[유니티 기초] 20. 마무리

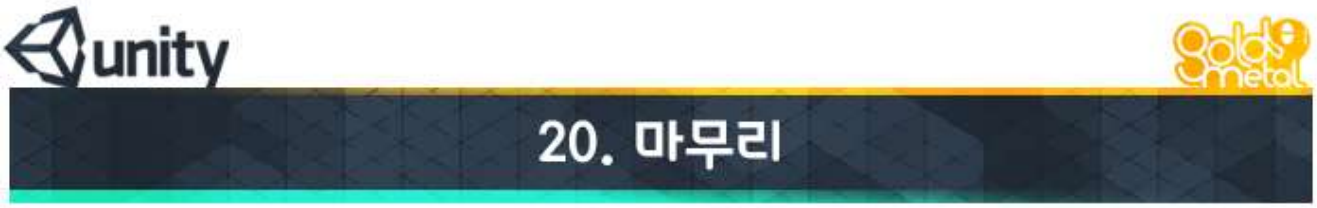
유니티·개발

2015. 11. 26. 0:31

http://blog.naver.com/gold_metal/220550327676

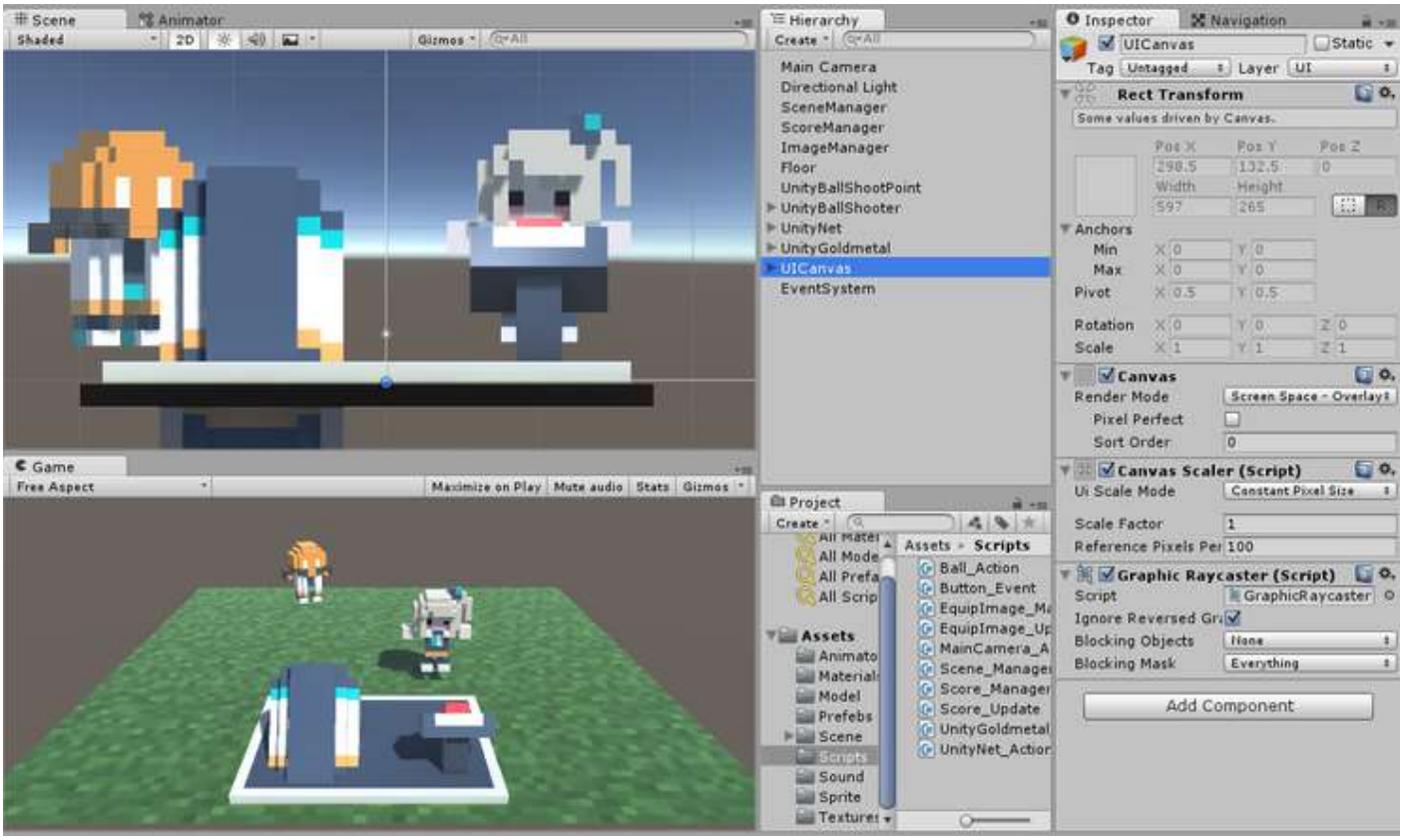
안녕하세요.
골드메탈입니다.

드디어 대망의 마지막 강좌.
마무리 편이 되겠습니다.

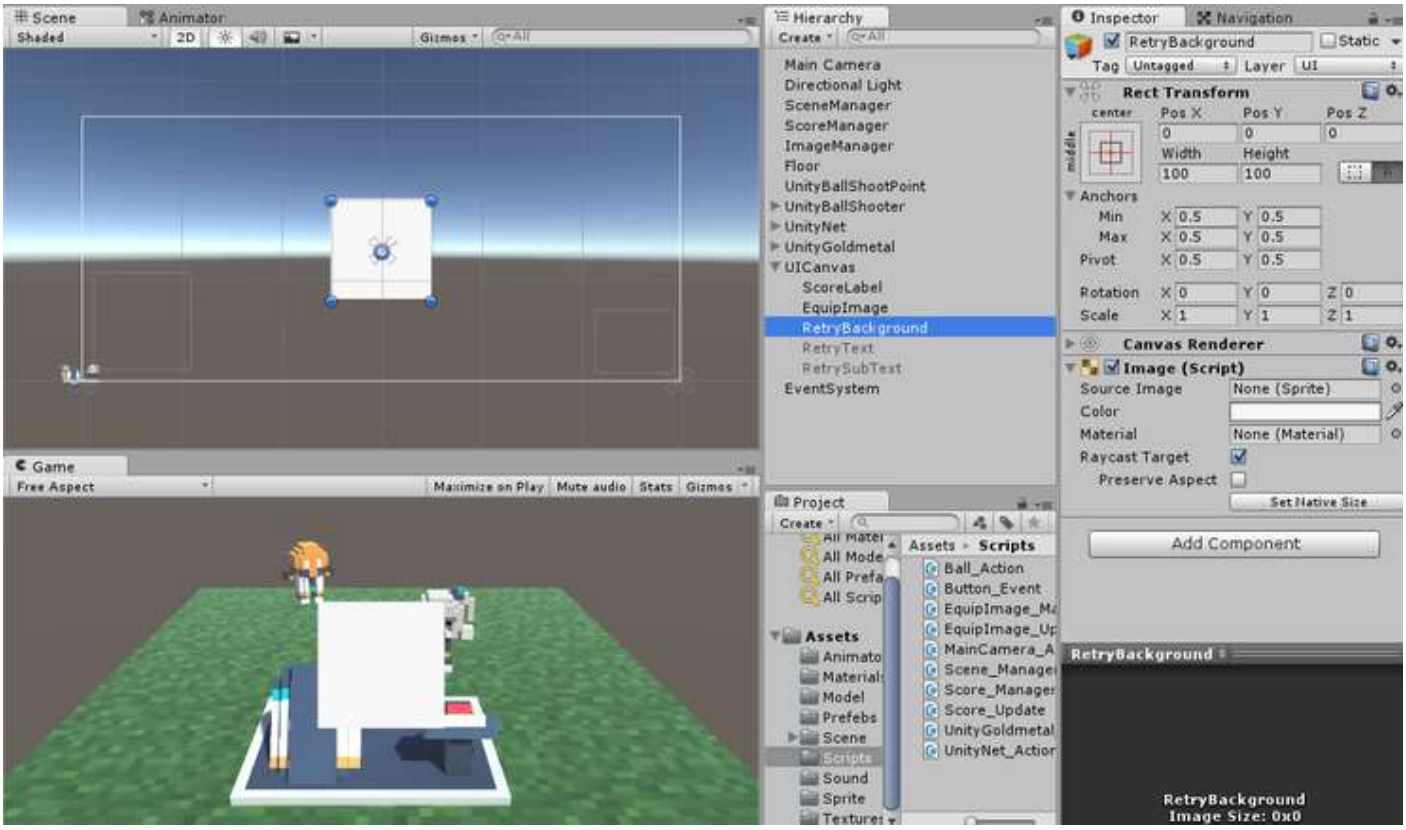


먼저 게임오버 처리부터
구현해보도록 합시다.

