# 3D콘텐츠 이론 및 활용

## 13주(2). Scene 전환

- 씬 전환처리

# 학습개요

## 학습목표

- 게임 인터페이스를 활용할 수 있다.
- 스크립트를 통해 장면 전환에 대하여 구현할 수 있다.

## 학습내용

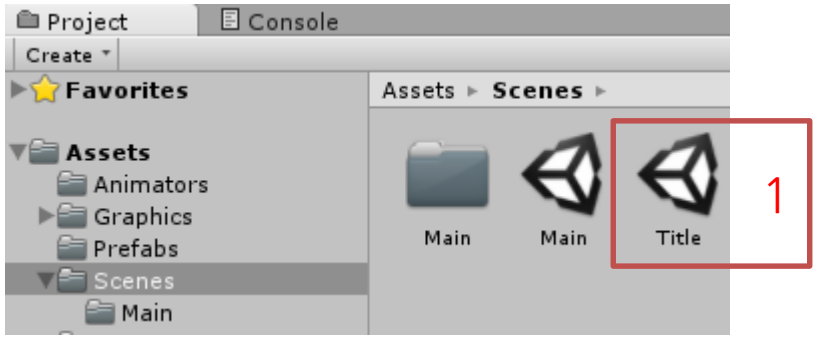
- 게임 인터페이스 제작
- 씬 전환 스크립트

# 1. 타이틀, 승리, 패배 장면 만들기

게임 상황에 따라 다양한 장면을 만들어 적용해 보자

## 1) 새로운 씬 생성

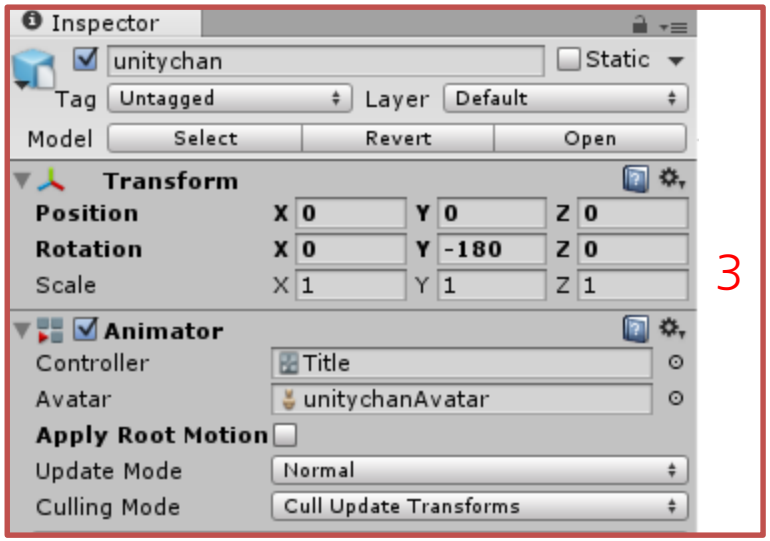
- 새로운 씬을 만든 후 Title로 저장한다.



# 1. 타이틀, 승리, 패배 장면 만들기

## 2) 게임 오브젝트 추가

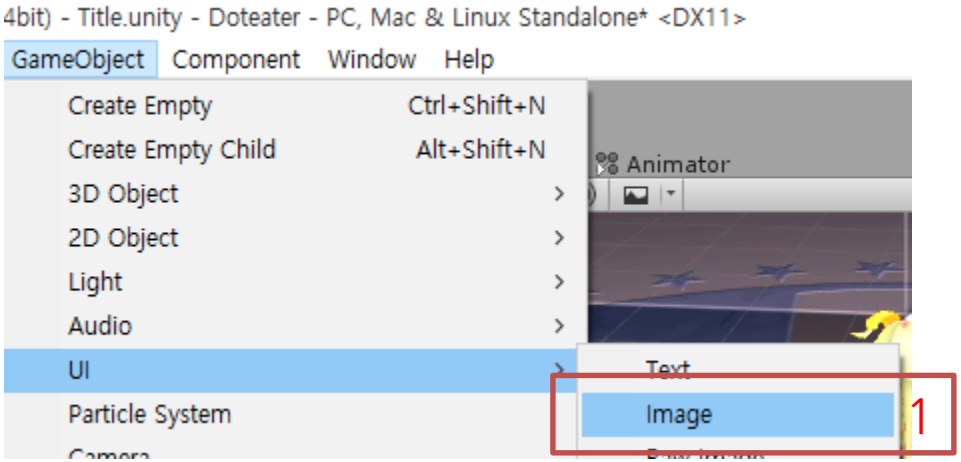
- 바닥, 광원, 카메라 위치와 각도 설정
- 유니티짱 모델 추가
- Animator Controller 생성 및 모션 지정 (3)

