

- ◆ [Toy Apps](#)
- ◆ [License](#)
- ◆ [들어가기 전에](#)
- ◆ [프로젝트 프리뷰](#)
- ◆ [프로젝트 생성](#)
 - [프로젝트 생성](#)
 - [리소스 import](#)
 - [참고](#)
- ◆ [주인공 생성](#)
- ◆ [총알 발사](#)
 - [Scene에 총알 생성](#)
 - [Bullet 스크립트 생성](#)
 - [총알의 이동](#)
 - [참고](#)
 - [Script 적용](#)
 - [동영상](#)
 - [Bullet Prefab 만들기](#)
 - [동영상](#)
 - [참고](#)
 - [마우스로 총알발사](#)
 - [동영상](#)
 - [참고](#)
 - [총알의 각도 조절](#)
 - [동영상](#)
 - [참고](#)
- ◆ [적 생성](#)
 - [적의 이동](#)
 - [다이나믹한 적의 이동](#)
 - [동영상](#)
 - [참고](#)
 - [다이나믹한 적의 위치 설정](#)
 - [동영상](#)
 - [참고](#)
 - [적의 위치 선정](#)
 - [동영상](#)
 - [참고](#)
- ◆ [적 제거](#)
 - [적 수정](#)
 - [동영상](#)
 - [참고](#)
 - [총알 수정](#)
 - [동영상](#)

- 참고
 - [적이 화면 밖으로 나갈때 제거 동영상](#)
- ◆ [게임오버](#)
 - [동영상](#)
 - [참고](#)
- ◆ [소리추가](#)
 - [총을 쏠 때 소리 동영상](#)
 - [참고](#)
 - [게임오버 소리](#)
- ◆ [배포](#)
 - [참고](#)

◆ Toy Apps

email : store@toy-apps.com

asset store : <https://www.assetstore.unity3d.com/en/#!/publisher/7834>

◆ License

<http://creativecommons.org/licenses/by-nc/2.0/>

◆ 들어가기 전에

이 프로젝트는 유니티 2d기능을 이용한 간단한 예제 프로그램으로, c#으로 이루어져 있다. 스텝 바이 스텝으로 간단한 게임을 만들수 있도록 하였지만 기본적인 유니티 사용법이라던가 c# 문법에 관한 설명은 쓰여 있지 않다.

들어가기 전에 이곳들을 참고하면 도움이 될 것이다.

+ Unity document : <http://docs.unity3d.com/Manual/UnityBasics.html>

+ Unity tutorial : <http://unity3d.com/learn/tutorials/modules>

+ c# Tutorial : <http://www.csharp-station.com/Tutorial/CSharp/SmartConsoleSetup.aspx>

문의 사항이 있다면 store@toy-apps.com로 연락을 주면 된다. 나도 아직 유니티에 대해 공부하면서 제작을 하고 있기 때문에 모든 것을 대답해 줄 수는 없겠지만 아는 범위 내에서 성심성의껏 답변을 해주도록 하겠다 (문의는 영어이나 한글로 해주면 된다.)

이 프로젝트 이외에도 [이곳](#)에 들어오면 간단한 다른 프로젝트들도 만나볼 수 있다. 아직은 유료 어셋이 많지만 이 프로젝트를 시작으로 무료 프로젝트도 많이 올릴 예정이니 종종 들려주기 바란다.

◆ 프로젝트 프리뷰

이 튜토리얼 게임은 몰려오는 적에게 총알을 쏘서 경계선을 넘어가지 않게 하는 게임이다.

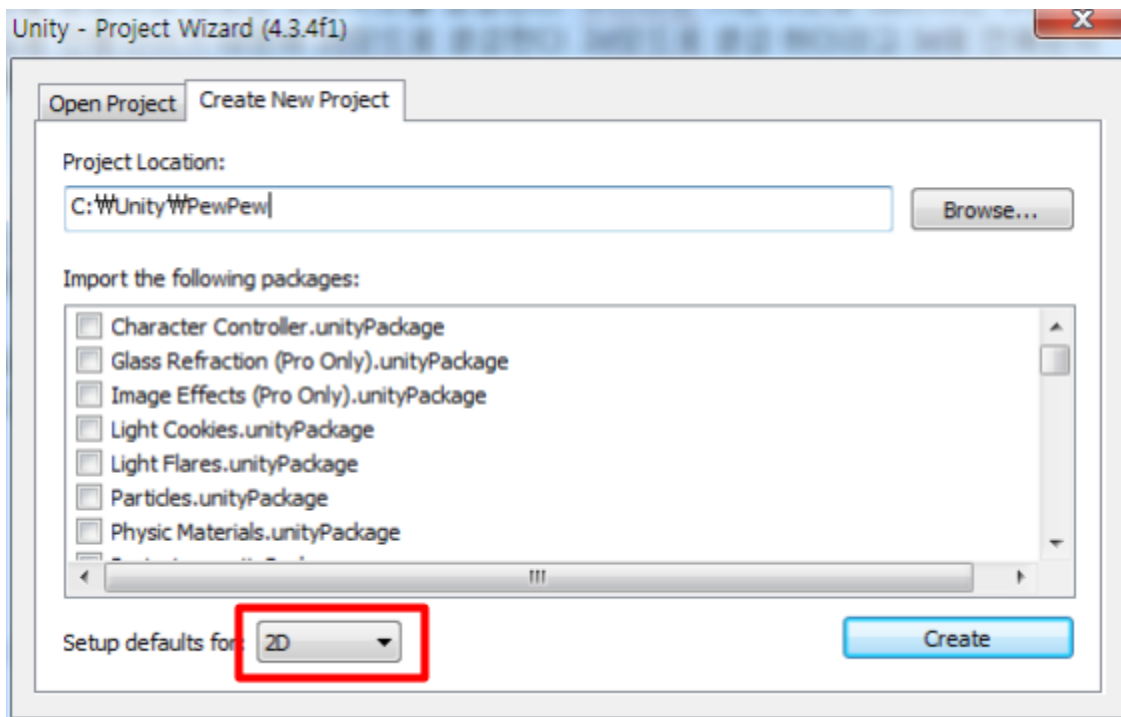
게임플레이 데모 : <http://toyapps.github.io/PewPewPlayer/>

플레이영상 :

http://www.youtube.com/watch?v=lrp3it8Udxc&list=PLFXLeC2Hh_3db7XFynySEMJkkQ6umELpC&index=1

◆ 프로젝트 생성

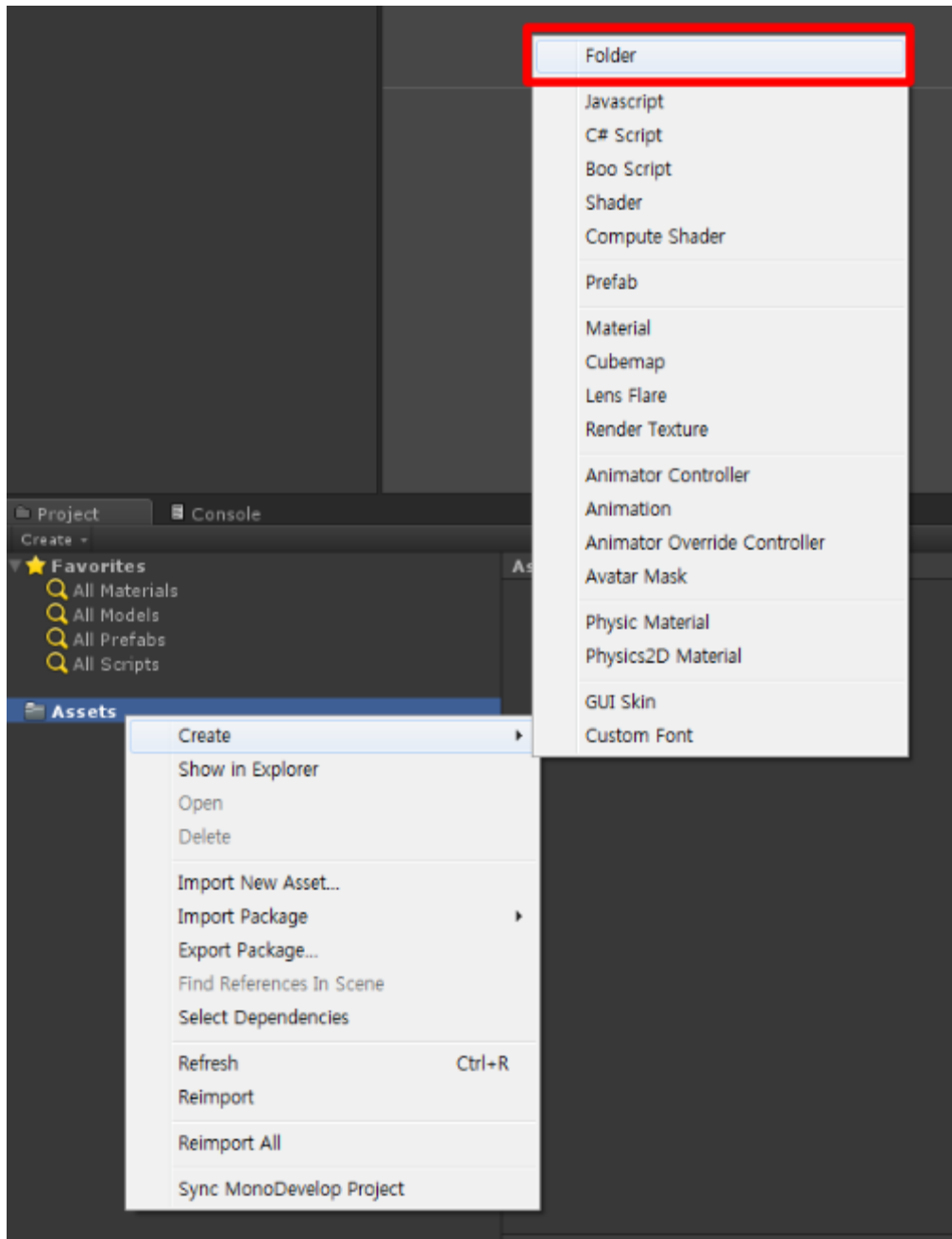
프로젝트 생성



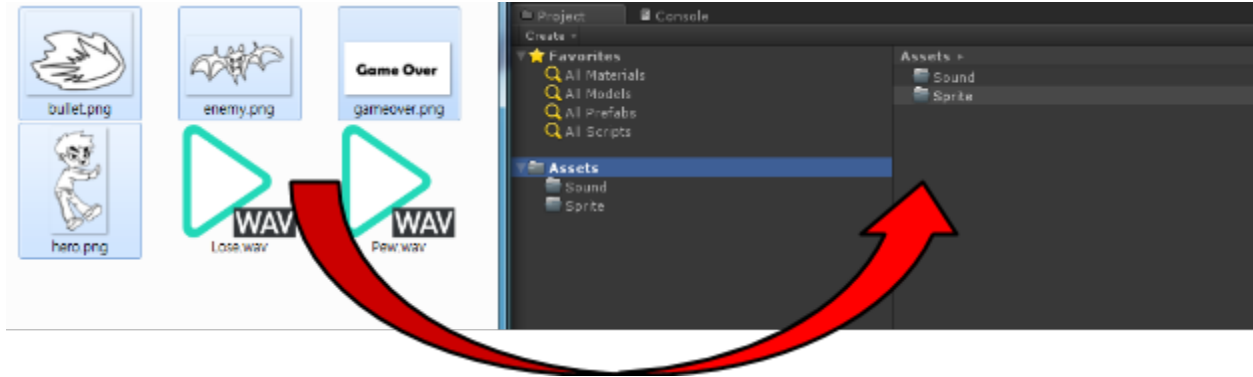
유니티를 실행하고 새로운 프로젝트를 생성한다. 유니티는 기본적으로 3d이지만 우리는 2d게임을 만들 것이기 때문에 2d모드를 선택하고 Create 버튼을 눌러 프로젝트를 생성한다.

리소스 import

유니티에서는 파일들을 프로젝트 내에서 관리한다. 리소스 파일들도 유니티 프로젝트 내에서 관리가 되는데 일단 import 과정을 거쳐야 한다.



먼저 Assets폴더 안에 Create -> Folder를 선택하고 Sprite와 Sound폴더 (폴더 이름은 자기 마음대로 지어도 상관없다) 를 생성하자.
포함되어 있는 resource.zip의 압축을 풀면 그림 파일들과 소리 파일들이 들어있는 것을 볼 수 있다.