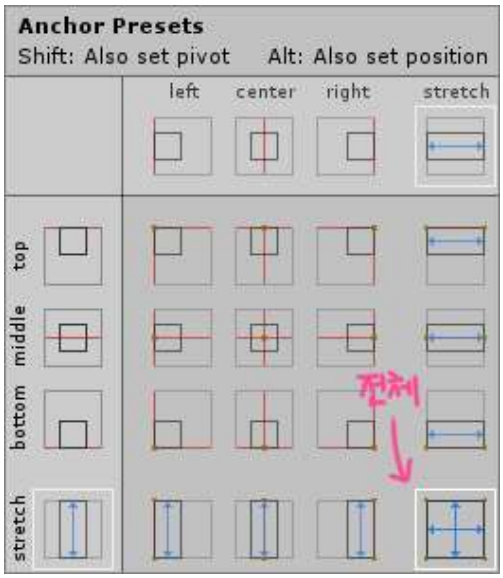
1. 게임오버 UI



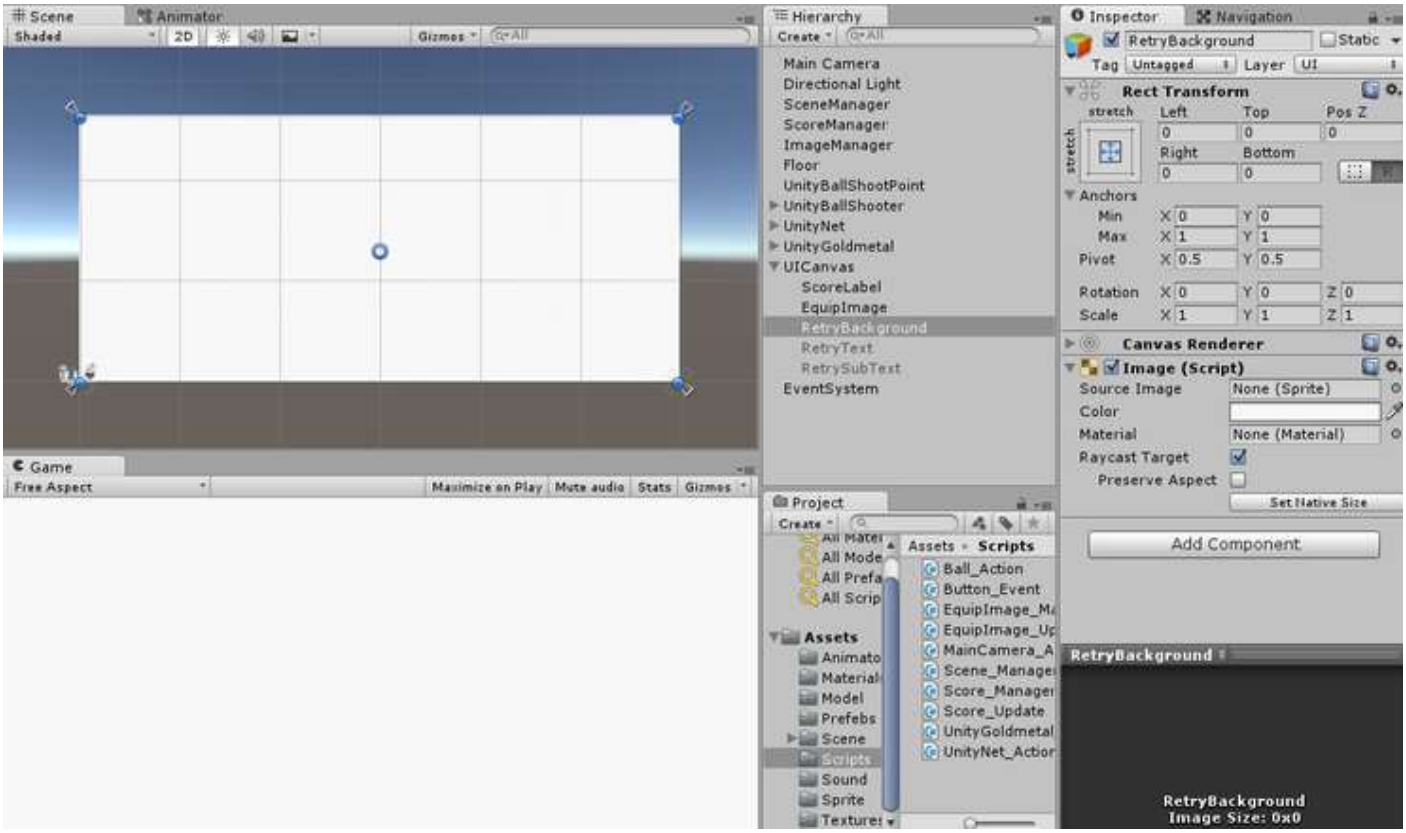
게임오버가 되면
배경이 하얗게 되면서 텍스트가 나타나는 식으로
해보려고 합니다.



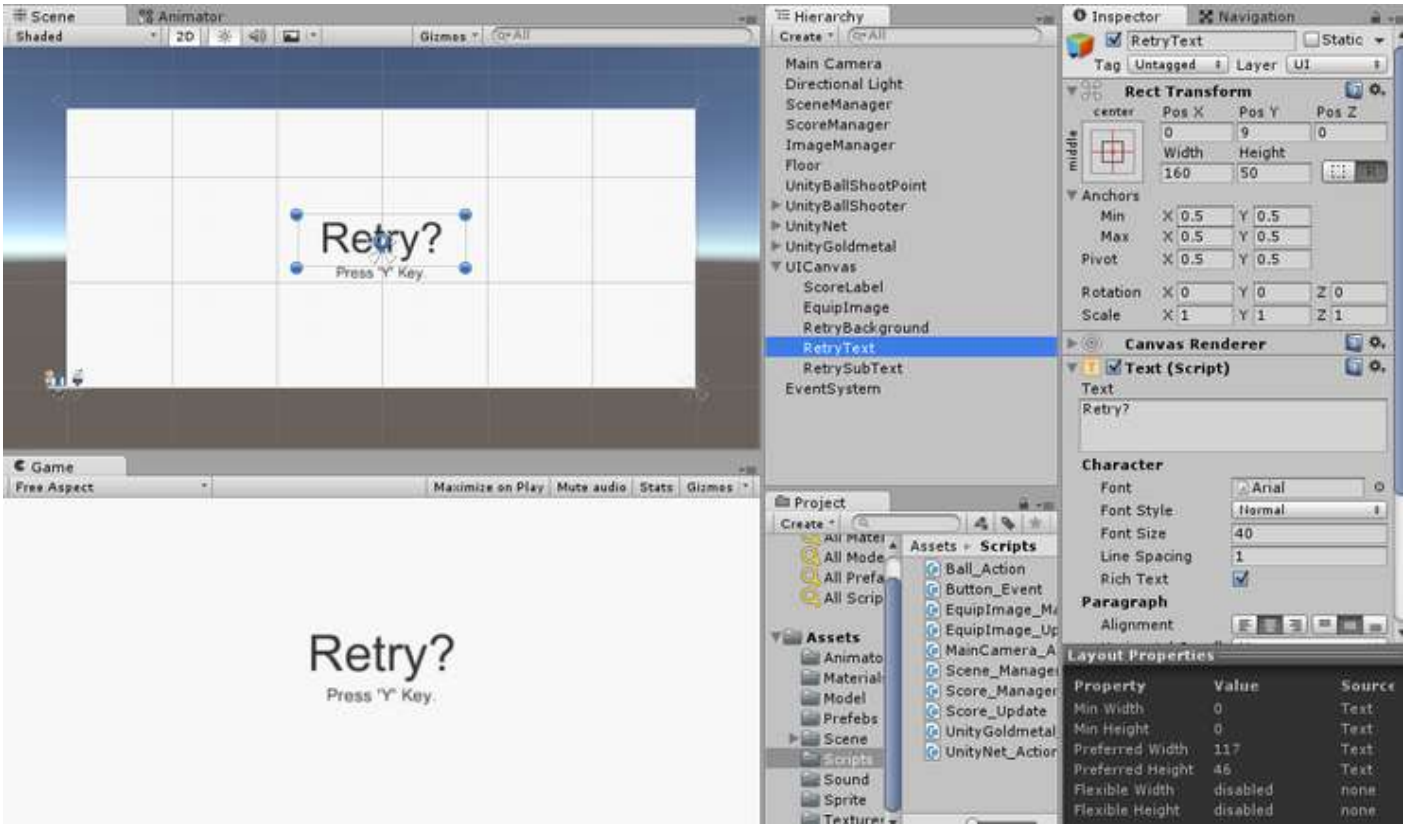
캔버스에 이미지 하나를 추가합니다.
이것이 배경이 될 녀석입니다.