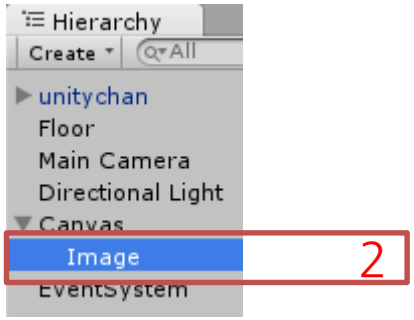


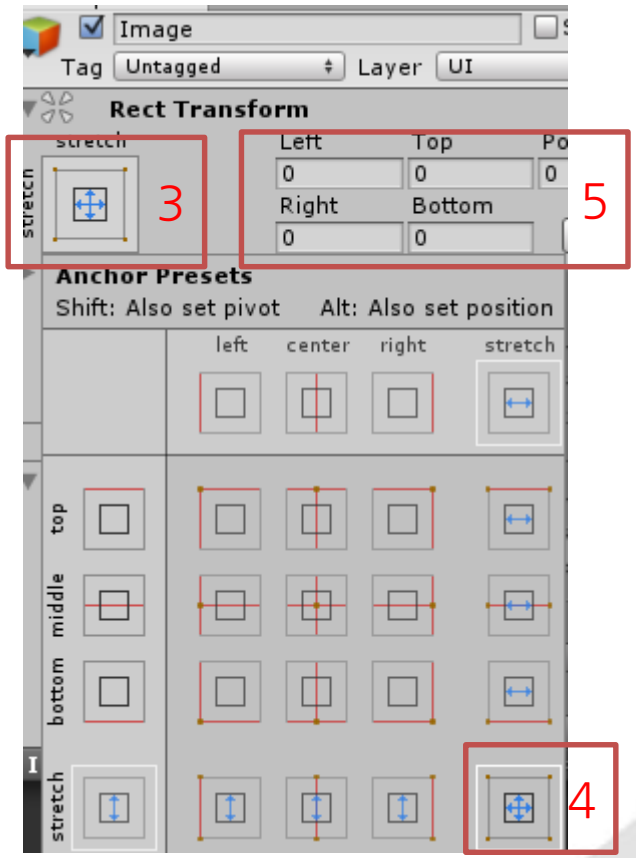
# 1. 타이틀, 승리, 패배 장면 만들기

## 3) 스프라이트로 문자 표시

- GameObject - UI - Image 선택하면 계층 뷰에 Canvas와 Image가 생성됨
- 이미지 위치 크기 조절 (3)(4)(5)