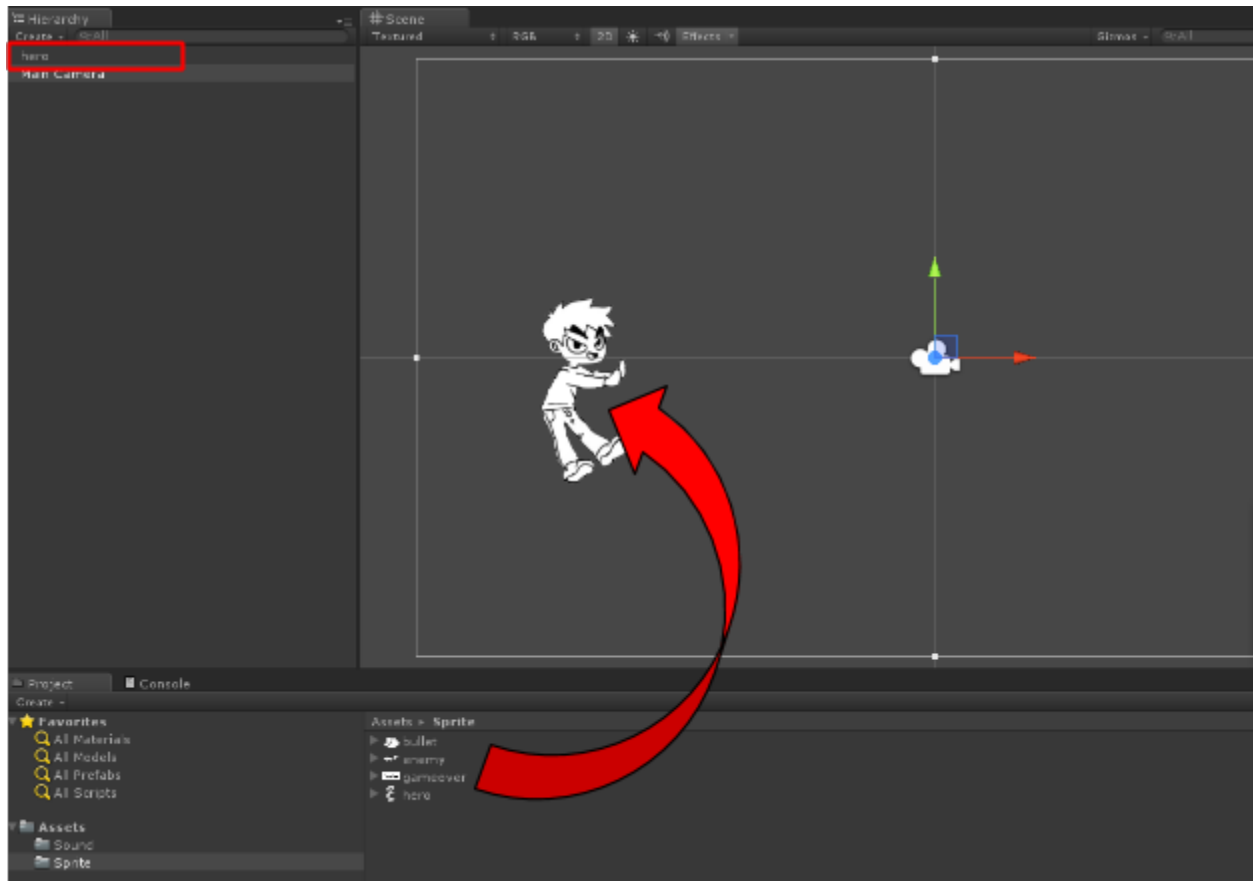


그림파일들(png)들은 Sprite폴더 안에, 소리파일(wav)파일들은 Sound 폴더에 Drag & Drop 해주면 쉽게 import를 완료할 수 있다. 이 파일들을 이동하거나 복사하는 작업을 절대로 해서는 안된다. Assets 안에 있는 파일들은 유니티 내에서 따로 관리가 되고 있다. 만약 유니티 안에서가 아닌 다른곳에서 이 파일들을 이동하거나 이름을 바꾸게 되면 Asset들간의 링크가 깨지게 되어 돌이킬 수 없는 결과를 맞이할 수 있으니 그런 작업은 꼭 유니티 내에서만 꼭 하도록 하자.

참고

<http://docs.unity3d.com/Manual/ImportingAssets.html>

◆ 주인공 생성



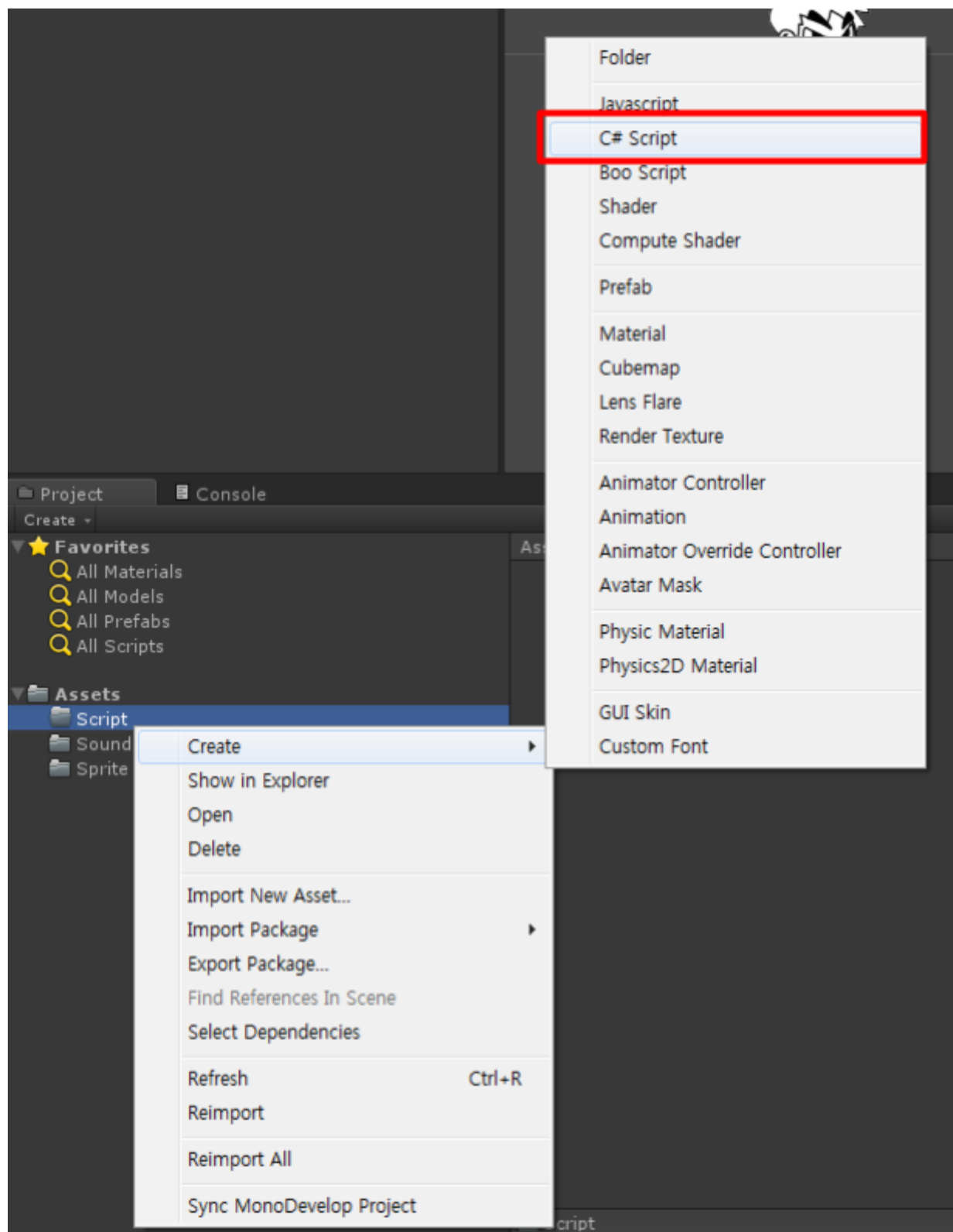
좀 전에 import한 그림 파일들 중에 hero.png 파일을 scene에 Drag & Drop 해준다. 그렇게 되면 Scene에 Hero가 생긴다. Hierarchy에 Hero가 생긴것을 볼 수 있다.

이때 주의할 것은 Scene에서 보이더라도 Object 들이 Camera 에 비치지 않으면 플레이 할 때 보이지 않을 수 있으니 카메라 안에 들어와 있는지 잘 확인한다.

◆ 총알 발사

Scene에 총알 생성

맨 처음으로는 영웅에게서 총알이 나가도록 할 것이다. Hero를 만든 것과 마찬가지로 Drag&Drop 을 통해 Scene에 추가해준다.



Bullet 스크립트 생성

Script 폴더를 만들고 Create -> C# Script를 선택한 후 Bullet이라고 이름을 지어준다. 이 파일을 더블클릭 하게되면 MonoDevelop이 실행되고 방금 만든 스크립트가 보인다.

```
using UnityEngine;
using System.Collections;

public class Bullet : MonoBehaviour
{
    // 1
    void Start ()
    {
    }
    // 2
    void Update ()
    {
    }
}
```

- 1) 스크립트가 실행될 때 처음 실행된다.
- 2) 매 프레임마다 실행된다.

총알의 이동

먼저 총알이 오른쪽으로 움직이도록 여기에 내용을 추가할 것이다.

```
public class Bullet : MonoBehaviour
{
    // 1
    public float m_speed = 0.3f;

    void Start ()
    {
        // 2
        Destroy (gameObject, 3.0f);
    }

    void Update ()
    {
        // 3
        transform.position += transform.right * m_speed;
    }
}
```

- 1) 총알이 움직이는 스피드를 설정하기 위해 변수를 설정한다. public으로 Scene의 Inspector에서 쉽게 수정 가능하다.
- 2) 총알이 생성된 후 무한대로 날아가지 않도록 Destroy함수를 이용하여 생성된 총알을 삭제한다

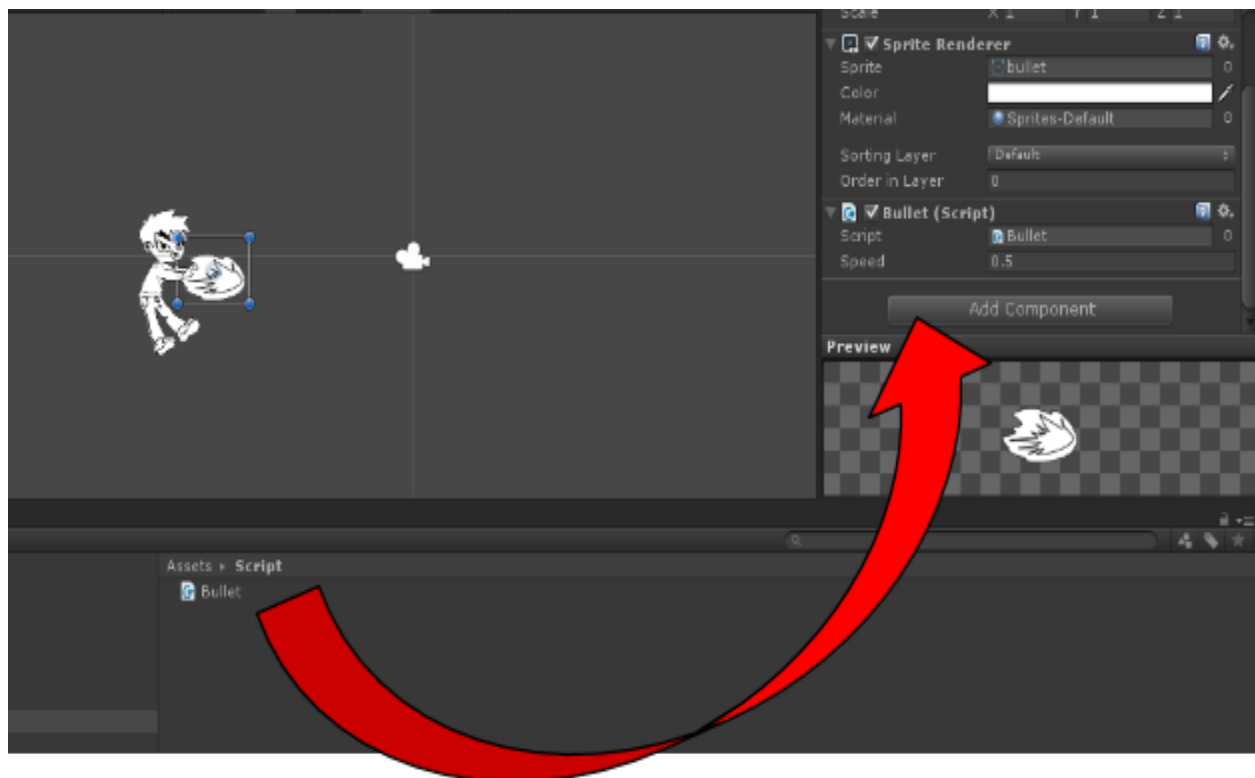
3) 총알이 오른쪽으로 m_speed만큼씩 이동하도록 한다.

참고

<http://unity3d.com/learn/tutorials/modules/beginner/scripting/variables-and-functions>

http://docs.unity3d.com/412/Documentation/ScriptReference/index.Member_Variables_26_Global_Variables.html