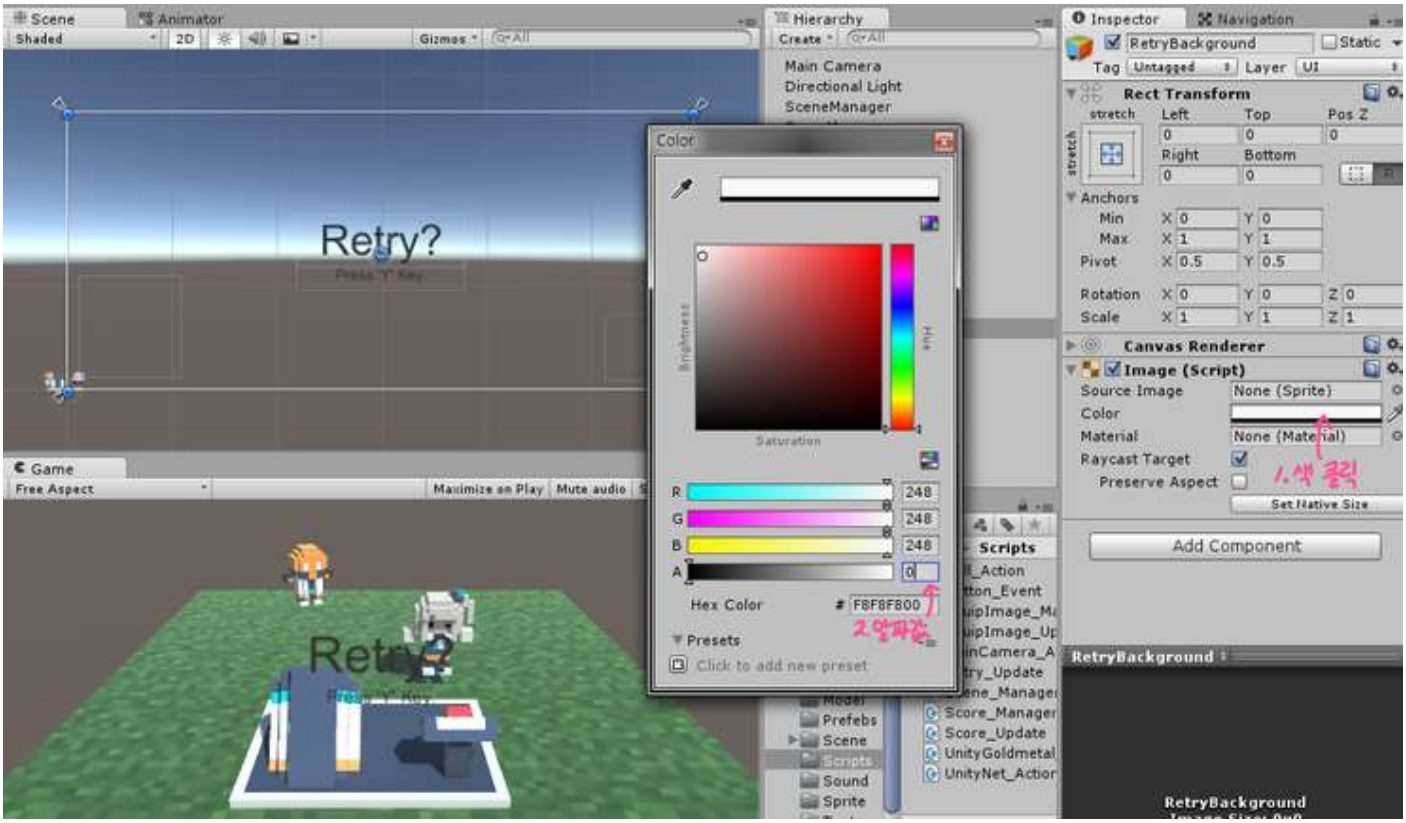
속성 창의 위치 버튼을 누르고 Alt키를 눌러서
전체 확장을 선택합니다.



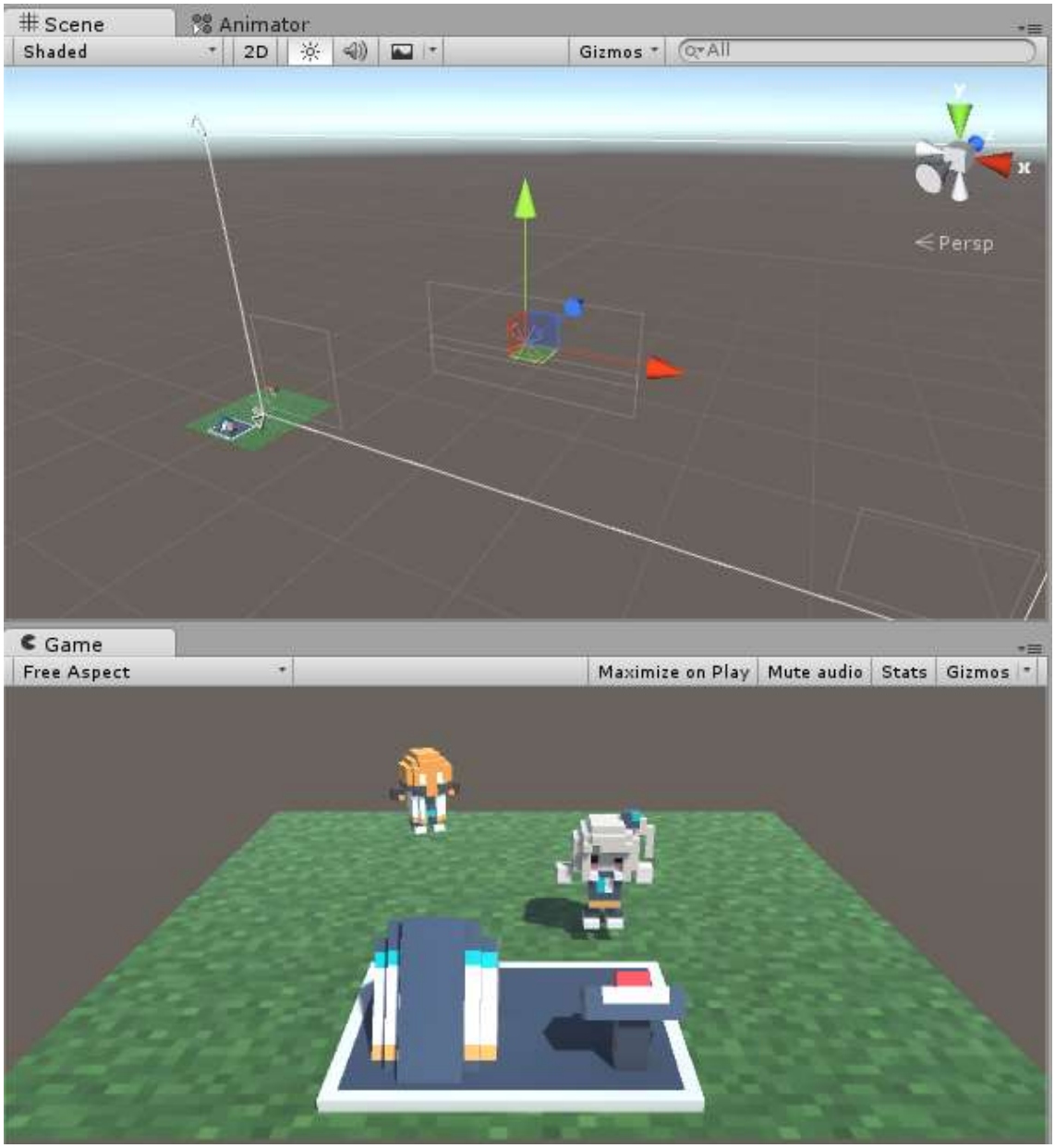
그럼 이렇게 화면 전체가 하얗게
덮어씌어집니다.