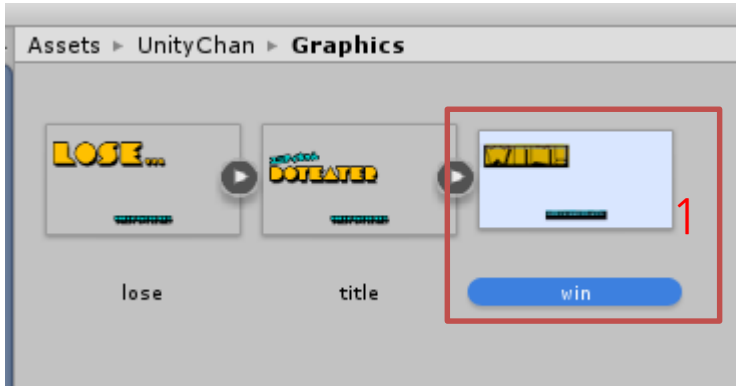



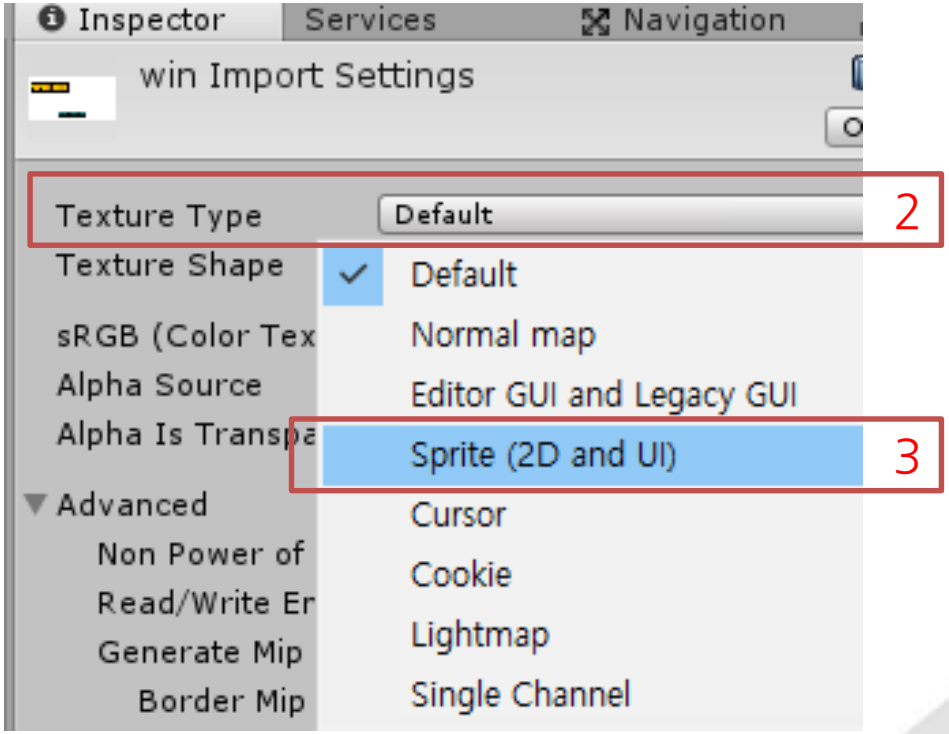



# 1. 타이틀, 승리, 패배 장면 만들기

## 3) 스프라이트로 문자 표시

- Textures폴드에 있는 준비된 이미지는 투명배경을 가진 이미지 파일로 Texture Type을 스프라이트 타입으로 지정함.