Script 적용



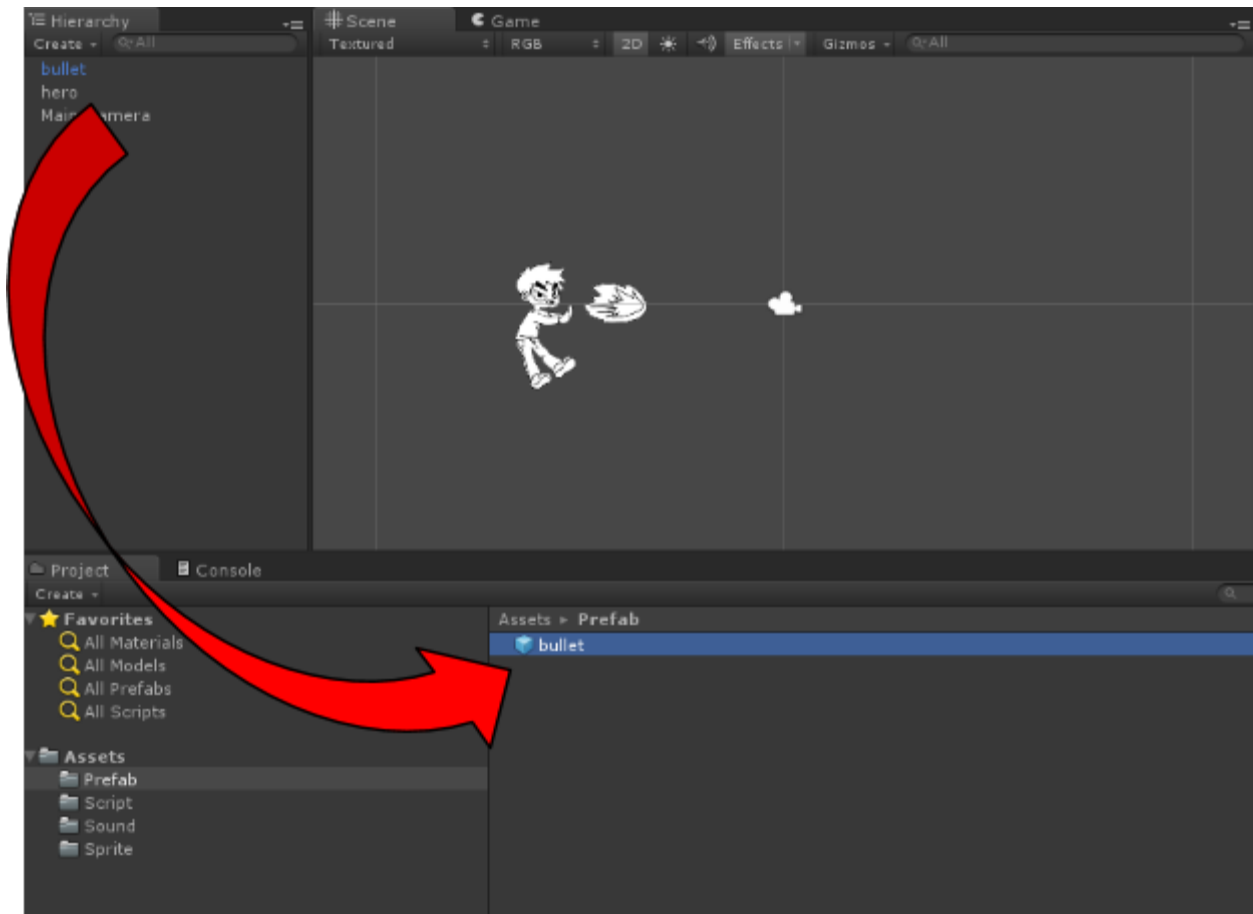
방금 만든 Script를 아까 만든 Sprite에 적용하면 Script가 작동하게 된다. Bullet Script를 Drag & Drop하게 되면 스크립트가 적용된다. Inspector에 있는 Speed 변수를 조정하면 실행할 때 총알 속도가 변하는 것을 볼 수 있다. 이 속도는 원하는 만큼 적당히 설정하도록 하자.

동영상

http://www.youtube.com/watch?v=odMPtM512MQ&index=1&list=PLFXLeC2Hh_3db7XFynySEMJkkQ6umELpC

Bullet Prefab 만들기

시작을 하게되면 총알이 한발만 나가기 때문에 게임으로 쓰기에는 부족해 보인다. 이번에는 화면을 클릭하면 주인공이 총알을 쏘도록 해보자.



먼저 여러발이 나가게 하기 위해서는 방금 만든 Bullet을 Prefab으로 만들어야 한다. 먼저 Prefab라는 폴더를 만들고 Hierarchy에 있는 bullet을 Drag&Drop한다. 이렇게 하면 Prefab이 생성된다. Prefab을 만들면 복사본을 몇개든 생성 가능하다.

테스트로 생성된 bullet prefab를 scene으로 Drag&Drop하면 복제본이 생성되는 것을 볼 수 있다.

동영상

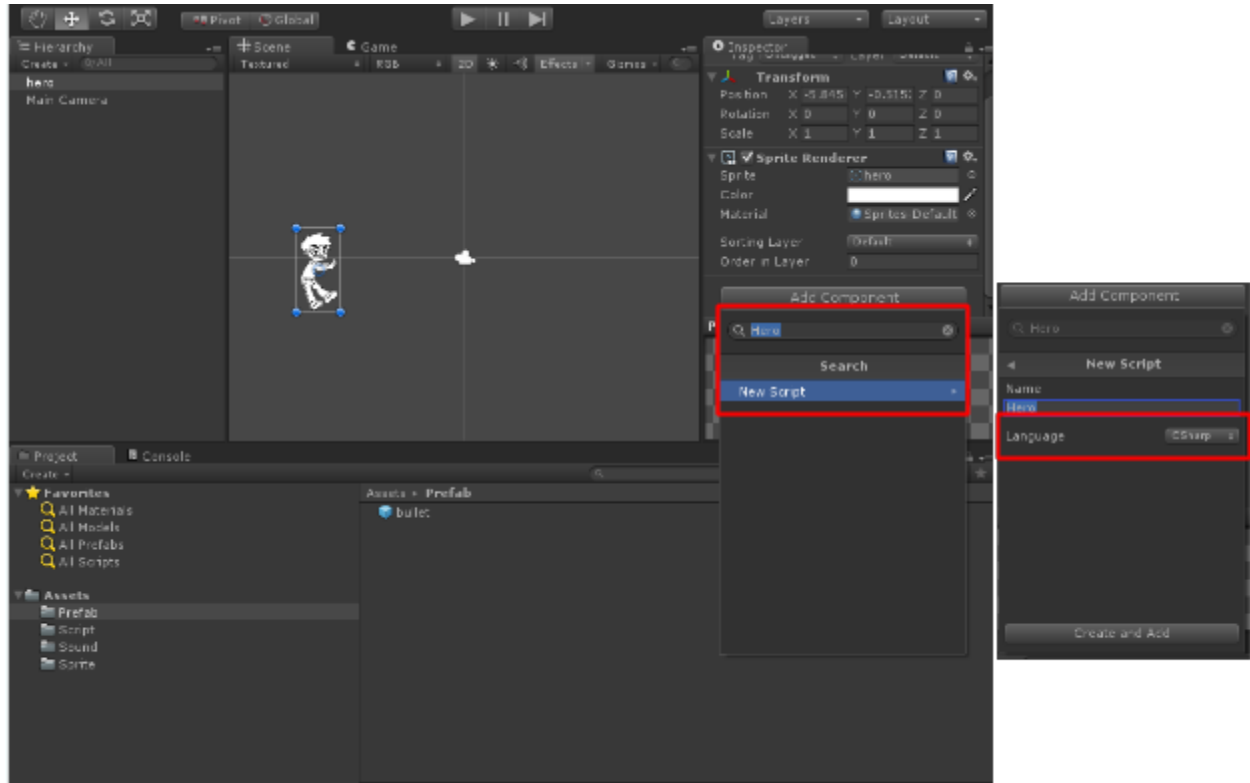
http://www.youtube.com/watch?v=2YFd0emLIw&index=3&list=PLFXLeC2Hh_3db7XFynySEMJkKQ6umELpC

참고

<http://docs.unity3d.com/Manual/Prefabs.html>

마우스로 총알발사

이번에는 Hero에 Script를 추가하여 마우스로 총알을 발사할 수 있도록 할 것이다. 이번에는 아까와는 다른 방법으로 Script를 추가해보자.



Hero를 선택한 후에 오른쪽 Inspector에서 Add Component를 누른다. 만들고 싶은 Script 이름을 넣게 되면 New Script라는 항목이 생기는데 그것을 클릭하고 Create and Add를 클릭해서 Script를 생성한다. 이렇게 만들어진 Script는 Assets의 바로 아래 생성되므로 Script폴더로 이동해준다.

앞에서도 언급했지만 유니티 파일의 이동은 꼭 유니티 안에서만 이루어져야 한다.

추가한 Script를 더블클릭해서 에디터를 열고 코드를 추가해보도록 하자.

```
using UnityEngine;
using System.Collections;

public class Hero : MonoBehaviour
{
    // 1
    public GameObject m_bullet;

    void Start ()
    {
    }
}
```

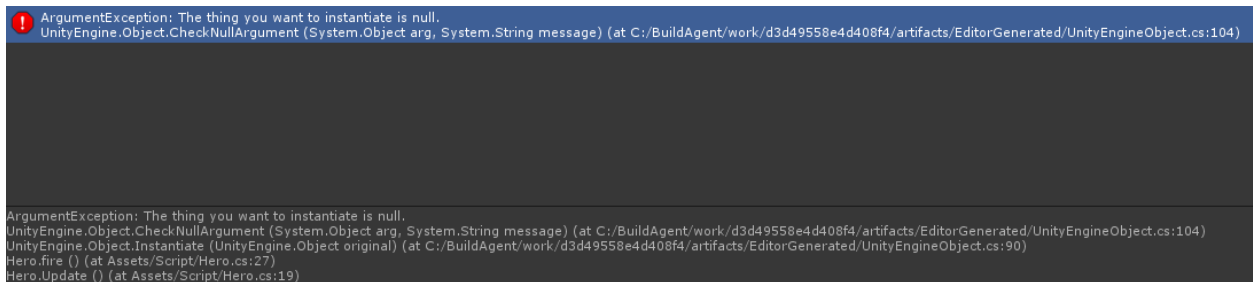
```

void Update ()
{
    // 2
    if (Input.GetMouseButtonDown(0))
    {
        fire();
    }
}

private void fire()
{
    // 3
    GameObject bullet = Instantiate (m_bullet) as GameObject;
    // 4
    bullet.transform.position = transform.position;
}
}

```

- 1) 총알을 세팅하기 위한 변수이다. public으로 만들어 scene에서 세팅할 수 있도록 할 것이다.
- 2) 마우스를 다운 할 때 안에 있는 fire 함수가 실행될 수 있도록 한다.
- 3) m_bullet을 이용하여 새로운 object를 생성해서 bullet 변수에 저장한다. Instantiate 쓸 때 어떤 변수에 넣고 싶다면 Casting해 주어야 한다. 위 코드에서는 GameObject로 캐스팅 하였다.
- 4) 생성된 Object의 위치를 Hero의 위치로 만들어 준다.



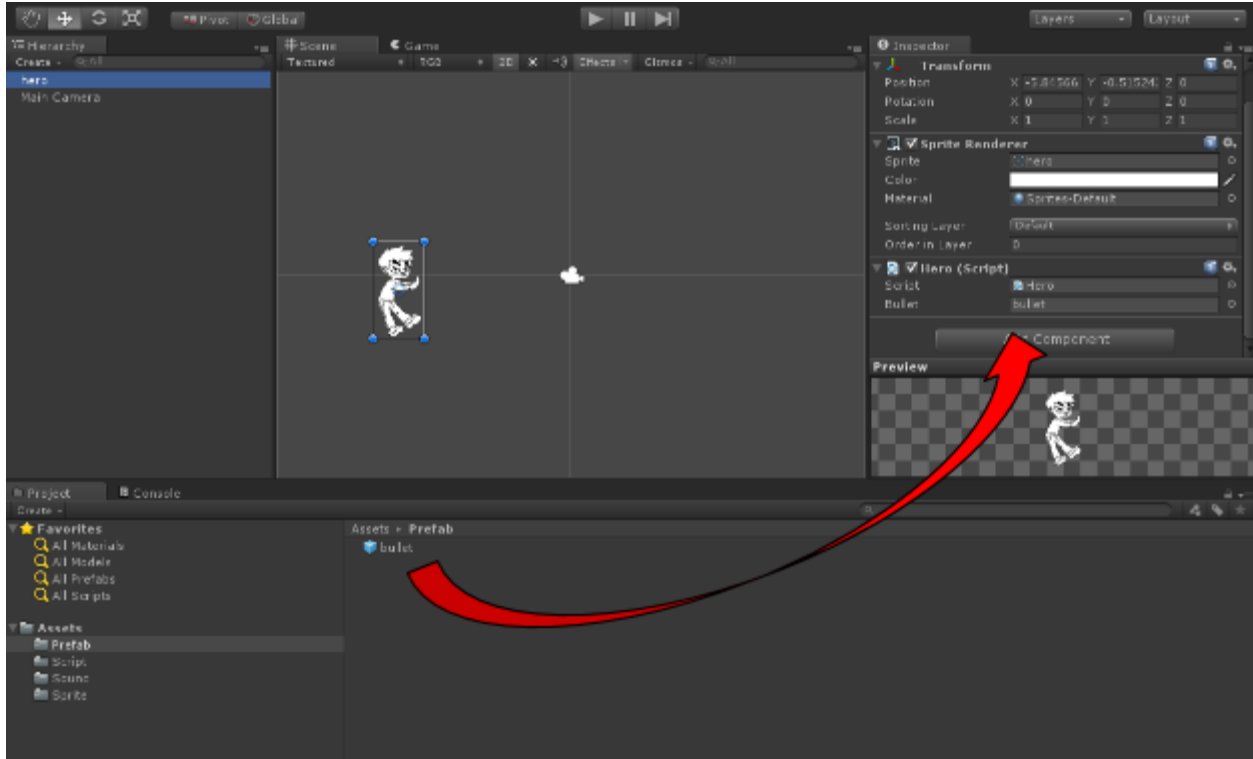
```

! ArgumentException: The thing you want to instantiate is null.
UnityEngine.Object.CheckNullArgument (System.Object arg, System.String message) (at C:/BuildAgent/work/d3d49558e4d408f4/artifacts/EditorGenerated/UnityEngineObject.cs:104)

ArgumentException: The thing you want to instantiate is null.
UnityEngine.Object.CheckNullArgument (System.Object arg, System.String message) (at C:/BuildAgent/work/d3d49558e4d408f4/artifacts/EditorGenerated/UnityEngineObject.cs:104)
UnityEngine.Object.Instantiate (UnityEngine.Object original) (at C:/BuildAgent/work/d3d49558e4d408f4/artifacts/EditorGenerated/UnityEngineObject.cs:90)
Hero.fire () (at Assets/Script/Hero.cs:27)
Hero.Update () (at Assets/Script/Hero.cs:19)

```

코드를 다 넣은 후 실행해 보면 위와 같은 에러가 나는 것을 볼 수 있다. 이유는 public으로 생성했던 m_bullet에 아무값도 세팅되어 있지 않기 때문이다.