그 다음 게임오버 텍스트를 추가해줍시다.
아래 게임 창으로 게임오버 장면이 어떻게 나오는지 확인도 해봅니다.



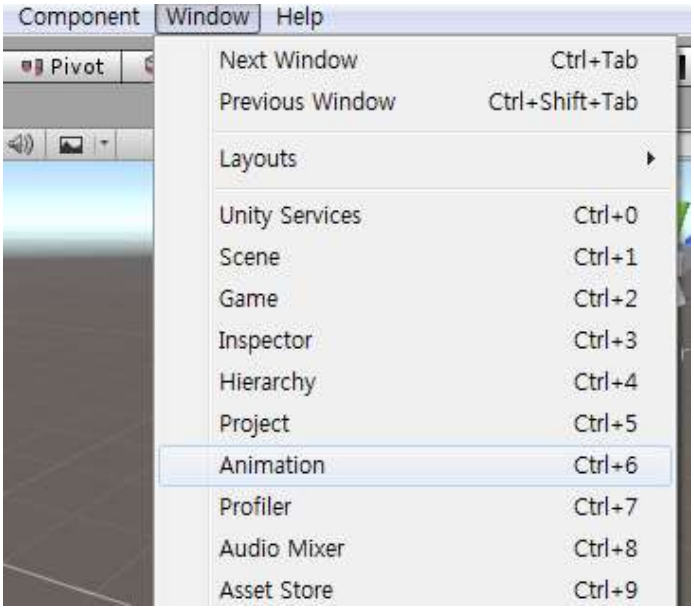
이제 추가했던 UI들의 색상을 클릭하고,
알파 값을 0으로 하여 투명하게 만들어줍니다.



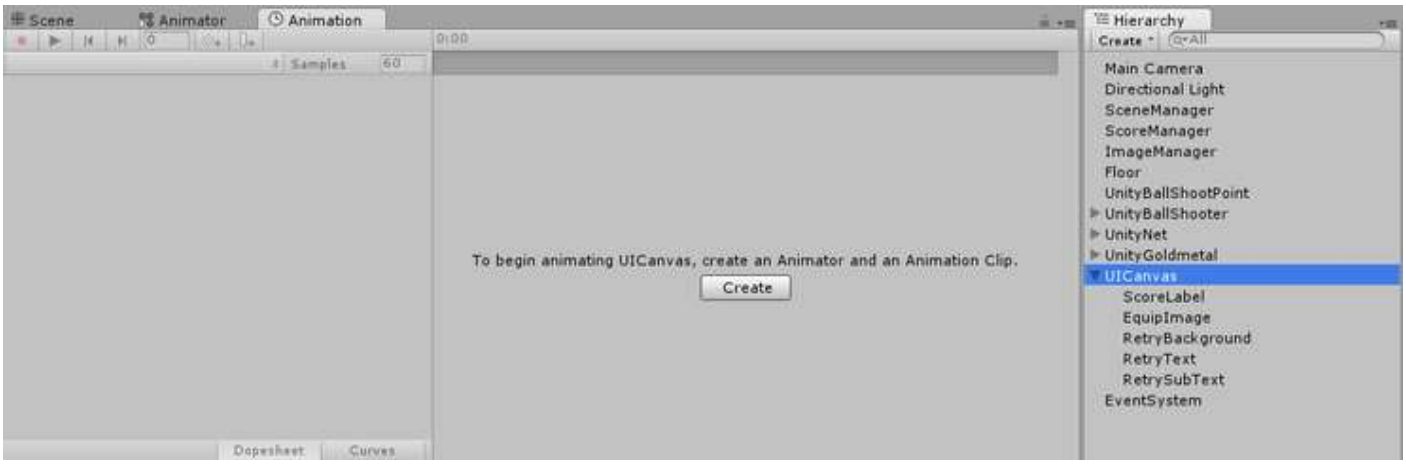
이처럼 숨겨둔다고 보시면 됩니다.
게임 시작할 때 게임오버가 나타나면 안되겠죠?

2. UI 애니메이션

이제 게임오버 되면
UI들이 나타나도록 애니메이션을 넣을 것입니다.

