[유니티 기초] 12. 캐릭터 카메라

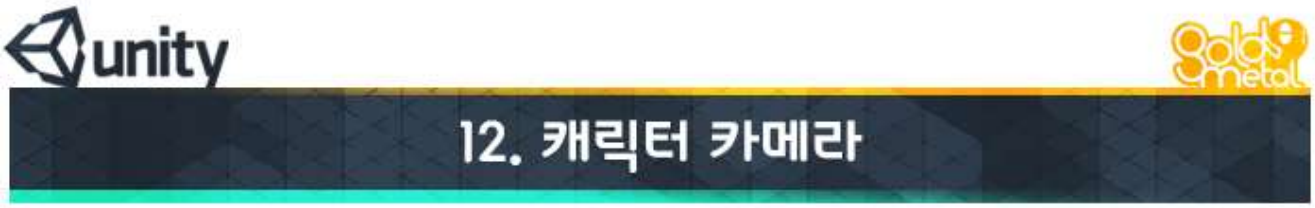
유니티·개발

2015. 10. 3. 21:01

http://blog.naver.com/gold_metal/220498575590

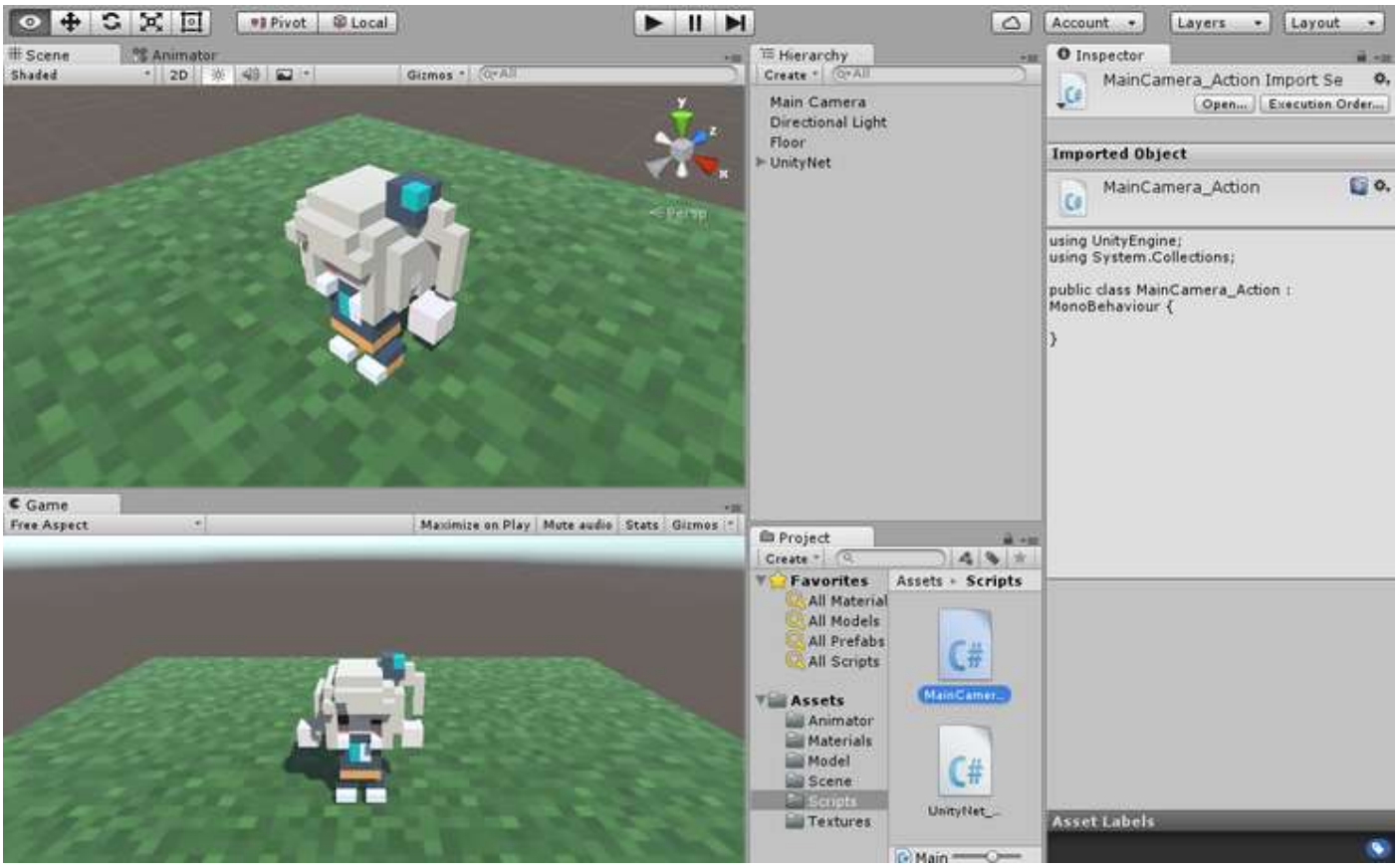
안녕하세요.
골드메탈입니다.