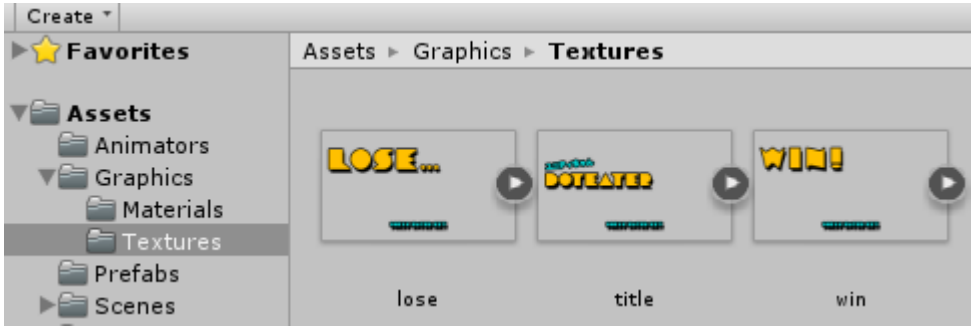


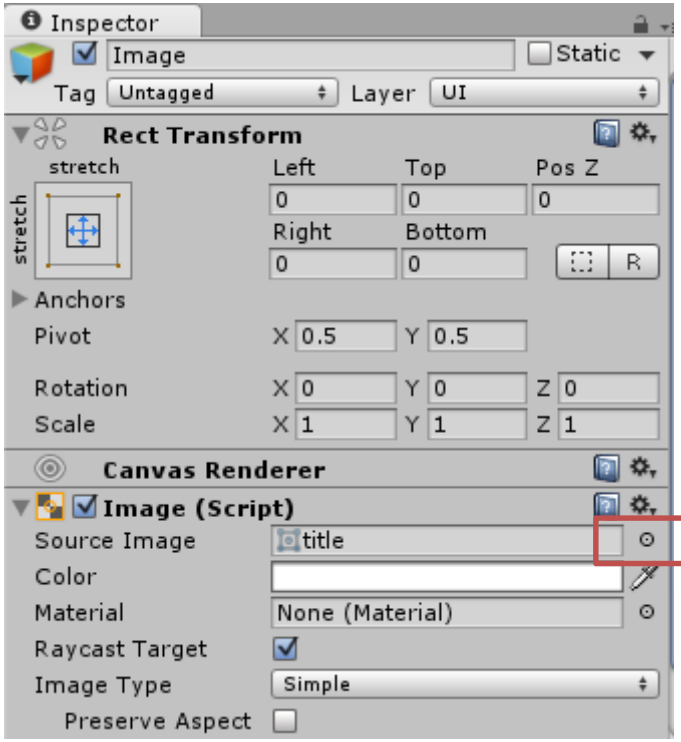



# 1. 타이틀, 승리, 패배 장면 만들기

## 3) 스프라이트로 문자 표시

- Textures폴드에 있는 스프라이트 이미지 적용하기 (5)(6)

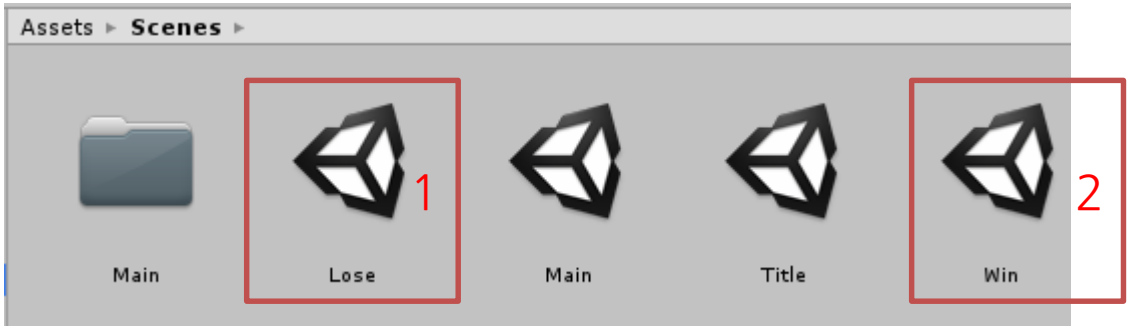


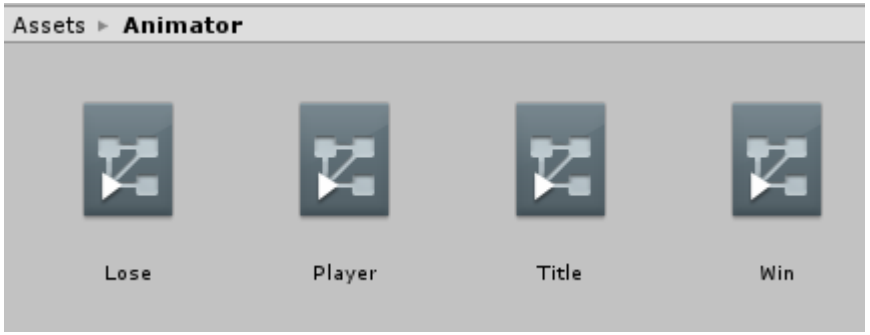

# 1. 타이틀, 승리, 패배 장면 만들기

## 4) 승리 장면, 패배 장면 만들기

- 타이틀 장면과 동일하게 Win, Loss 씬을 추가로 만든다. (씬 복사가능)