동영상

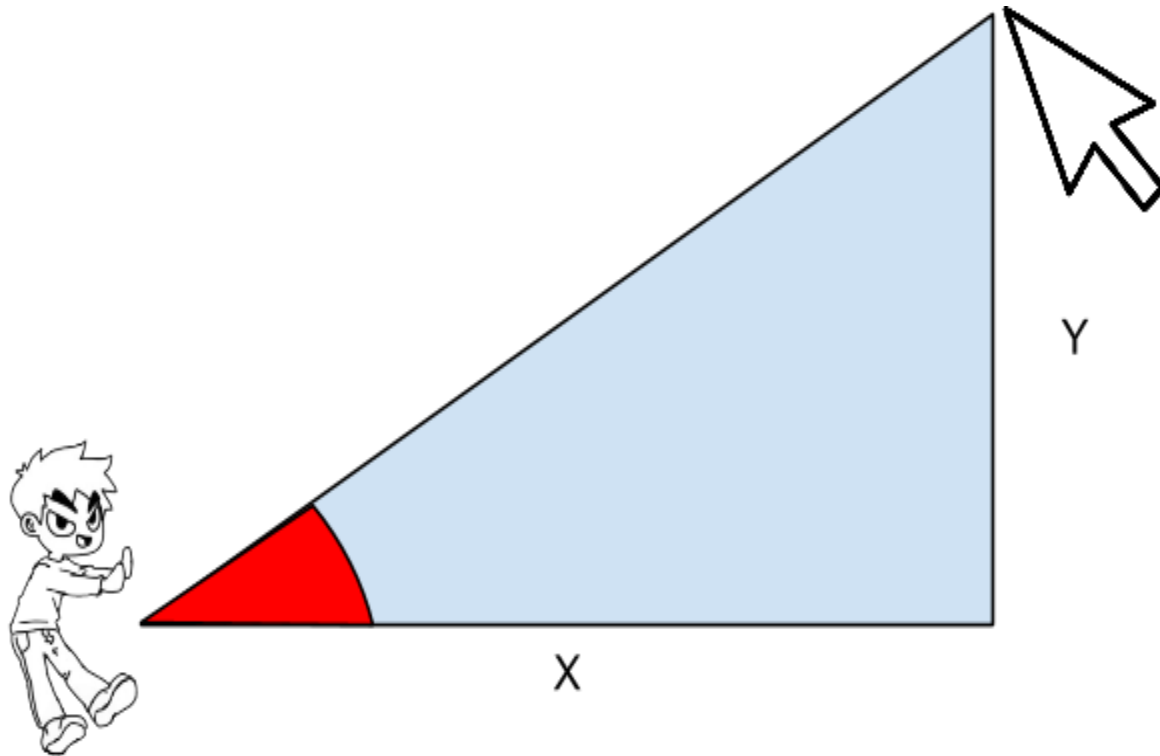
http://www.youtube.com/watch?v=2lnu05tEd18&index=4&list=PLFXLeC2Hh_3db7XFynySEMJkkQ6umELpC

참고

<http://docs.unity3d.com/ScriptReference/Object.Instantiate.html>

총알의 각도 조절

총알이 나가는 것은 하지만 오른쪽으로만 똑바로 나가기 때문에 적이 위나 아래에서 온다면 맞출 수가 없을 것이다. 이번에는 총알을 마우스가 누르는 방향으로 나가도록 해보도록 하자. 이미 총알은 이미 오른쪽으로 나가고 있기 때문에 총알을 마우스의 좌표방향으로 틀어주기만 한다면 내가 원하는 방향으로 쏠 수 있게 될 것이다. 삼각함수를 이용하면 이것을 쉽게 알아낼 수 있다.



총알을 발사하기 위해서는 위 그림에 있는 빨간 부분의 각도를 구해야 한다. 먼저 영웅과 내가 클릭한 포인트 사이의 x길이를 y길이를 구한다. x의 길이는 (마우스 포인트의 x좌표 - 영웅의 x좌표)를 하면 쉽게 구할 수 있다. y길어도 같은 방법으로 구할 수 있다. x, y 길이를 안 상태에서 각도를 구하면 Atan2라는 함수를 이용하면 쉽게 알아낼 수 있다.

이것을 코드에 적용해 보면 아래와 같이 된다.

```
using UnityEngine;
using System.Collections;

public class Hero : MonoBehaviour
{
    public GameObject m_bullet;
    void Start ()
    {
    }

    void Update ()
    {
        if (Input.GetMouseButtonDown(0))
        {
            fire();
        }
    }
}
```

```

private void fire()
{
    // 1
    Vector3 mouse = Camera.main.ScreenToWorldPoint (Input.mousePosition);
    GameObject bullet = Instantiate (m_bullet) as GameObject;
    bullet.transform.position = transform.position;
    // 2
    float rad = Mathf.Atan2 (mouse.y - transform.position.y, mouse.x -
transform.position.x);
    // 3
    bullet.transform.Rotate (new Vector3(0,0, Mathf.Rad2Deg * rad));
}
}

```

- 1) ScreeToWorldPoint함수를 이용해서 마우스의 좌표를 가져온다.
- 2) 방금 설명했던 방법으로 영웅과 마우스의 각도를 구한다.
- 3) 그 각도만큼 생성된 총알의 각도를 돌린다. Atan2로 나오는 값은 Radian이므로 Rad2Deg함수를 사용하여 Degree로 변환하여 사용한다.

동영상

http://www.youtube.com/watch?v=Jk_6s45ee34&list=PLFXLeC2Hh_3db7XFynySEMJkkQ6umELpC&index=5

참고

<http://docs.unity3d.com/ScriptReference/Mathf.Atan2.html>
<http://docs.unity3d.com/ScriptReference/Camera.ScreenToWorldPoint.html>
<http://docs.unity3d.com/ScriptReference/Mathf.Rad2Deg.html>
<http://docs.unity3d.com/ScriptReference/Transform.Rotate.html>
<http://en.wikipedia.org/wiki/Radian>

◆ 적 생성

적은 총알과 동작 방식이 유사하다. 총알과 마찬가지로 그림을 Drag&Drop을 이용해 적당한 곳에 배치하고 Enemy이름으로 Script를 생성한 후 적용해준다.

적의 이동

총알은 오른쪽으로 이동을 했지만 적은 왼쪽으로 움직이도록 할 것이다. 매 프레임마다 왼쪽으로 움직일 수 있도록 x좌표를 빼준다.

```

public class Enemy : MonoBehaviour
{
    void Start ()
    {

```

```

    }

    void Update ()
    {
        // 1
        transform.Translate (new Vector2 (-0.1f, 0));
    }
}

```

1) Enemy 가 왼쪽으로 -0.1f만큼씩 움직이도록 해준다.

다이나믹한 적의 이동

적이 나올때 마다 그냥 똑같은 속도로 나온다면 게임의 재미가 떨어질 것이다. 조금 박진감 넘치는 게임을 만들기 위해서 생성될 때마다 속도를 다르게 해보자.

```

using UnityEngine;
using System.Collections;

public class Enemy : MonoBehaviour
{
    // 1
    public float m_minSpeed = -0.01f;
    // 2
    public float m_maxSpeed = -0.3f;
    // 3
    private float m_speed = 0.0f;

    void Start ()
    {
        // 4
        m_speed = Random.Range (m_minSpeed, m_maxSpeed);
        // 5
        Debug.Log (m_speed);
    }

    void Update ()
    {
        // 6
        transform.Translate (new Vector2 (m_speed, 0));
    }
}

```

- 1) 적이 생성될 때 설정되는 최소 속도. public으로 scene에서도 설정할 수 있도록 한다.
- 2) 적이 생성될 때 설정되는 최대 속도. public으로 scene에서도 설정할 수 있도록 한다.
- 3) 적이 움직일 스피드를 저장할 변수.
- 4) Range함수를 이용하여 두 변수의 중간 값을 m_speed에 저장한다.
- 5) 실행했을 때는 속도가 변하는지 알 수 없으니 Log메시지를 찍어서 확인한다.
- 6) 랜덤으로 선택된 속도 만큼 이동시킨다.

아래 동영상을 보면 실행할때마다 속도가 변하는것을 볼 수 있다. 속도는 실행해 보면서 Scene에서 적당하게 설정하도록 하자.

동영상

http://www.youtube.com/watch?v=kRig2Q71Blc&index=6&list=PLFXLeC2Hh_3db7XFynySEMJkKQ6umELpC

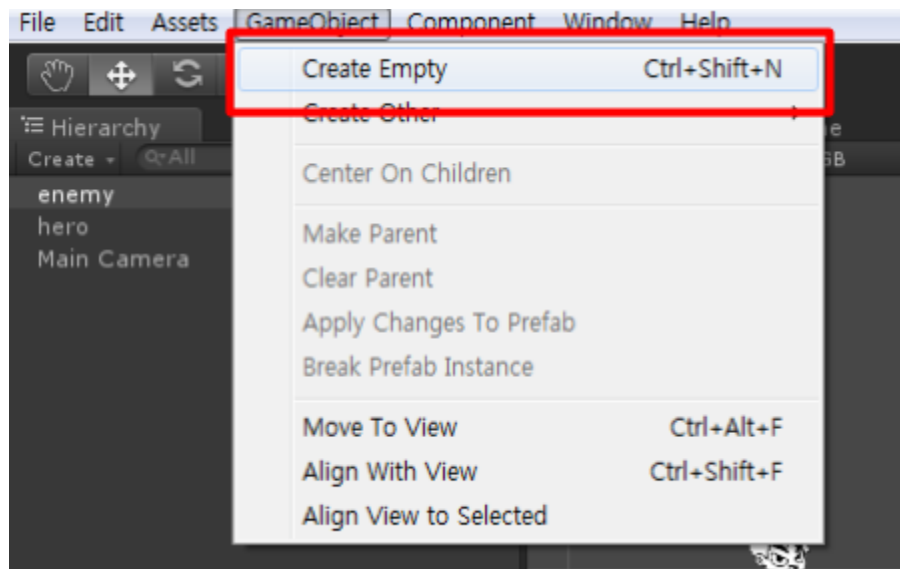
참고

<http://docs.unity3d.com/ScriptReference/Random.Range.html>

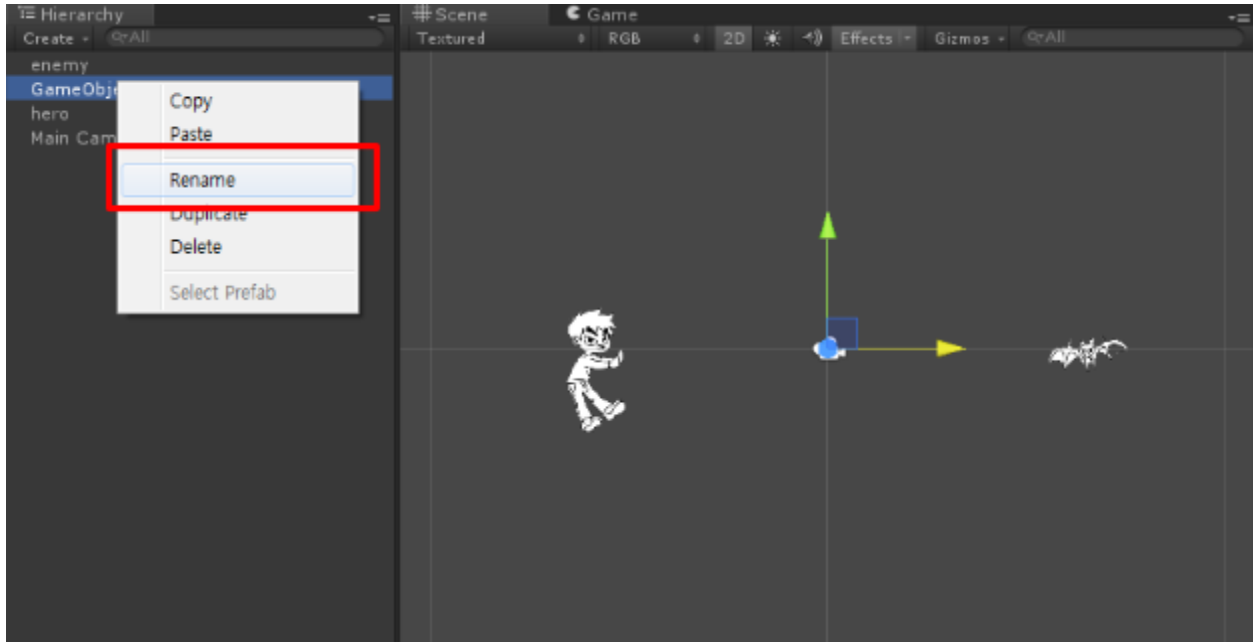
<http://docs.unity3d.com/ScriptReference/Debug.Log.html>

다이나믹한 적의 위치 설정

적당한 스피드를 설정한 후 적을 Prefab로 만든다. 만드는 방법은 아까 총알을 만들 때처럼 Assets 아래에 있는 Prefab 폴더에 Drag&Drop을 하면 된다. 이번에도 총알처럼 여러 개의 적이 생성되도록 할 것이다.



적을 생성하는 스크립트를 만들기 위해 GameObject -> Create Empty를 선택하여 빈 오브젝트를 생성한다.



생성을 하면 Hierarchy에 GameObject 라는 이름으로 새로운 오브젝트가 생성이 되는데 다른 오브젝트들과 혼란이 올 수도 있다. 마우스 오른쪽 클릭을 한 후 Rename을 선택해서 적당한 이름으로 고쳐주도록 하자.

