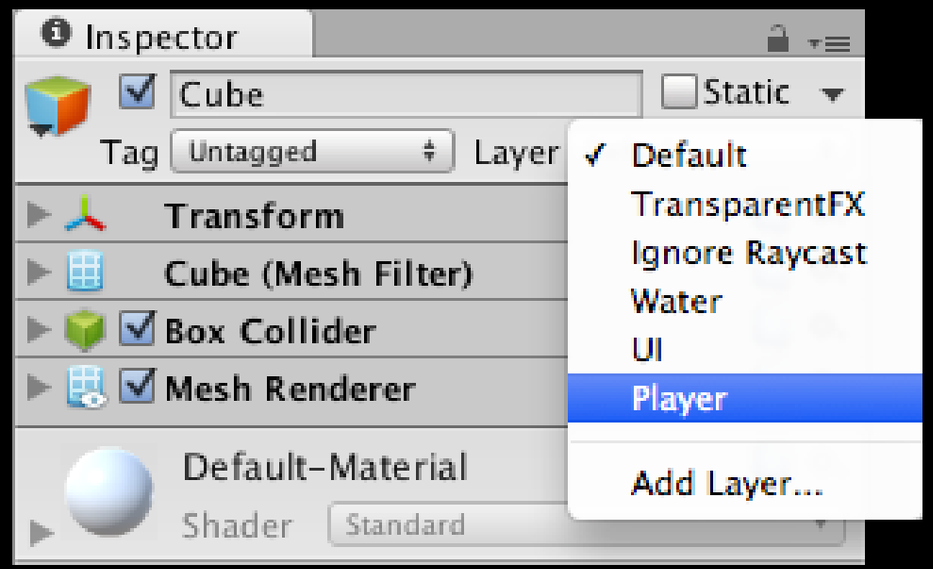
- 트리거 충돌의 경우
  실제 물리적인 충돌과 튕겨나감 등은 발생하지 않음

  - 그래서 상대방 콜라이더만 입력으로 받아오는 것

태그 / 레이어

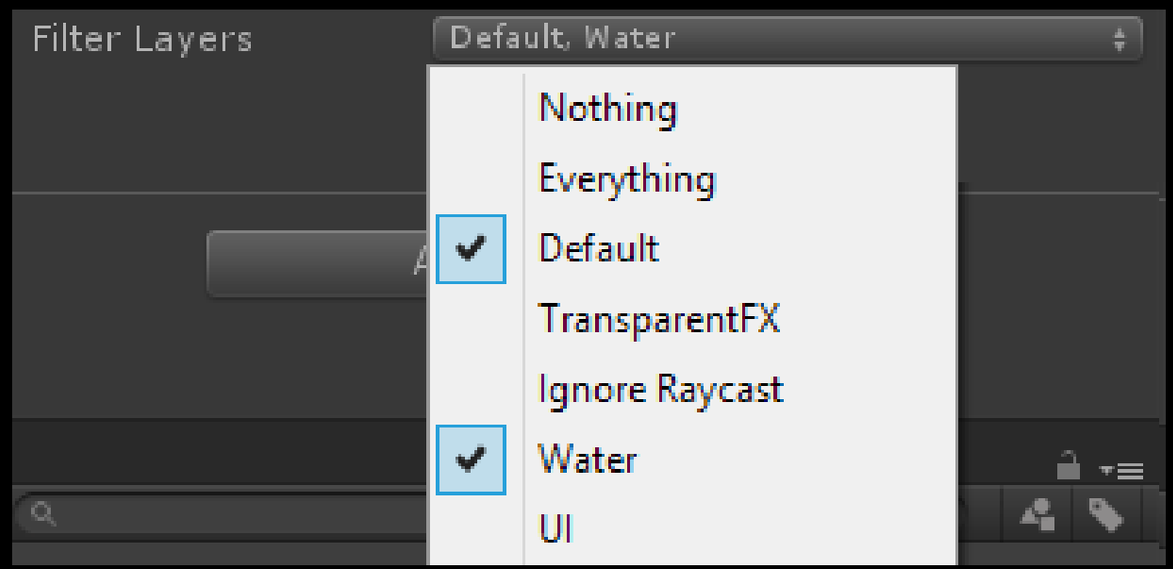
- 태그는 게임 오브젝트를 개발자 임의로 분류할때 사용
  - 블로그에서 포스팅할때의 태그와 역할이 같음
  - 태그의 출력 gameObject.tag
  - 태그의 비교 두가지 방법 (결과는 같다)
    - gameObject.CompareTag("TagName");
    - gameObject.tag == "TagName";

태그 / 레이어



- 레이어는 물리 처리를 위한 태그
  - 레이 캐스팅에서 특정 레이어의 Collider 들만 체크(혹은 무시)
  - 카메라가 특정 레이어의 물체들만 랜더링
  - 라이트가 특정 레이어의 오브젝트만 밝히기

태그 / 레이어



- LayerMask 변수를 통해 다수의 레이어를 필터링 가능
  - 태그와 달리 한번에 여러개를 필터링 가능
- 내부적으로는 int
  - 태그는 string 임