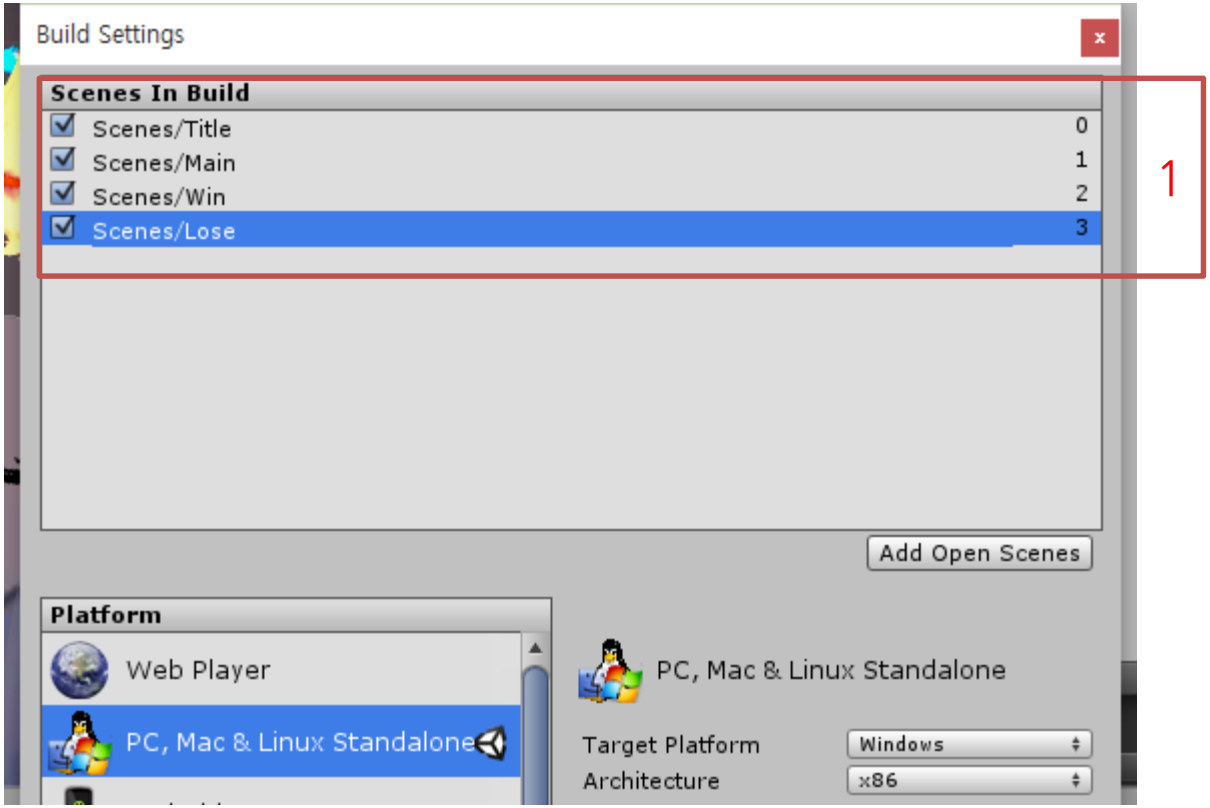

- 각 씬별 적합한 Animator Controller를 생성하고 적용
  - Title : WAIT01
  - Win : jump01B
  - GameOver : LOSE00



# 1. 타이틀, 승리, 패배 장면 만들기

## 5) 제작된 씬 적용하기

- 생성된 장면을 게임 과정에 적용하려면 사용할 씬을 등록해야 한다.
  - File - Build Settings에서 제작된 장면을 차례로 드래그한다.

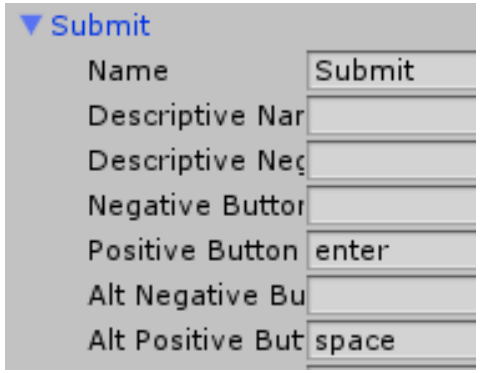


# 2. 스크립트로 씬 전환하기

## 1) 스크립트

- 엔터키 또는 스페이스 키를 누르면 장면이 전환되도록 테스트 해 보자
- SceneChanger.cs 를 만들고 각각의 씬에 추가(빈오브젝트)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class SceneChanger : MonoBehaviour {

    public string nextSceneName;