그리고 이 Object에 EnemySpawner라는 Script를 생성한 후에 적용해준다.

```
using UnityEngine;
using System.Collections;

public class EnemySpawner : MonoBehaviour
{
    // 1
    public GameObject m_enemy;

    void Start ()
    {
        // 2
        StartCoroutine ("make");
    }

    void Update ()
    {
    }

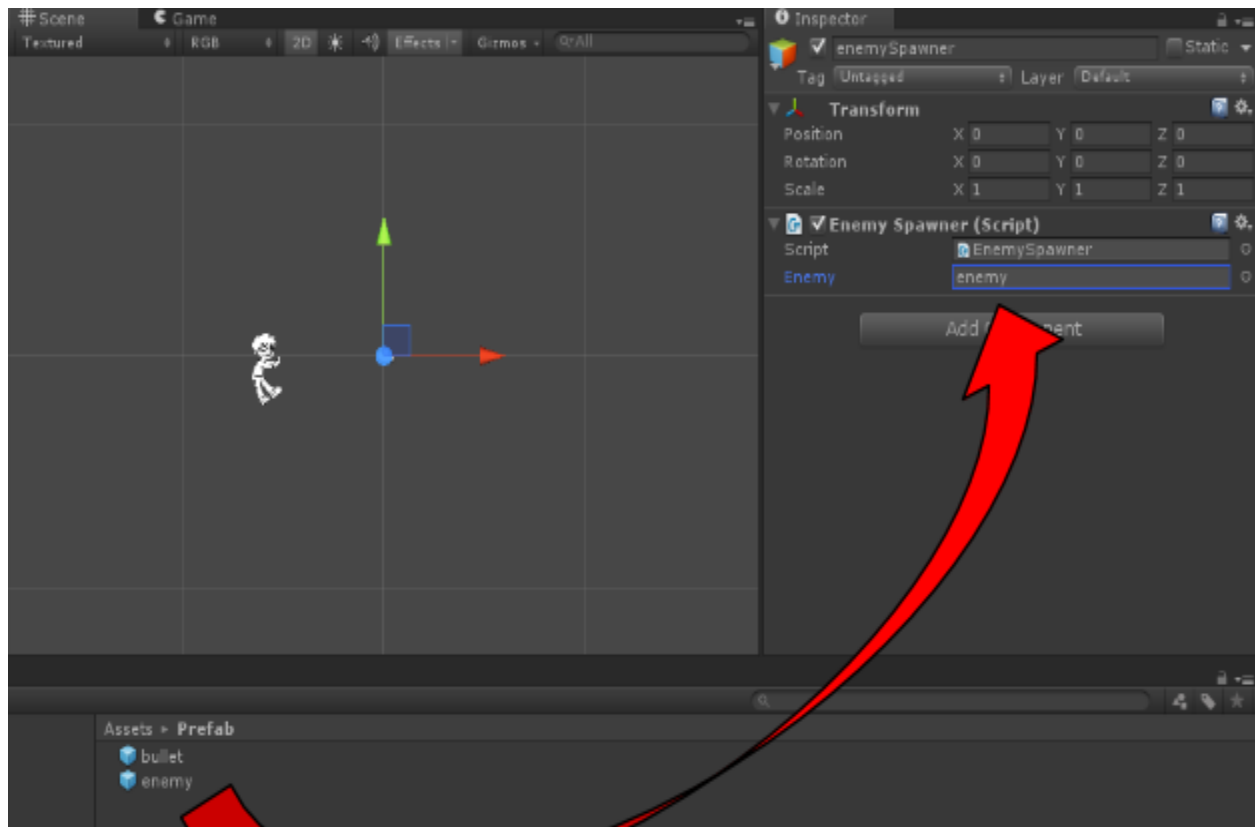
    // 3
    private IEnumerator make()
    {
        // 4
        while (true)
        {
            // 5
            yield return new WaitForSeconds (1.0f);
        }
    }
}
```

```

        // 6
        GameObject enemy = Instantiate (m_enemy) as GameObject;
        // 7
        enemy.transform.position = new Vector2 ( transform.position.x,
transform.position.y );
    }
}

```

- 1) Scene에서 적을 설정할 수 있도록 변수를 선언해준다.
- 2) 적은 생성하기 위한 Coroutine을 실행한다.
- 3) 적은 생성하기 위한 Coroutine 함수
- 4) 적을 무한으로 생성하기 위해 while을 이용하여 무한으로 실행 될 수 있도록 해준다.
- 5) Coroutine을 1초 동안 잠시 멈춘다.
- 6) m_enemy를 이용하여 새로운 Object 를 생성한다.
- 7) 생성된 Obejct 의 좌표를 설정해준다. 일단 임시로 현재 Object의 좌표로 설정해준다.



마지막으로 아까 만들어 놓은 enemy Prefab를 Drag & Drop 해준다.

실행을 해보면 적이 1초마다 생성되는 것을 볼 수 있다. 하지만 한 부분에서만 나오기 때문에 게임이 단조로워 보인다.

동영상

http://www.youtube.com/watch?v=rpLZMfbeY40&index=7&list=PLFXLeC2Hh_3db7XFynySEMJkKQ6umELpC

참고

<http://docs.unity3d.com/Manual/Coroutines.html>

<http://docs.unity3d.com/ScriptReference/MonoBehaviour.StartCoroutine.html>

<http://docs.unity3d.com/ScriptReference/WaitForSeconds.html>

적의 위치 선정

이번에는 적이 정해진 위치 사이에서 랜덤으로 생성되도록 할 것이다. 억지로 숫자를 집어넣어 설정해줘도 되지만 눈으로 보면서 쉽게 세팅할 수 있도록 transform을 생성하여 값을 세팅하는 방법을 이용할 것이다. 이 방법을 사용하면 나중에도 위치를 쉽게 수정할 수 있다.

```
using UnityEngine;
using System.Collections;

public class EnemySpawner : MonoBehaviour
{
    // 1
    public Transform m_top;
    public Transform m_bot;

    public GameObject m_enemy;

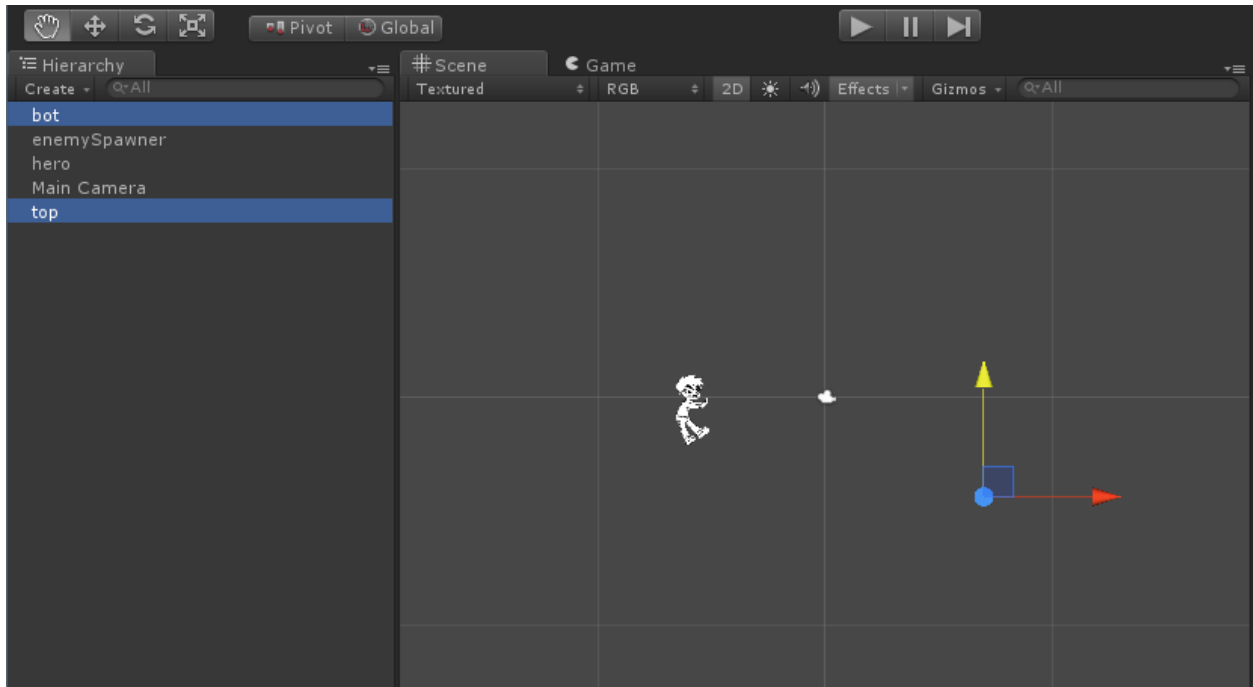
    void Start ()
    {
        StartCoroutine ("make");
    }

    void Update ()
    {
    }

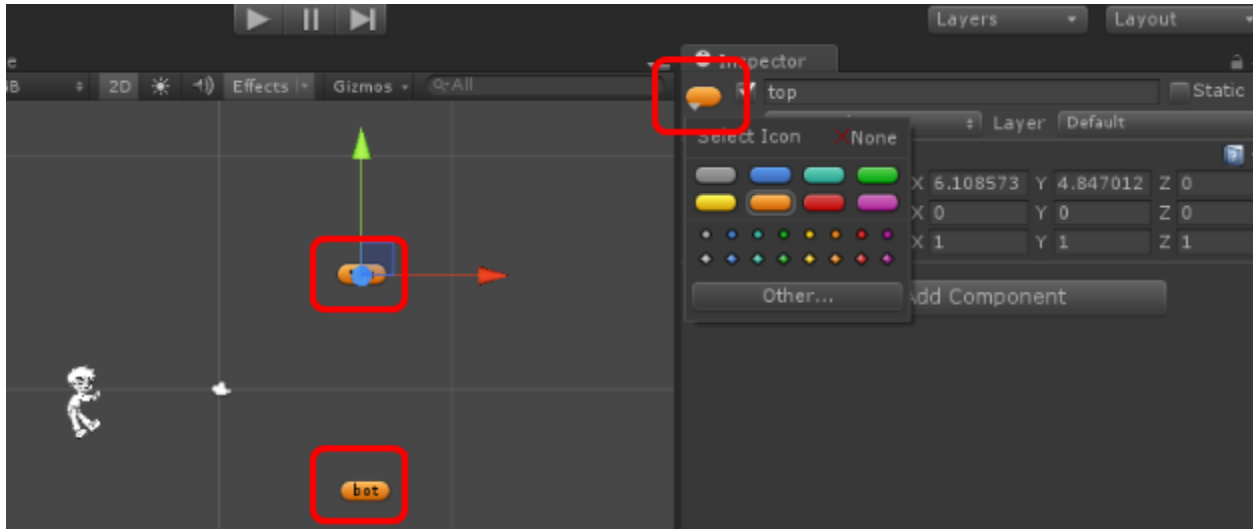
    private IEnumerator make()
    {
        while (true)
        {
            yield return new WaitForSeconds (1.0f);
            GameObject enemy = Instantiate (m_enemy) as GameObject;
            // 2
            enemy.transform.position = new Vector2 (
m_top.transform.position.x , Random.Range(m_top.transform.position.y,
m_bot.transform.position.y) );
        }
    }
}
```

```
}
```

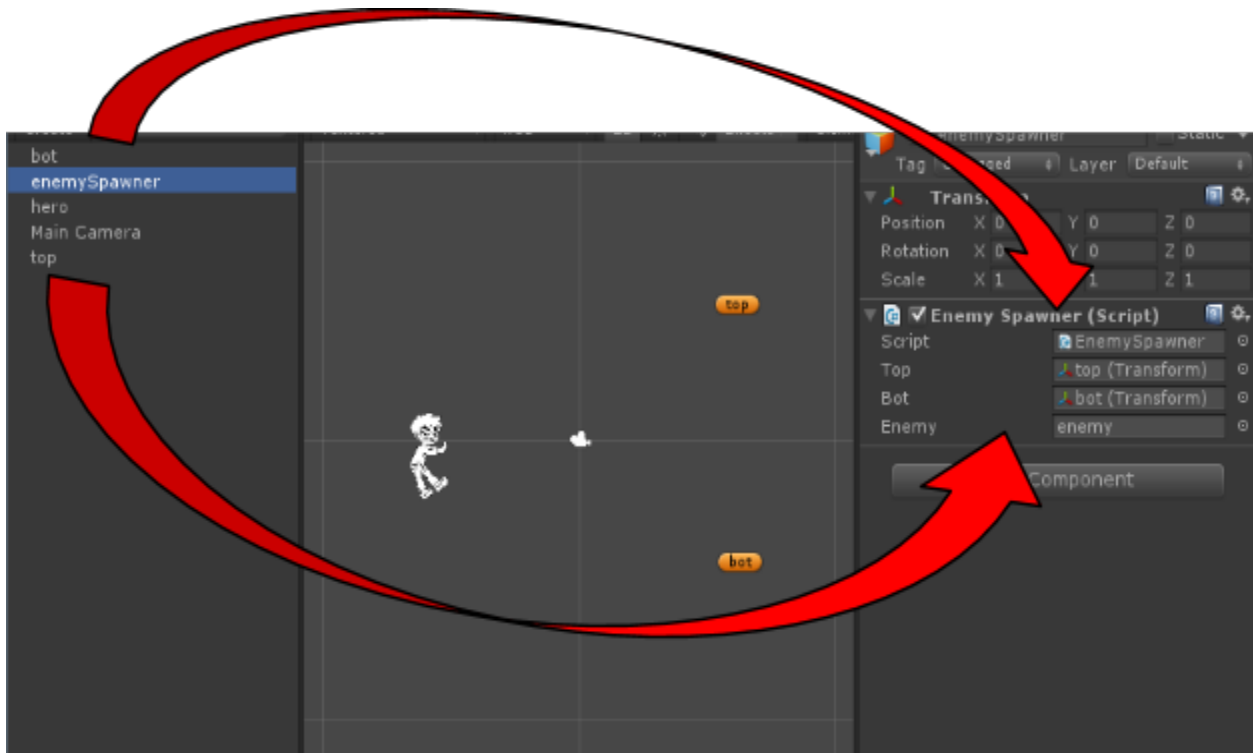
- 1) Scene에서 설정을 할 수 있도록 public 으로 transform을 두개 만든다. 이 transform 중간에 적이 생기도록 할 것이다.
- 2) 두개 중간에 있는 y좌표를 랜덤으로 설정한다. x 좌표는 첫번째 있는 좌표로 설정하였다.