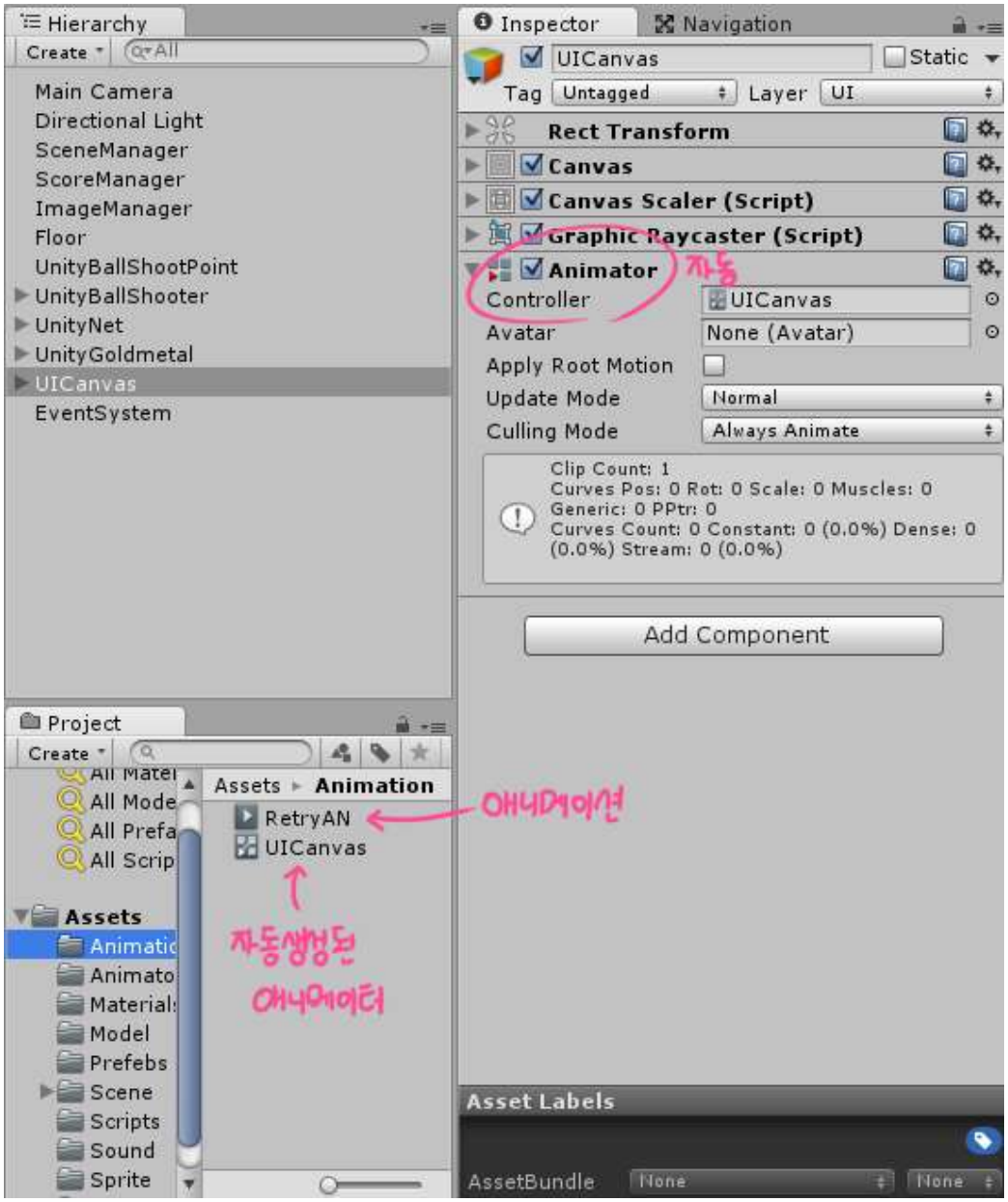


Window - Animation 메뉴로 들어갑니다.



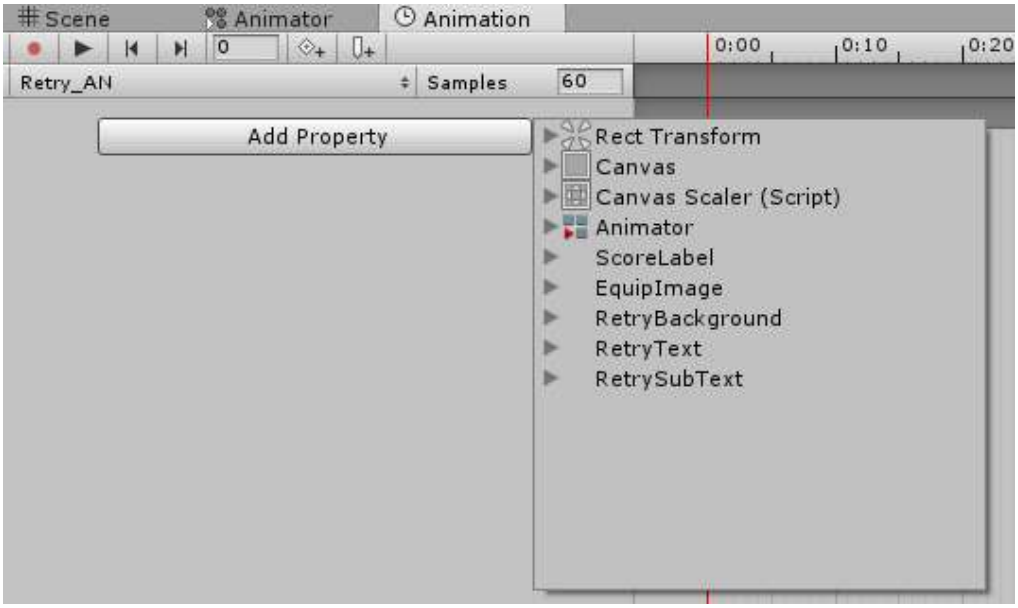
캔버스를 클릭하면 이렇게 애니메이션을 추가하라는
메세지와 함께 버튼이 나타납니다.

버튼을 클릭하면 폴더 선택 창이 나오는데
애니메이션 애셋 폴더를 새로 만들어서 선택합니다.



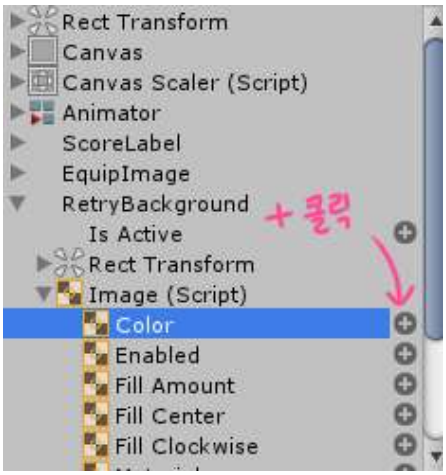
그럼 이처럼 애니메이션 파일과
애니메이터 파일이 생성됩니다.

애니메이션의 속성을 보면
반복 옵션이 있는데 해제해둡시다.

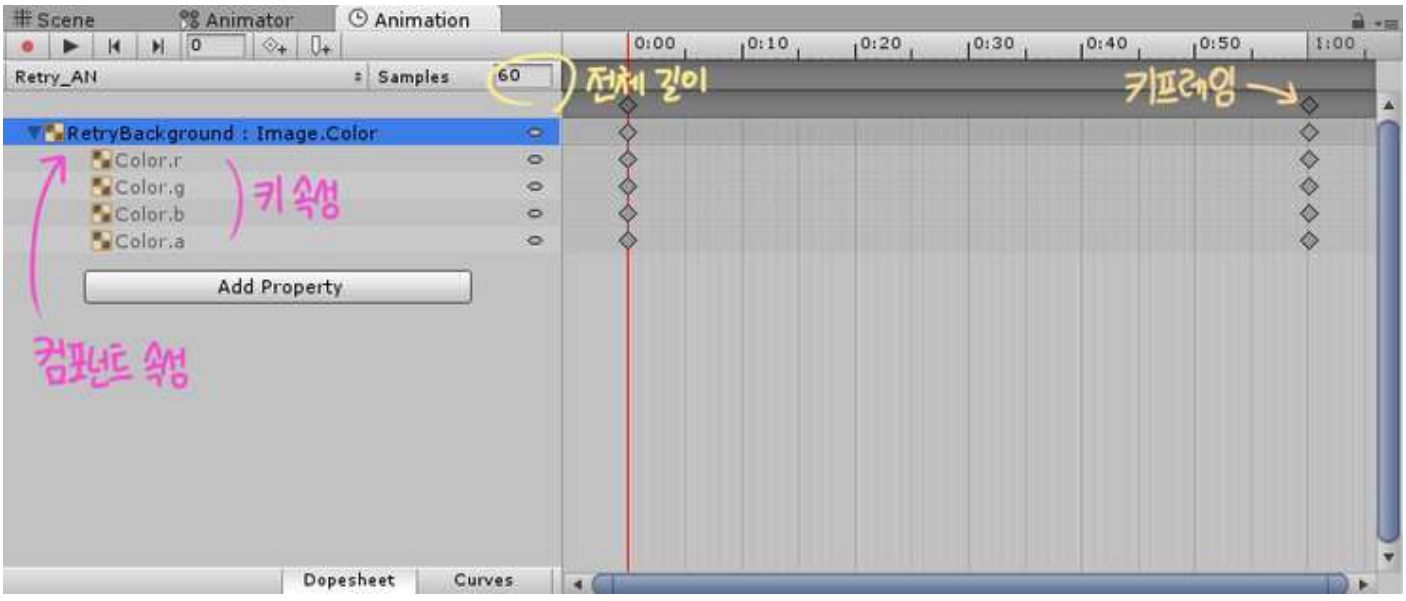


이제 본격적으로 애니메이션을 만들어보도록 하죠.
먼저 **Add Property** 버튼을 눌러봅니다.

그럼 해당 오브젝트 속성 외에
자식으로 포함된 게임 오브젝트들도 함께 나타납니다.