    void Update () {
        if (Input.GetButtonDown("Submit")) {
            SceneManager.LoadScene(nextSceneName);
        }
    }
}
```

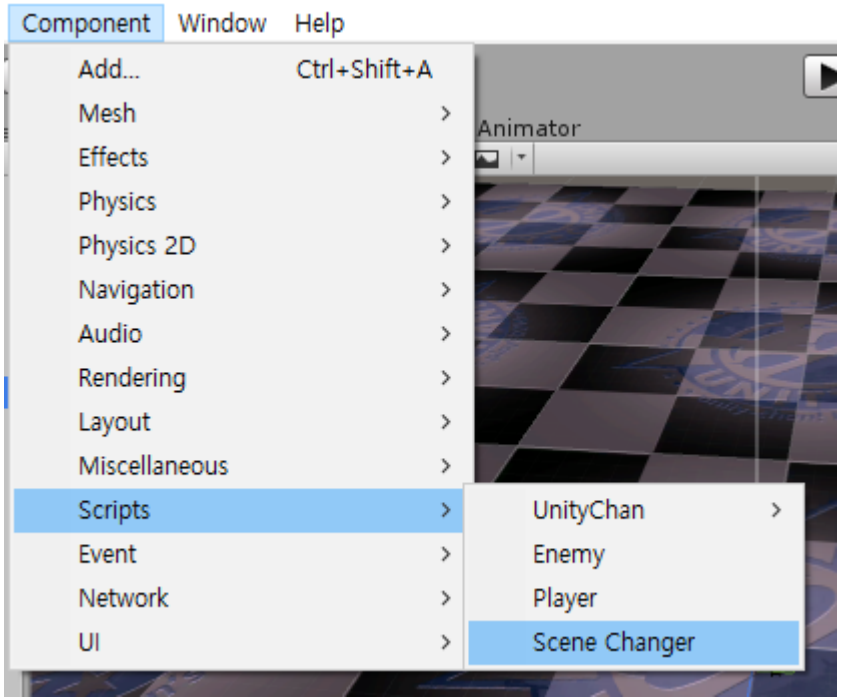


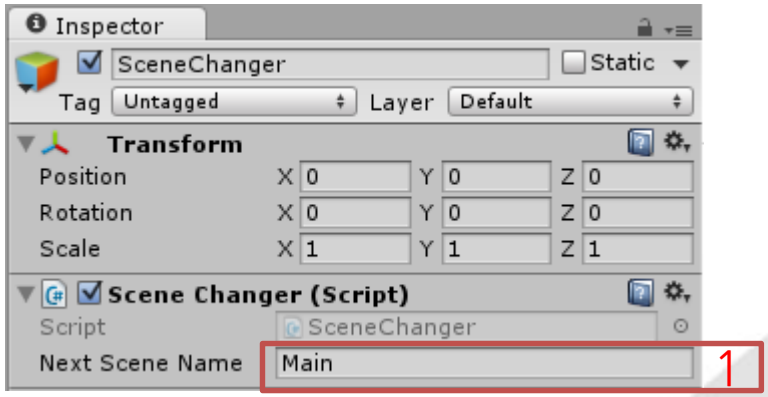
# 2. 스크립트로 씬 전환하기

## 2) 전환할 씬 지정하기

- Title, Win, Lose 씬에 있는 게임 오브젝트에 스크립트 추가

  - 각각의 씬에 빈 게임오브젝트를 만들고 SceneChanger 스크립트 선택
  - Title 씬에서는 Next Scene Name을 Main으로
  - Win, Lose 씬에서는 Title로 준다.



# 2. 스크립트로 씬 전환하기

## 3) 승리 장면과 패배 장면 전환하기

- 동전을 모두 획득 했을 때, 적과 충돌했을 때의 씬 이름을 지정하고 전환처리
- Player.cs 수정
- Title 씬에서 플레이 및 확인

```
void Update () {

    if (GameObject.FindGameObjectsWithTag("Dot").Length == 0) {
        SceneManager.LoadScene ("Win"); // 수정
    }
}