이번 편은 간단하게
캐릭터를 따라 움직이는 카메라를 구현해보겠습니다.



카메라는 비추는 각도만 바뀌어도
게임성이 확 바뀔 정도로 중요한 오브젝트입니다.

우리는 기본적인 액션 게임이니
플레이어을 따라가가며 비추어야겠지요?



메인 카메라에 스크립트를 하나 만듭니다.

```
public class MainCamera_Action : MonoBehaviour {

    public GameObject player;

}
```

public 전역변수를 하나 만듭니다.

이 변수에 플레이어를 담을 것입니다.

다시 유니티로 돌아와서 속성 창을 보면
전역변수가 나타납니다.

초기화를 해주지 않았기 때문에
값이 없다고 하네요.

오브젝트 목록에서 플레이어 캐릭터를
드래그해서 집어 넣습니다.

외부 오브젝트는 이렇게 가져올 수 있답니다.

```
public GameObject player;

void LateUpdate ()
{

}
```

자, 이제 로직을 만들어봅시다.
이번에는 LateUpdate 사이클을 사용할 것입니다.

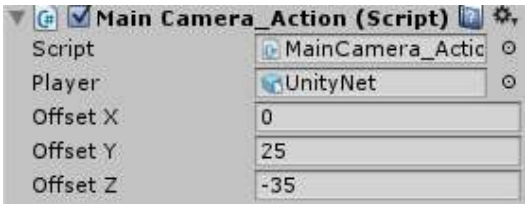
LateUpdate 함수는?
게임 상의 모든 Update 로직을 마친 후,
실행하는 마지막 Update 사이클입니다.

플레이어가 이동한 위치로
따라가야 하는 카메라에겐 안성맞춤이죠.

```
public float offsetX = 0f;
public float offsetY = 25f;
public float offsetZ = -35f;

Vector3 cameraPosition;

void LateUpdate ()
{
    cameraPosition.x = player.transform.position.x + offsetX;
    cameraPosition.y = player.transform.position.y + offsetY;
    cameraPosition.z = player.transform.position.z + offsetZ;

    transform.position = cameraPosition;
}
```

카메라의 위치는 플레이어에서 조금 떨어져 있어야하니까
플레이어 위치에서 적당한 값을 더해서 카메라 위치로 지정해줍니다.

참고로 오브젝트의 **transform**은
내부적으로 선언되어 있어서 따로 변수를 만들지 않아도
바로 쓸 수 있습니다.



public 전역변수의 좋은 점은
유니티에서 바로 값을 바꾸는 것이 가능하다는 점입니다.

게임을 켜고 바로바로 값을 바꾸어
훨씬 빠르게 알맞은 값을 찾을 수 있죠.

이제 카메라가 플레이어를 따라가면서
비춰주게 되었습니다.

여기서 조금 더 부드러운 느낌이면 좋을텐데요.

```
transform.position = Vector3.Lerp(transform.position, cameraPosition, followSpeed * Time.deltaTime);
```

회전에 Slerp 함수가 있다면,
위치에는 **Lerp** 함수가 있습니다.

이전 편에서 했듯이
현재 위치, 목표 위치, 속도를 넣어줍니다.

여기서 속도를 캐릭터 이동 속도보다 느리게 설정하면은..

약간의 딜레이가 생기면서
훨씬 부드러운 카메라 이동이 되었습니다~

열두번째 강좌는 이렇게 마무리되었습니다.
다음 시간에는 이벤트 트리거에 대해 다뤄보겠습니다.
그럼.