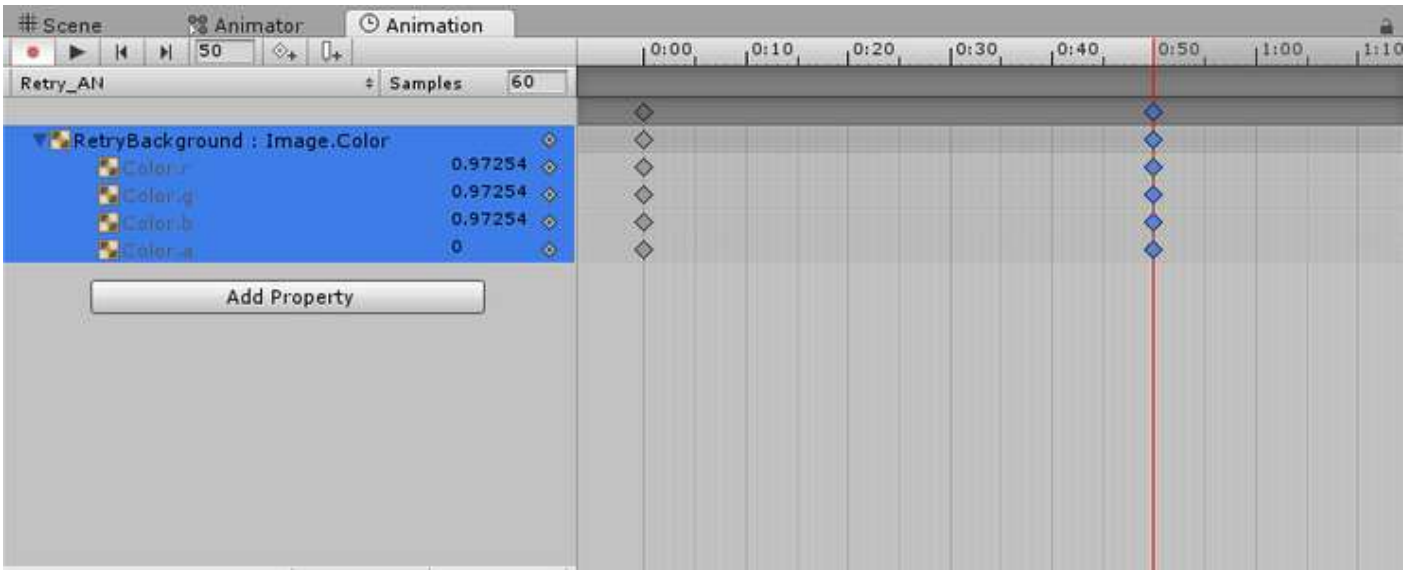


먼저 배경부터 건들여봅시다.
이미지 - 색상 까지 들어간 다음 옆의 + 버튼으로 추가해줍니다.

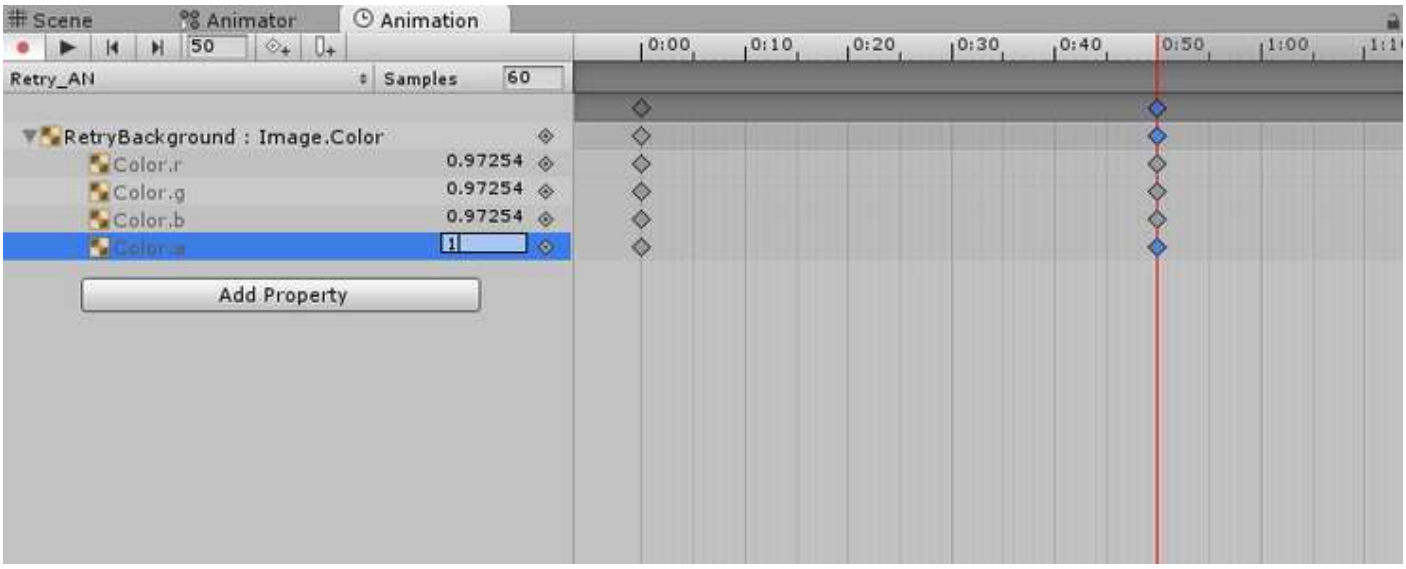


애니메이션은 이처럼 컴포넌트의 속성이 기준입니다.

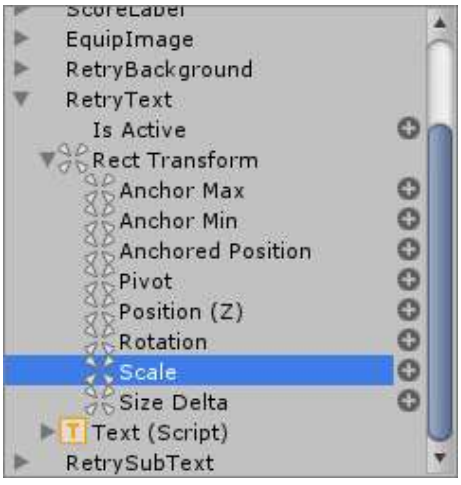
모습은 플래시 혹은 다른 3D툴의 애니메이션과 비슷하네요.



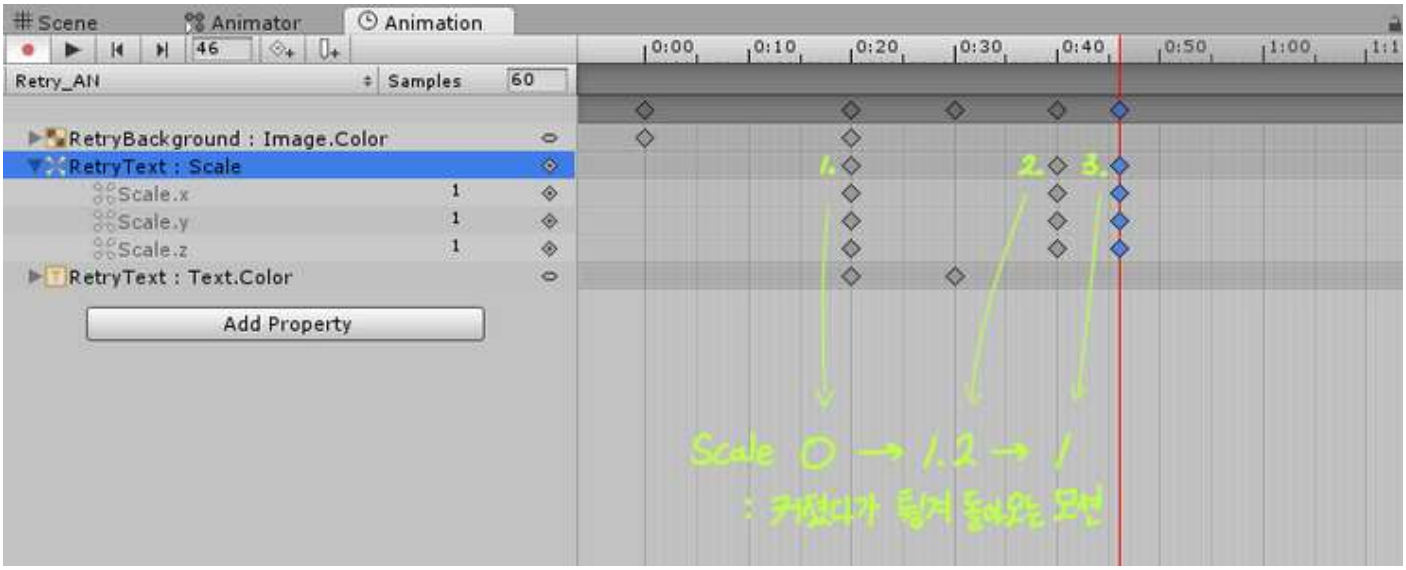
먼저 전체 길이를 조절합니다.
왼쪽의 Samples 숫자를 입력해도 되고
가장 위에 있는 키프레임을 움직여도 됩니다.