void OnTriggerEnter (Collider hit) {
    if (hit.tag == "Dot") {
        Destroy(hit.gameObject);
    }
    if (hit.tag == "Enemy") {
        SceneManager.LoadScene ("Lose"); // 수정
    }
}
```

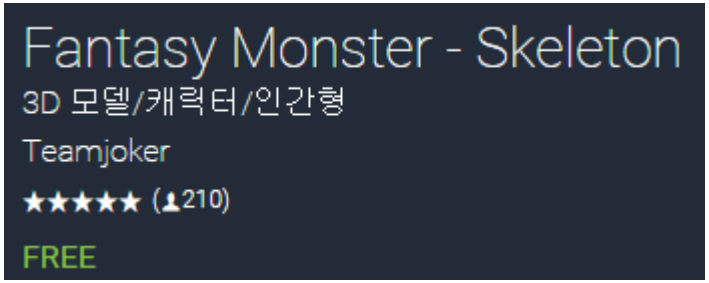
# 3. 보스 스테이지 추가

## 1) 조건

- 메인 게임에서 승리한 경우 보스판으로 진행

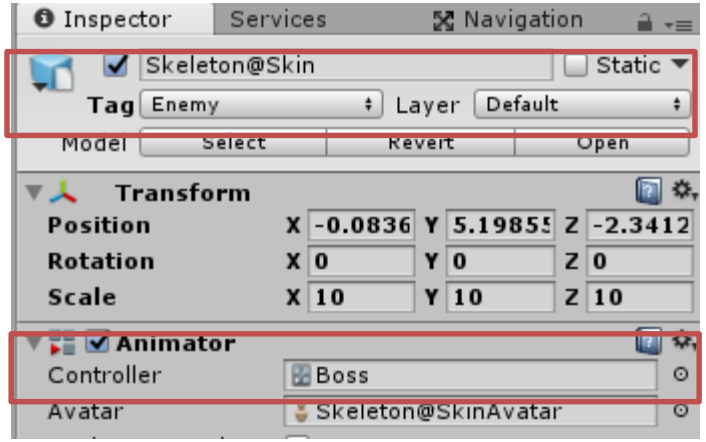
## 2) 에셋 다운로드

- 에셋 다운로드




## 3) Enemy 오브젝트 설정

- Scale
- Animator Controller

# 3. 보스 스테이지 추가

## 4) 방향전환처리

- 플레이어가 일정 범위내 접근하면 쳐다보도록 처리

```
public class Boss : MonoBehaviour {
    public Transform player;
    void Update ()
    {
Vector3 direction = player.position - this.transform.position;
// 추가
        if(Vector3.Distance(player.position, this.transform.posi
tion) < 5 ) //5m내 접근하면 쳐다보기
        {
direction.y = 0; // NPC 넘어가지 않도록 Y축 고정
            this.transform.rotation = Quaternion.Slerp(this.tran
sform.rotation, Quaternion.LookRotation(direction), 0.1f);
        }

    }
}
```

# 3. 보스 스테이지 추가

## 4) 방향전환처리

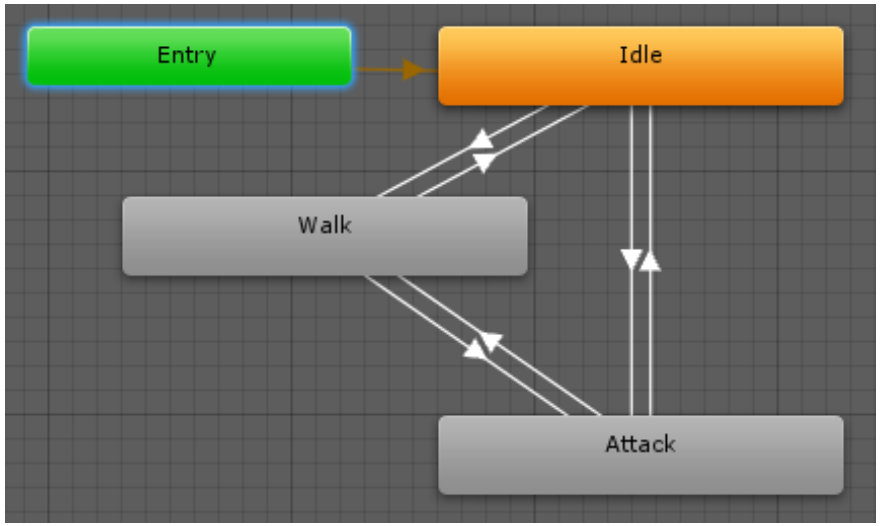
- 플레이어가 일정 범위내 접근하면 쫓아가도록 처리

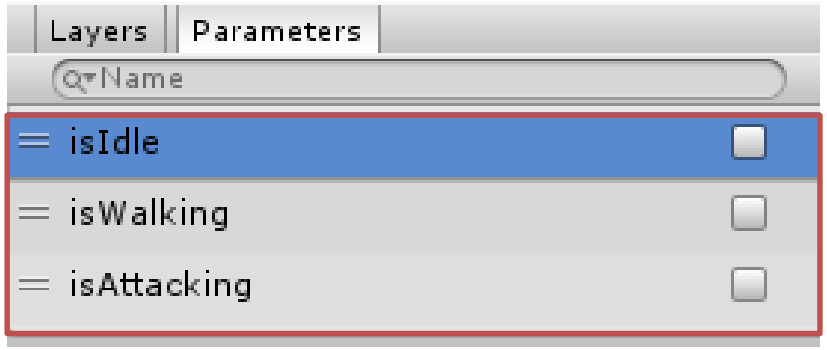
```
    void Update ()
    {