이번에는 기준이 될 transform을 생성할 것이다. GameObject -> Create Empty 를 이용하여 생성 가능하다.
두 개를 생성 한 후에 각각 top, bot라고 이름을 지어준 후 적당한 위치에 배치 시켜준다. 생성 된 두 가지 object는 눈으로 보이지 않기 때문에 불편할 수도 있다.



Inspector에서 이 기능을 이용하면 Scene 화면에서 보이기 때문에 쉽게 위치 수정이 가능하다. 이 표시는 실제 게임에서는 표시되지 않는다.



(드래그 드랍 하는 그림)

마지막으로 top과 bot를 Script에 Drag & Drop 해준다. 실행해보면 설정한 위치 사이에서 적이 생성 되는 것을 볼 수 있다.

동영상

http://www.youtube.com/watch?v=0foKCtrlLE&list=PLFXLeC2Hh_3db7XFynySEMJkkQ6umELpC&index=8

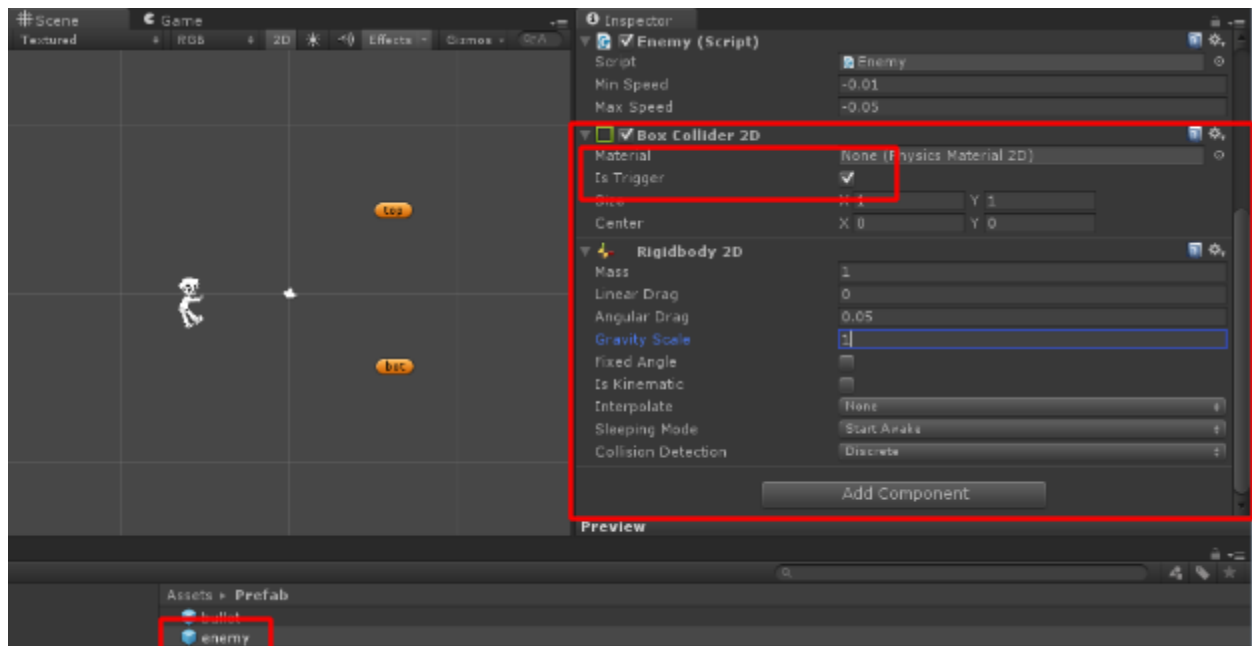
참고

<http://docs.unity3d.com/Manual/Transforms.html>

◆ 적 제거

적 수정

총알을 아무리 쏘아도 적이 제거 되지 않는데 이번에는 적을 제거해 보도록 하겠다.



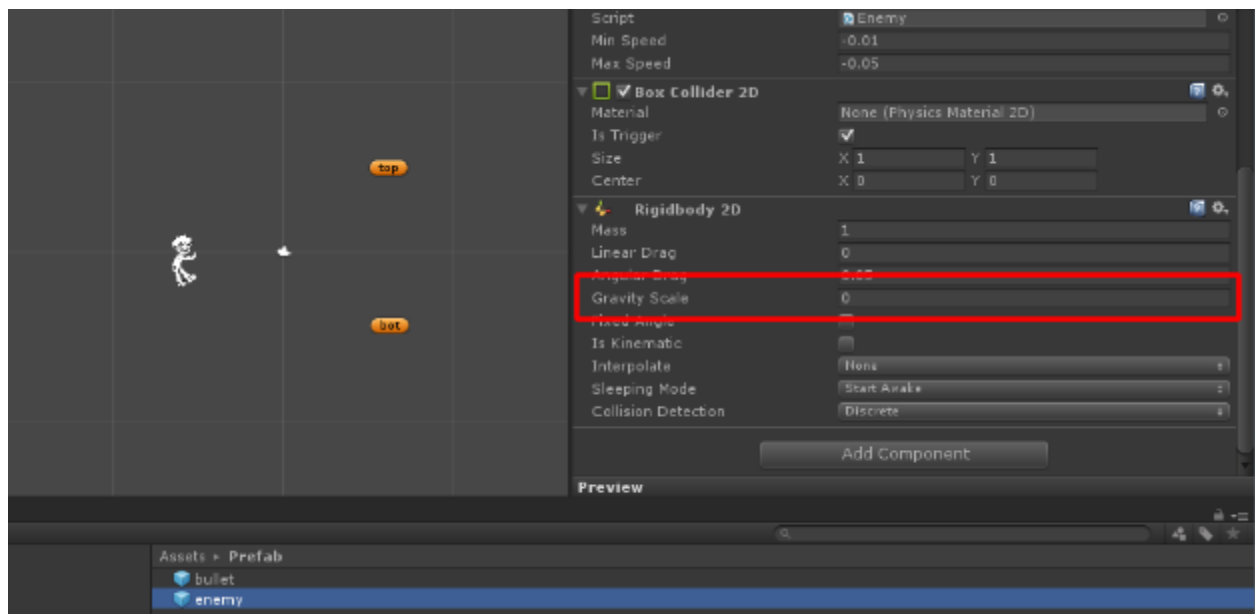
먼저 충돌을 감지하기 위해서는 Collider와 rigidbody를 추가해줘야 한다. 현재 하고 있는 프로젝트는 2d이기 때문에 그림과 같이 2d로 추가해준다. 그리고 Is Trigger를 체크해준다. 이것을 체크해주면 rigidbody 간에 충돌 물리 현상이 일어나지 않고 단순히 닿았는지 닿지 않았는지만 체크할 수 있다.

실행을 해보면 적이 바닥으로 떨어지는 것을 볼수 있다.

동영상

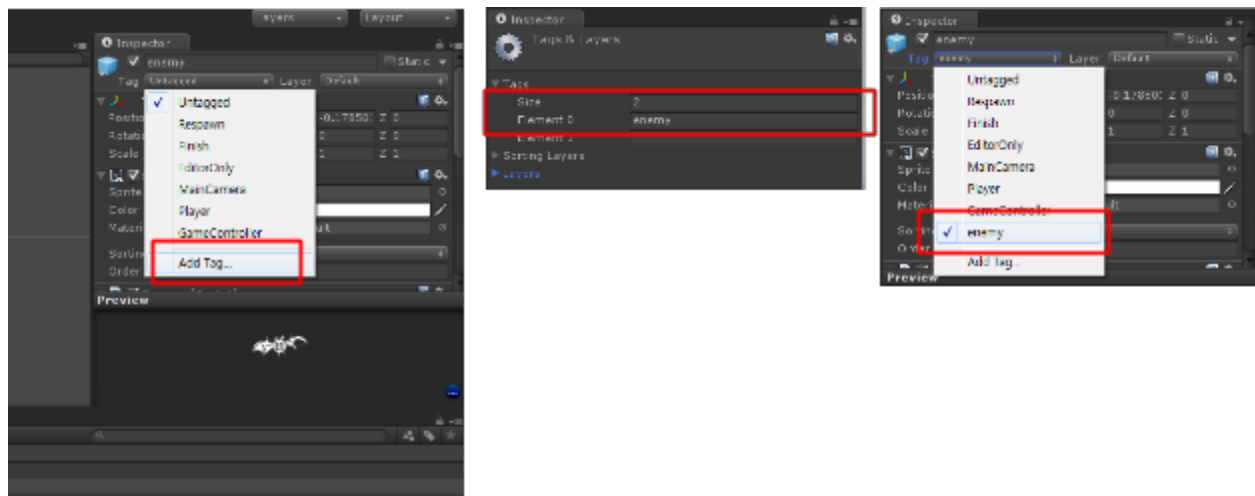
http://www.youtube.com/watch?v=B_xu1nGMARg&list=PLFXLeC2Hh_3db7XFynySEMJkkQ6umELpC&index=9

rigidbody가 추가되면서 물리 객체가 되었기 때문에 중력의 영향을 받아 바닥으로 떨어지고 있는 것이다.



우리가 원하는 것은 왼쪽으로 적이 계속 움직이는 것이기 때문에 중력의 영향을 받지 않도록 gravity 값을 0으로 세팅해준다.

어떤 물체가 다른 물체와 충돌 했을 때, 무엇과 충돌했는지 알아낼 수 있는 방법 중 하나는 tag를 추가하는 것이다.



Inspector에서 Add tag를 통해서 enemy tag 를 추가한 후에 enemy를 선택하도록 하자.

참고

<http://docs.unity3d.com/ScriptReference/Rigidbody2D.html>

<http://docs.unity3d.com/Manual/class-Rigidbody2D.html>

<http://docs.unity3d.com/ScriptReference/Collider.html>

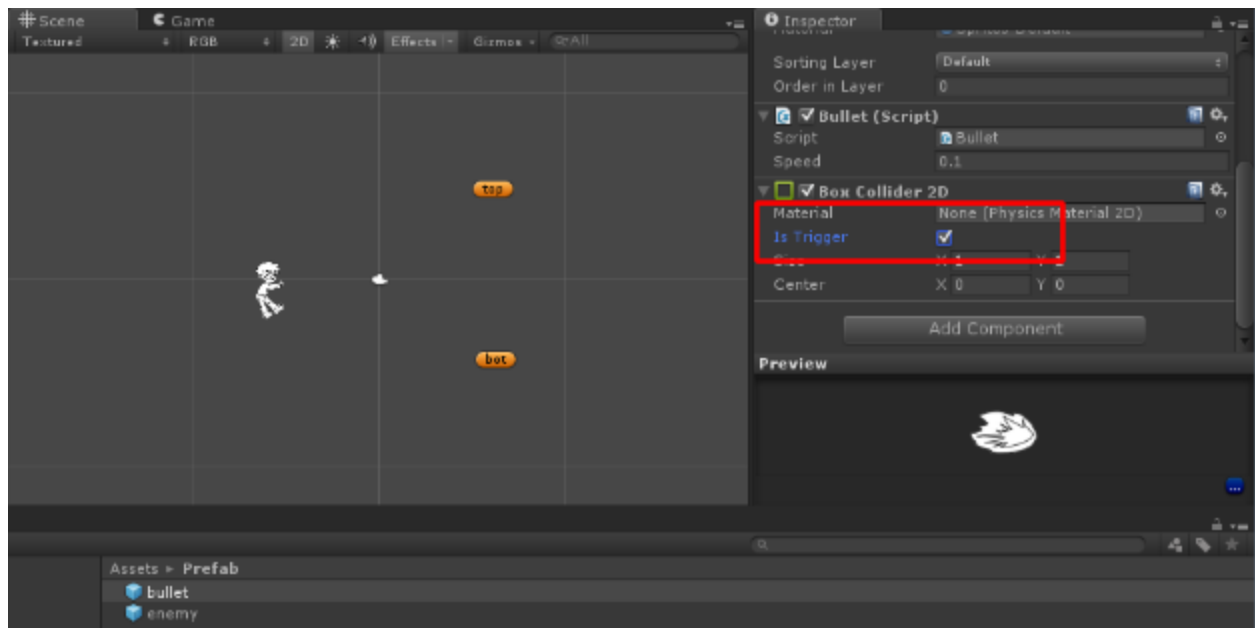
<http://unity3d.com/learn/tutorials/modules/beginner/physics/colliders>

<http://docs.unity3d.com/Manual/Tags.html>

<http://docs.unity3d.com/ScriptReference/GameObject-tag.html>

총알 수정

총알 Prefab에는 Box Collider 2D 만 추가를 한다. 둘 다 물리 객체처럼 움직이게 하고 싶다면 rigidbody를 추가 해줘야 하지만 우리는 단순히 충돌 여부만 알면 되기 때문에 추가할 필요가 없다.



추가한 후에 Is Trigger를 체크해준다. 이렇게 해주면 OnTriggerEnter2D라는 함수에서 충돌을 체크 할 수 있다.

```
using UnityEngine;
using System.Collections;

public class Bullet : MonoBehaviour
{
    public float m_speed = 0.3f;

    void Start ()
    {
        Destroy (gameObject, 3.0f);
    }

    void Update ()
    {
        transform.position += transform.right * m_speed;
    }
    // 1
```

```

void OnTriggerEnter2D(Collider2D other)
{
    // 2
    if (other.gameObject.tag == "enemy")
    {
        // 3
        Destroy(other.gameObject);
        Destroy(gameObject);
    }
}

```

- 1) Is Trigger를 체크한 후 두 개의 Collider가 충돌하게 되면 OnTriggerEnter2D가 실행된다. other는 충돌한 오브젝트를 의미한다.
- 2) 충돌한 오브젝트의 tag가 아까 설정한 enemy인지 체크한다.
- 3) 자신(총알)과 enemy 를 모두 제거한다.