마지막 키프레임들 중, 알파만 선택한 다음 1로 늘려줍니다.
이것이 바로 투명한 상태에서 점점 보여지는 **페이드인(Fade-In)**애니메이션이죠.

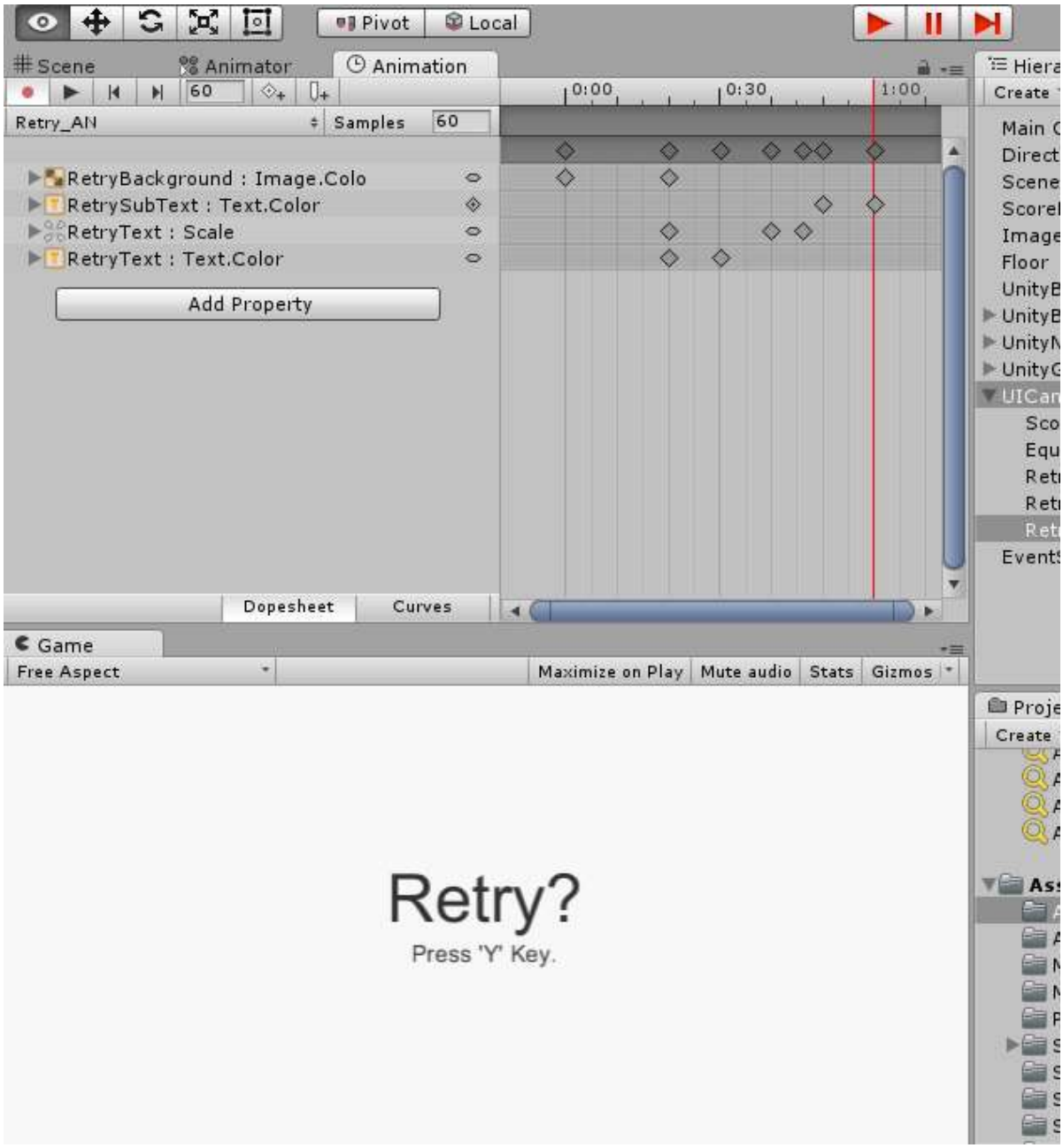


이번엔 크기 애니메이션을 만들어보겠습니다.
텍스트 - 스케일을 선택합니다.



좀 더 유연한 애니메이션을 위해
키프레임 배경을 더블클릭해서 키프레임 하나를 만들어줍니다.

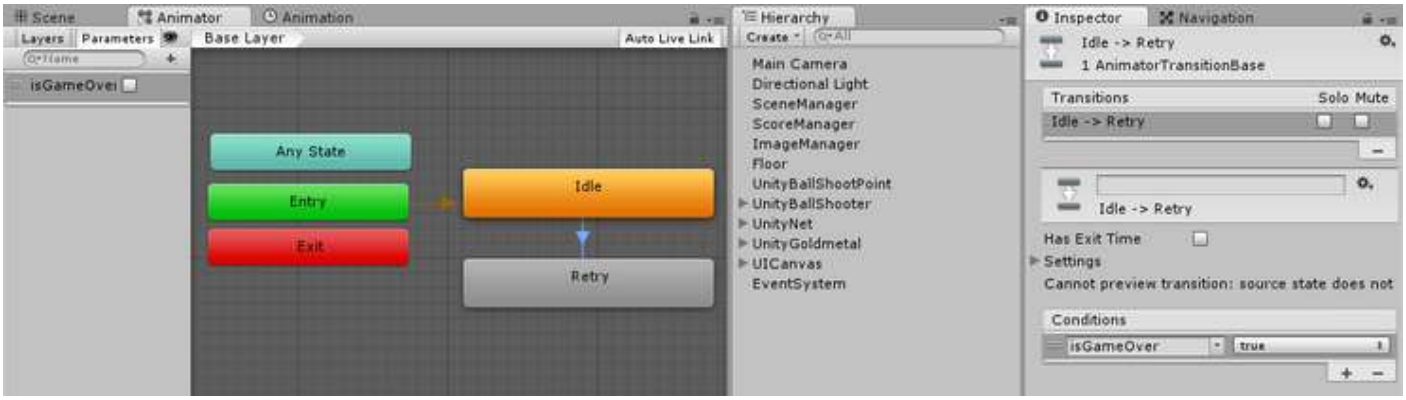
그리고 위 그림처럼 첫째-둘째를 멀리, 둘째와 셋째는 가까이 배치한 후
첫째는 0, 둘째는 1.2, 셋째는 1로 설정합니다.
이러면 커지면서 잠깐 바운싱되는 애니메이션이 되죠.



애니메이션 창의 왼쪽 녹화버튼이 눌러져있는 상태에서는
애니메이션을 직접 재생해볼 수 있습니다.
(맨 위 재생버튼이 붉은색으로 바뀌어 있지요.)

이렇게 바로 확인할 수 있답니다.

애니메이션 편집이 끝났으니
이제 애니메이터로 재생할 수 있도록 설정합니다.



자동으로 만들어진 애니메이터에 들어가서
bool 파라메터에 따라 작동되도록 설정해줍니다.

3. 게임오버 스크립트

이제 실제로 매니저에 의해서 게임오버 플래그가 세워지고,
그에 따라 게임오버 애니메이션을 불러오도록 하겠습니다.

```
public class Scene_Manager : MonoBehaviour {

    public static bool gameOver;

    //-------------------------------------------------------[ Override Function ]

    void Awake ()
    {
        gameOver = false;
    }

    void Update ()
    {
        if(gameOver && Input.GetKeyDown(KeyCode.Y)){

        }
    }
}
```

장면을 관리하는 매니저를 만들고,
게임오버 변수를 하나 둡니다.