        if(Vector3.Distance(player.position, this.transform.posi
tion) < 5 )
        {

            this.transform.rotation = Quaternion.Slerp(this.tran
sform.rotation,Quaternion.LookRotation(direction), 0.01f);

            if(direction.magnitude > 3) // 3m 이내에 접근하면 이동
            {
                this.transform.Translate(0,0,0.05f);
            }
```

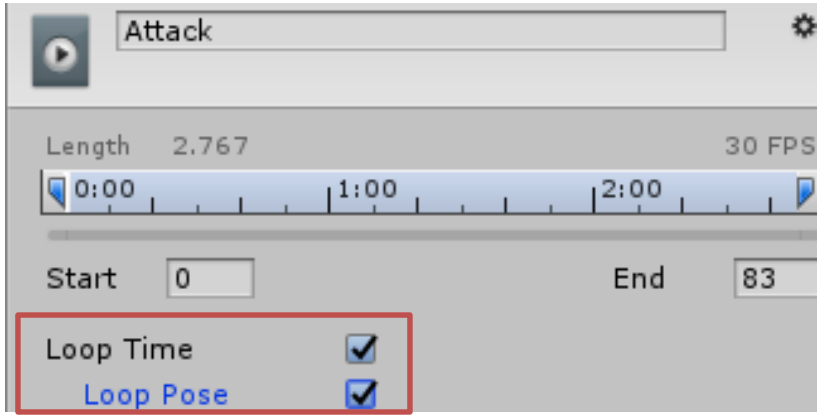
# 3. 보스 스테이지 추가

## 5) NPC 애니메이션 컨트롤러

- 애니메이션 상태 전환 조건 변수
- 상태 전환 조건 지정
- 애니메이션 Loop 처리



# 3. 보스 스테이지 추가

## 6) NPC 애니메이션 스크립트 처리

```
static Animator anim; // 상태 애니메이션 처리
void Start ()
{
    anim = GetComponent<Animator>(); // 애니메이션 처리 추가
}
void Update ()
{
        anim.SetBool("isIdle",false); // 애니메이션 처리 추가
        if(direction.magnitude > 3)   {
                                // 근접한 경우 걷거나 공격
            this.transform.Translate(0,0,0.01f);
            anim.SetBool("isWalking",true);
            anim.SetBool("isAttacking",false);
        }
        else
        {
            anim.SetBool("isAttacking",true);
            anim.SetBool("isWalking",false);
        }
```

# 3. 보스 스테이지 추가

## 6) NPC 애니메이션 스크립트 처리

```
void Update ()
{

    //
     }
     else //근접 상태가 아닌 경우 idle 애니메이션 처리
     {
          anim.SetBool("isIdle", true);
          anim.SetBool("isWalking", false);
          anim.SetBool("isAttacking", false);
     }

}
```

# 3. 보스 스테이지 추가

## 7) NPC 시야각 줄이기

- 플레이어가 시야를 벗어난 경우 공격 안함
- 앵글 변수와 각도 지정

```
    void Update ()
    {

            Vector3 direction = player.position -
 this.transform.position;

// 플레이어와의 시야범위 계산  추가
float angle = Vector3.Angle(direction,this.transform.forward);

if(Vector3.Distance(player.position, this.transform.position) <
 5 && angle < 30) // 지정한 각도보다 작은 경우에 반응하도록 수정
```