동영상

http://www.youtube.com/watch?v=9_iKxQN97P0&list=PLFXLeC2Hh_3db7XFynySEMJkkQ6umELpC&index=10

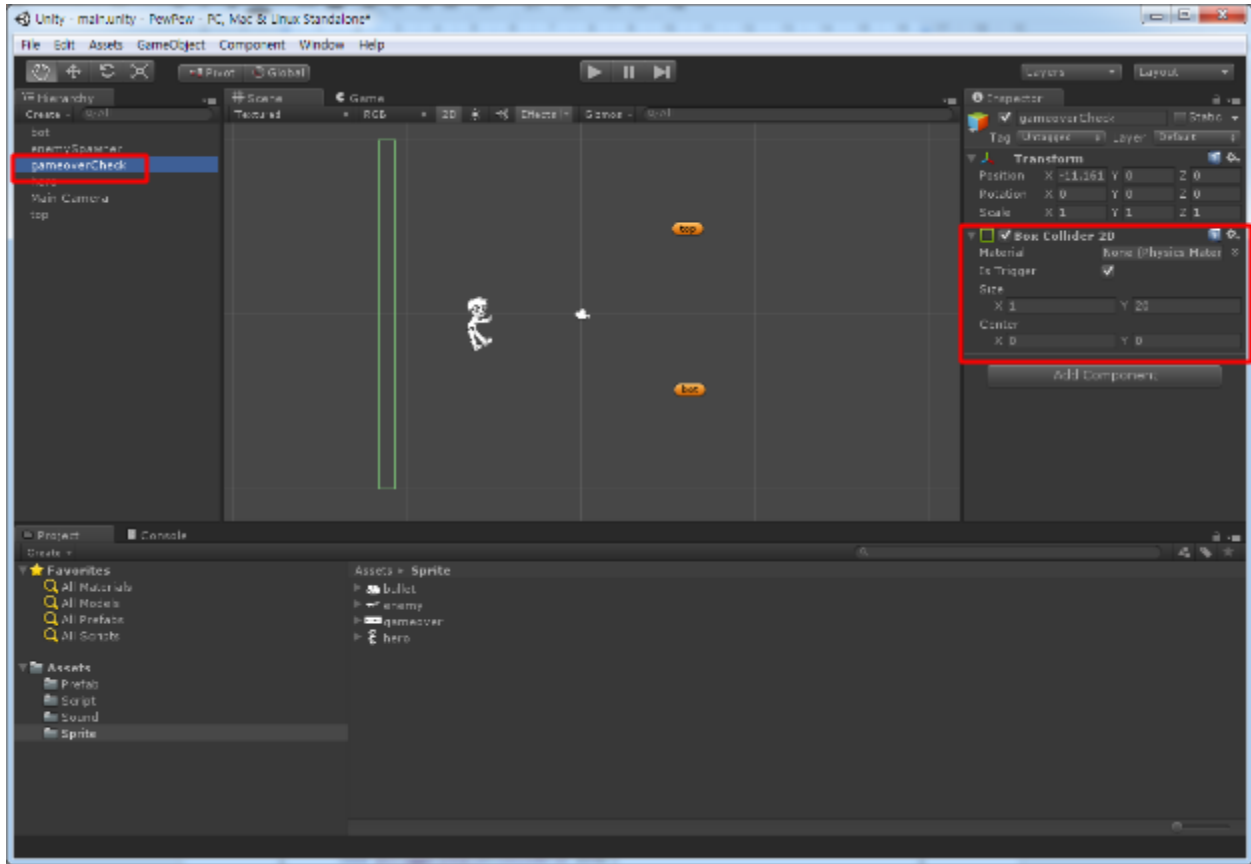
참고

<http://docs.unity3d.com/ScriptReference/MonoBehaviour.OnTriggerEnter2D.html>
<http://unity3d.com/learn/tutorials/modules/beginner/physics/colliders-as-triggers>

◆ 게임오버

적이 화면 밖으로 나갈때 제거

이제는 총알을 쏘서 적을 없앨 수 있게 만들었지만, 적을 제거하지 않아 게임오버 당하는 경우가 없다. 이번에는 적이 화면 끝으로 이동하면 게임이 끝나도록 만들 것이다.



GameObject -> Create Empty를 통해 GameObject를 생성한 후 적당한 이름을 지어주고 Box Collider 2D를 추가해준다. Is Trigger를 체크해주고 적의 체크 될 수 있을 만큼 크기를 올려준다. GameOverChecker라는 c#코드를 추가하고 아래 코드를 삽입하도록 하자.

```
using UnityEngine;
using System.Collections;

public class GameOverChecker : MonoBehaviour
{
    void Start ()
    {

    }

    void Update ()
    {

    }

    // 1
    void OnTriggerEnter2D(Collider2D other)
    {
        Destroy (other.gameObject);
    }
}
```

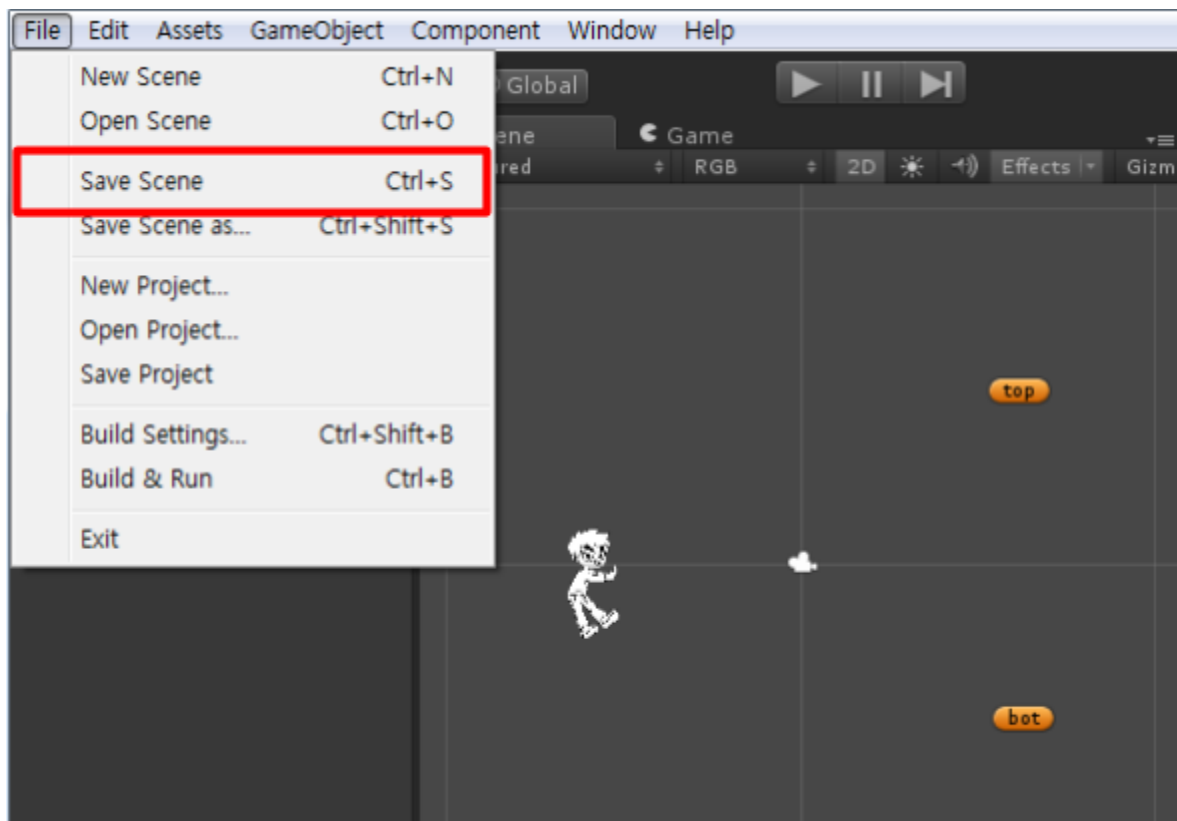
```
}
```

1) 적이 이 화면 끝으로 오게되면 적을 제거한다.

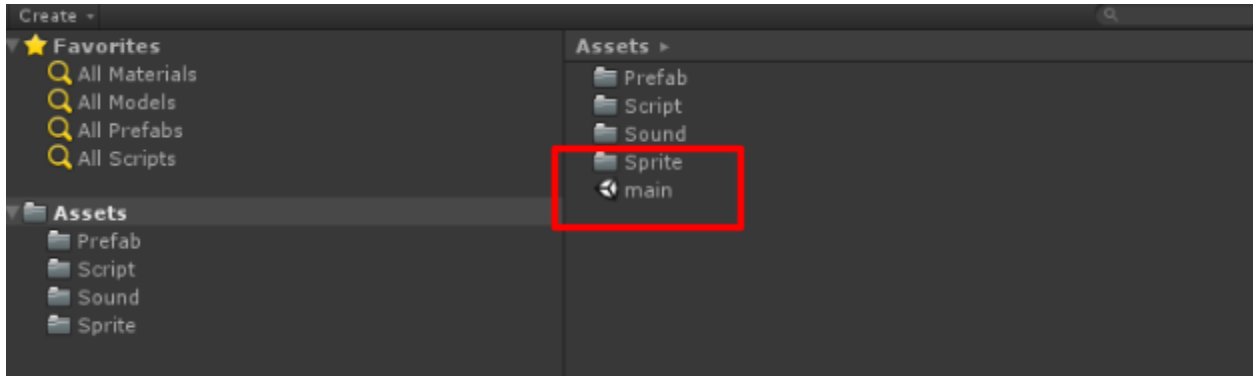
동영상

http://www.youtube.com/watch?v=vSxkjPB8oS8&list=PLFXLeC2Hh_3db7XFynySEMJkkQ6umELpC&index=11

◆ 게임오버



들어가기 전에 현재 작업하고 있는 main이라는 이름으로 Scene을 저장하도록 하자. 이 튜토리얼에서는 막바지에 Scene을 저장하지만 진짜 작업을 할 때는 처음부터 자주 저장하는 버릇을 들이는 것이 좋다.



Scene을 저장하면 Assets에 저장되는 것을 볼 수 있다. 위 그림에서는 Scene을 Assets에 바로 저장했지만 Scene이 많다면 따로 폴더를 만들어서 관리해도 된다.

이번에는 적이 화면 끝으로 오게 되면 제거 되도록 할 것이다. GameOverChecker Script를 아래와 같이 수정한다.

```
using UnityEngine;
using System.Collections;

public class GameOverChecker : MonoBehaviour
{
    // 1
    public GameObject m_gameOver;

    // 2
    private bool m_isOver = false;

    void Start ()
    {

    }

    void Update ()
    {

    }

    void OnTriggerEnter2D(Collider2D other)
    {
        // 3
        if (other.gameObject.tag == "enemy")
        {
            // 4
            if(m_isOver == false)
            {
                // 5
                m_isOver = true;
                // 6
                m_gameOver.SetActive(true);
                // 7
                StartCoroutine("restartGame");
            }
        }
    }
}
```

```

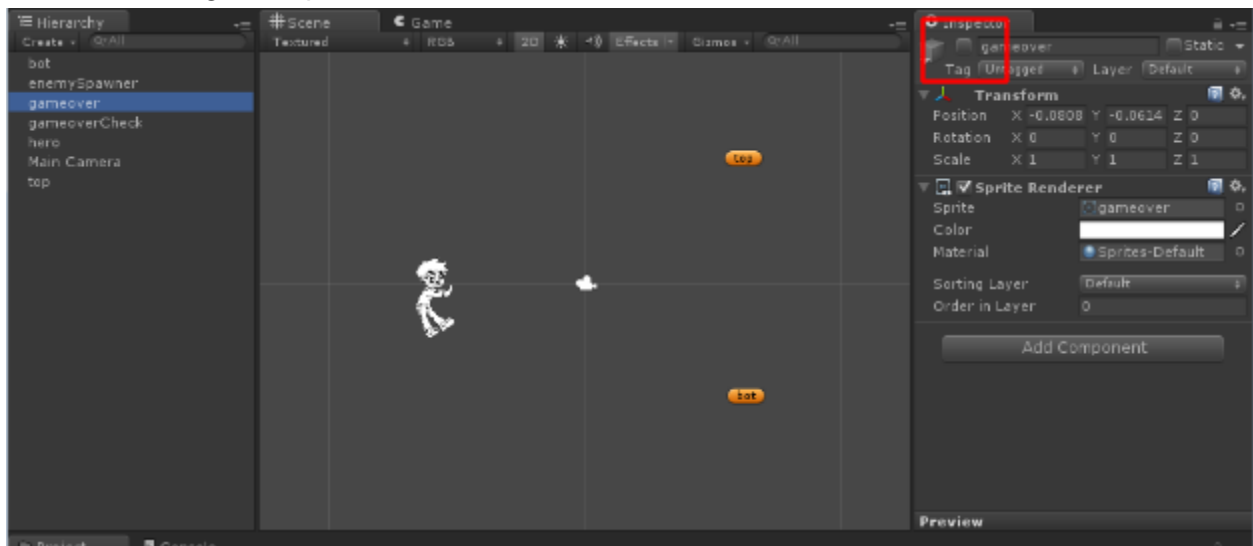
    }
    Destroy (other.gameObject);
}

private IEnumerator restartGame()
{
    // 8
    yield return new WaitForSeconds (3.0f);
    // 9
    Application.LoadLevel("main");
    yield return null;
}
}

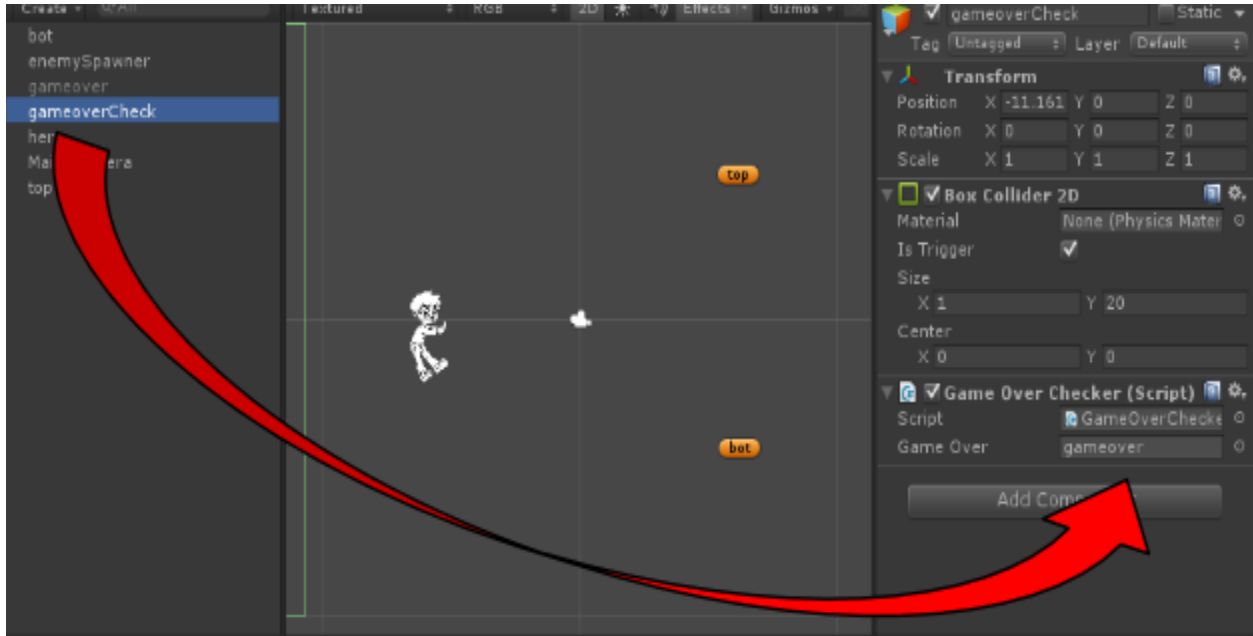
```