업데이트에 게임오버된 후 Y키를 누르는 로직이 있는데
나중에 여기에 다시 시작하기 로직을 넣을 것입니다.

```
public class Retry_Update : MonoBehaviour {

    Animator animator;

    void Awake ()
    {
        animator = GetComponent<Animator> ();
    }


    void Update ()
    {
        if(Scene_Manager.gameOver){
            animator.SetBool("isGameOver",true);
        }
    }
}
```

애니메이션을 만들었던 UI에도 스크립트를 만들어서
게임오버 변수를 항시 체크하여 애니메이션을 실행하도록 합니다.

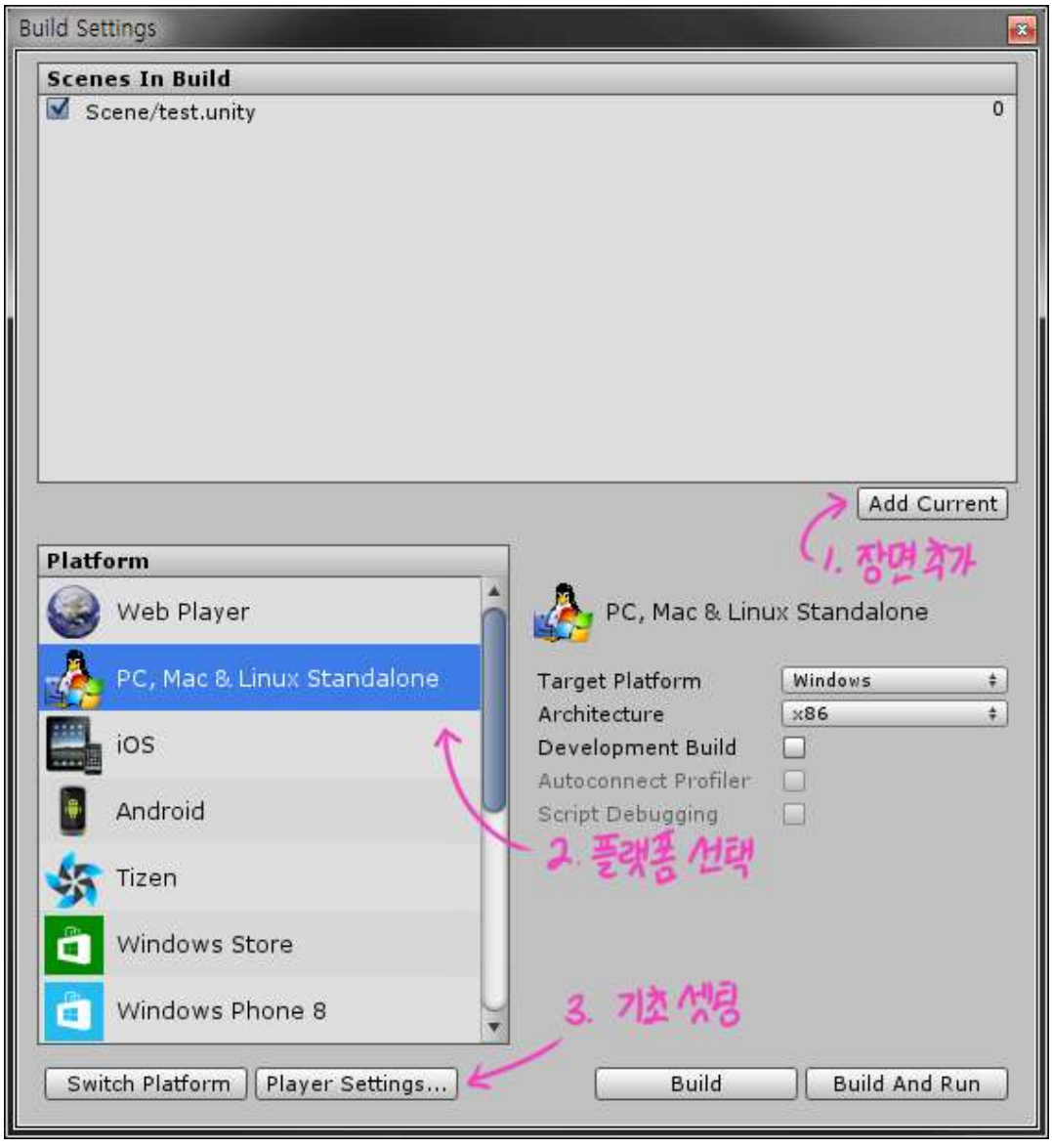
```
void onDeath ()
{
    Score_Manager.score += 10;
    //Destroy(gameObject, 0f);
    Invoke("onGameOver",0.5f);
}

void onGameOver ()
{
    Scene_Manager.gameOver = true;
}
```

필자는 NPC를 맞추면 게임오버 되도록
설정해두겠습니다.

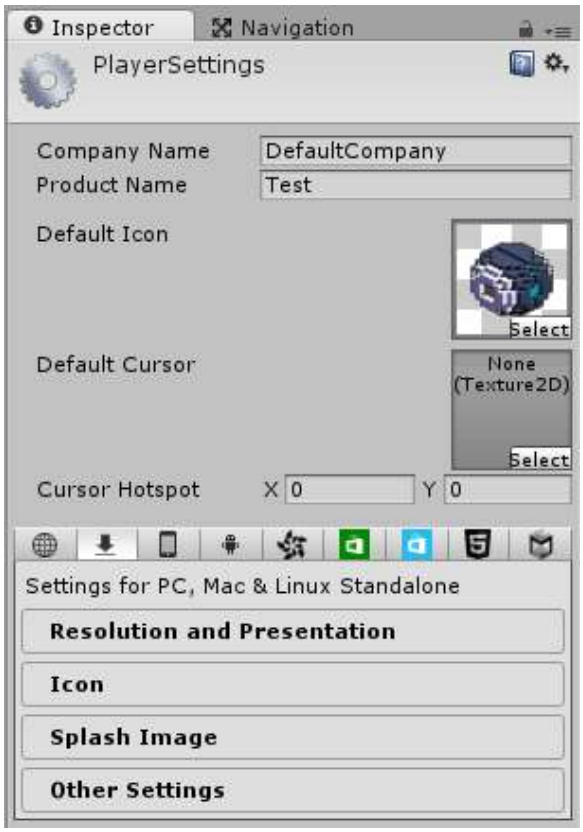
```
public class Scene_Manager : MonoBehaviour {

    public static bool gameOver;

    //-------------------------------------------------------[ Override Function ]

    void Awake ()
    {
        gameOver = false;
    }

    void Update ()
    {
        if(gameOver && Input.GetKeyDown(KeyCode.Y)){
            Application.LoadLevel(Application.loadedLevel);
        }
    }
}
```

```
public static void LoadLevel (
  int index
)

Summary
Loads the level by its name or index.
```

그리고 마지막으로
게임 재시작을 위해 **Application.LoadLevel(장면인덱스)**함수를 사용합니다.
우리는 현재 장면을 다시 불러오기 위해 **Application.loadedLevel** 을 넣어줍니다.



이왕 마무리된 김에 아예 실행 파일로 빌드해봅시다.
File - Build Settings 메뉴로 가서 장면, 플랫폼을 설정해줍시다.



Player Settings 에서는 실행파일 아이콘과
기본 커서를 설정할 수 있습니다.

[유니티 기초] 20. 마무리
4,201

00:00 00:37 480p

드디어 게임이 완성되었어요!
이 게임은 첨부파일로 올려두겠습니다.

이렇게하여!
유니티 기초편 강좌를 마치게 되었습니다.

여기까지 따라와주셔서 감사하고,
유니티 게임 개발이 어떻게 되는지
전반적으로 살펴보았습니다.

잠시 쉬는 시간을 가진 뒤,
유니티 중급 강좌를 시작하도록 하겠습니다.
그럼.