- 1) 게임 오버 그림을 띄워주기 위한 GameObject
- 2) 게임 오버가 여러번 체크되지 않게 하기 위한 Flag
- 3) tag 가 enemy 인 오브젝트만 체크한다.
- 4) m_isOver가 false일 경우에만 들어온다.
- 5) m_isOver를 true로 만들어줘서 다시 이 if문으로 들어오지 못하도록 한다.
- 6) setActive 함수를 이용하여 화면에 표시해 준다.
- 7) StarCoroutine 함수를 이용하여 restartGame 함수를 실행시킨다.
- 8) 3초 기다린다.
- 9) LoadLevel함수를 이용하여 Scene을 다시 로드하도록 하자.

다음으로는 m_gameOver를 세팅하기 위한 GameObject를 생성하도록 하자. gameover.png를 Scene으로 Drag & Drop 하여 생성한다.



게임을 시작하면 Gameover 오브젝트가 보이지 않도록 그림에 빨간 네모 부분에 있는 체크를 없애주자. Script에서 setActive 함수를 사용했는데 체크를 없애면 setActive를 false로 세팅할 것과 같다.



그리고 gameover 오브젝트를 Drag & Drop으로 적용해 주면 된다.

동영상

http://www.youtube.com/watch?v=yH5oPJU8zho&list=PLFXLeC2Hh_3db7XFynySEMJkkQ6umELpC&index=12

참고

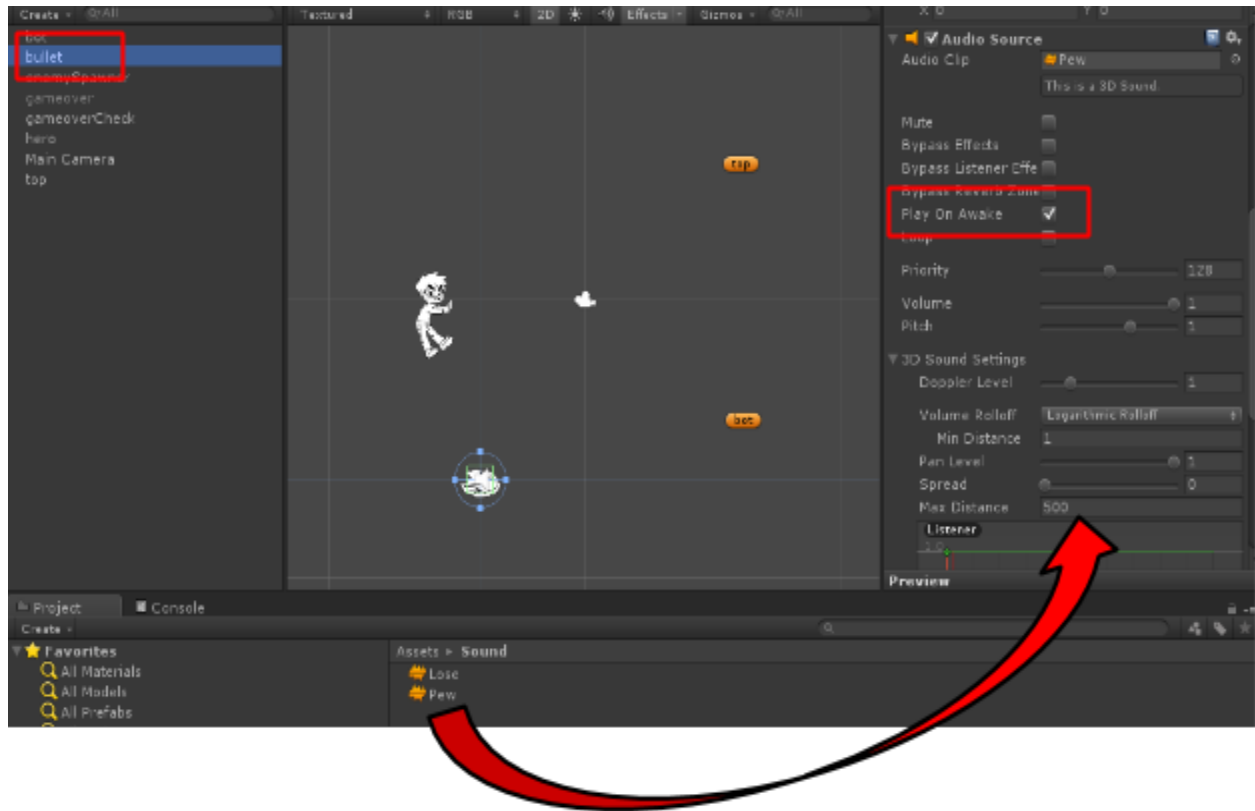
<http://docs.unity3d.com/ScriptReference/GameObject.SetActive.html>

<http://docs.unity3d.com/ScriptReference/Application.LoadLevel.html>

◆ 소리추가

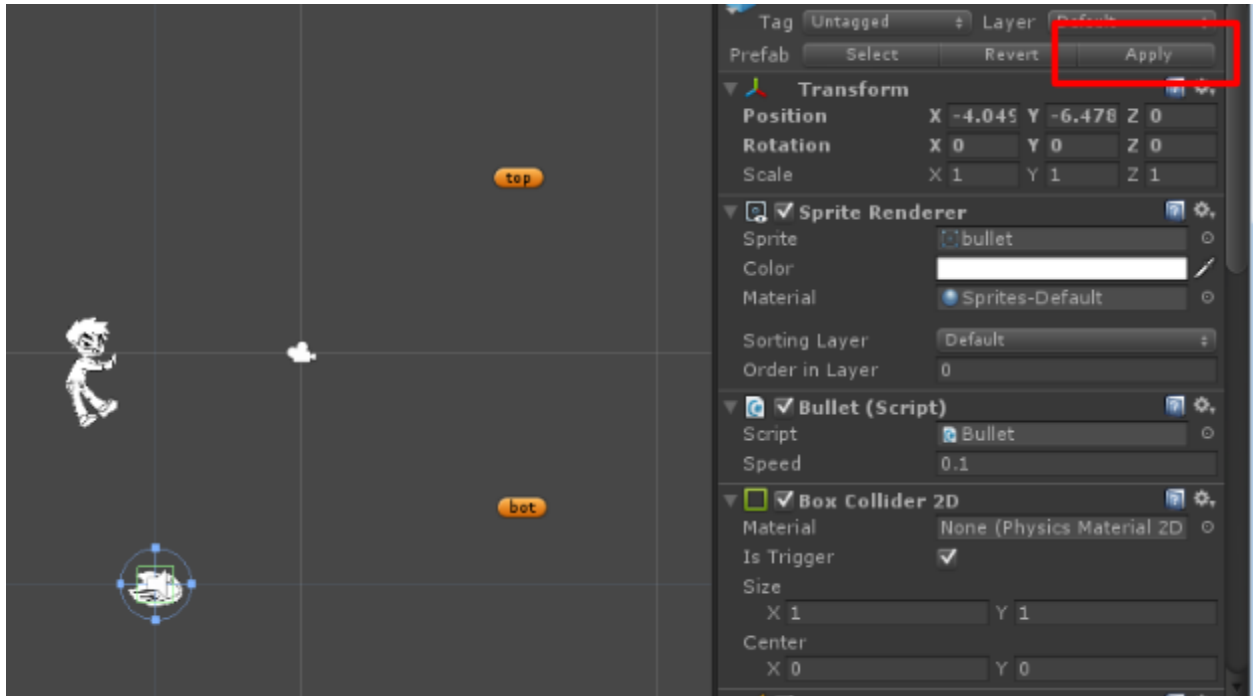
총을 쏠 때 소리

총알을 쏠때 소리가 나게 하려면 총알이 생성될 때 소리가 나도록 하면 될 것이다. Bullet Prefab를 수정하면 된다. 아까와는 다른 방법으로 Prefab를 수정해 볼 것이다. 일단 Bullet Prefab를 Scene에 Drag & Drop 하여 하나 만들도록 하자.



만들어진 Bullet을 선택하고 Pew사운드를 Drag & Drop 해준다. 그렇게 되면 Audio Source가 생긴다. Play On Awake가 체크 되어 있다면 생성 되자마자 소리가 나게 될 것이다.

하지만 플레이를 해보면 화면에 생성해놓은 총알 빼고는 소리가 나지 않는것을 볼 수 있다. 이것은 나와있는 Prefab을 아무리 수정해도 원본에는 아무 영향을 끼치지 못하기 때문이다.



만약에 수정한 Prefab를 원본에도 적용하고 싶다면 apply버튼을 눌러주면 된다. apply 누른 후에는 Scene에 나와있는 총알을 제거해준다.

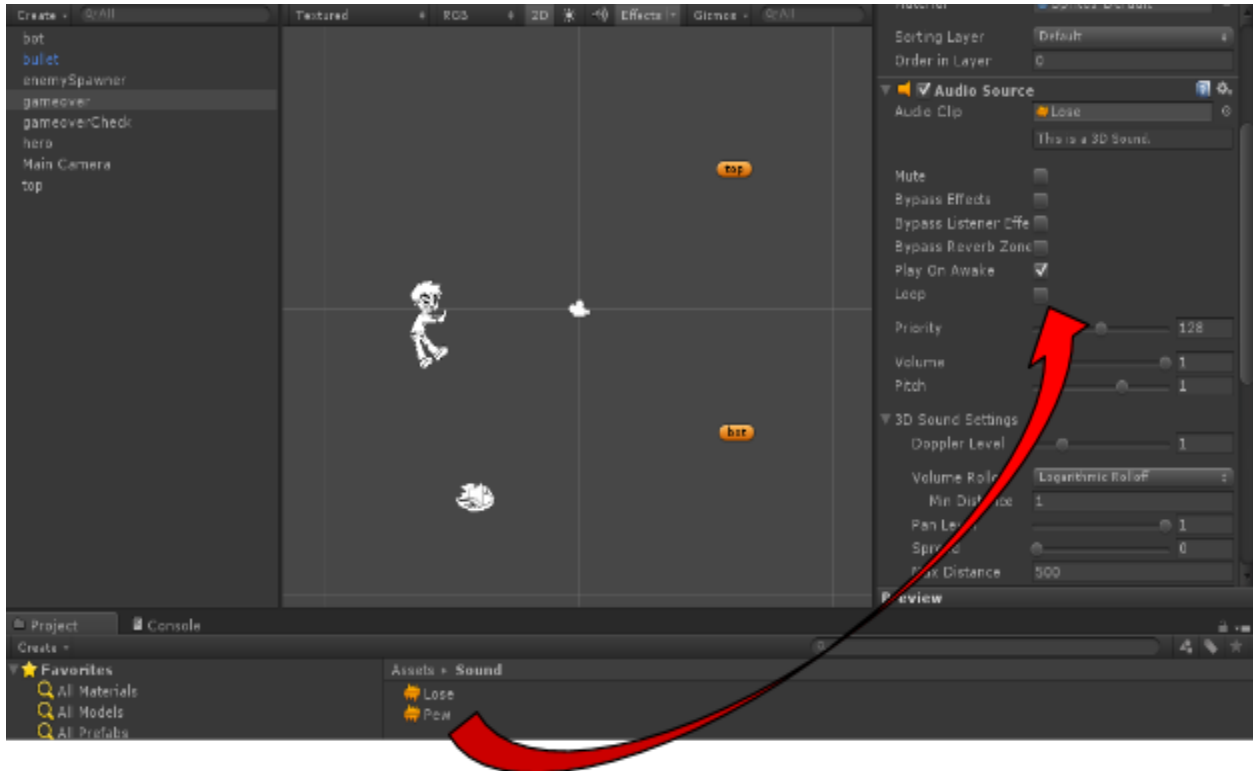
동영상

http://www.youtube.com/watch?v=JQcAjYyTsY&list=PLFXLeC2Hh_3db7XFynySEMJkkQ6umELpC&index=13

참고

<http://docs.unity3d.com/Manual/class-AudioSource.html>

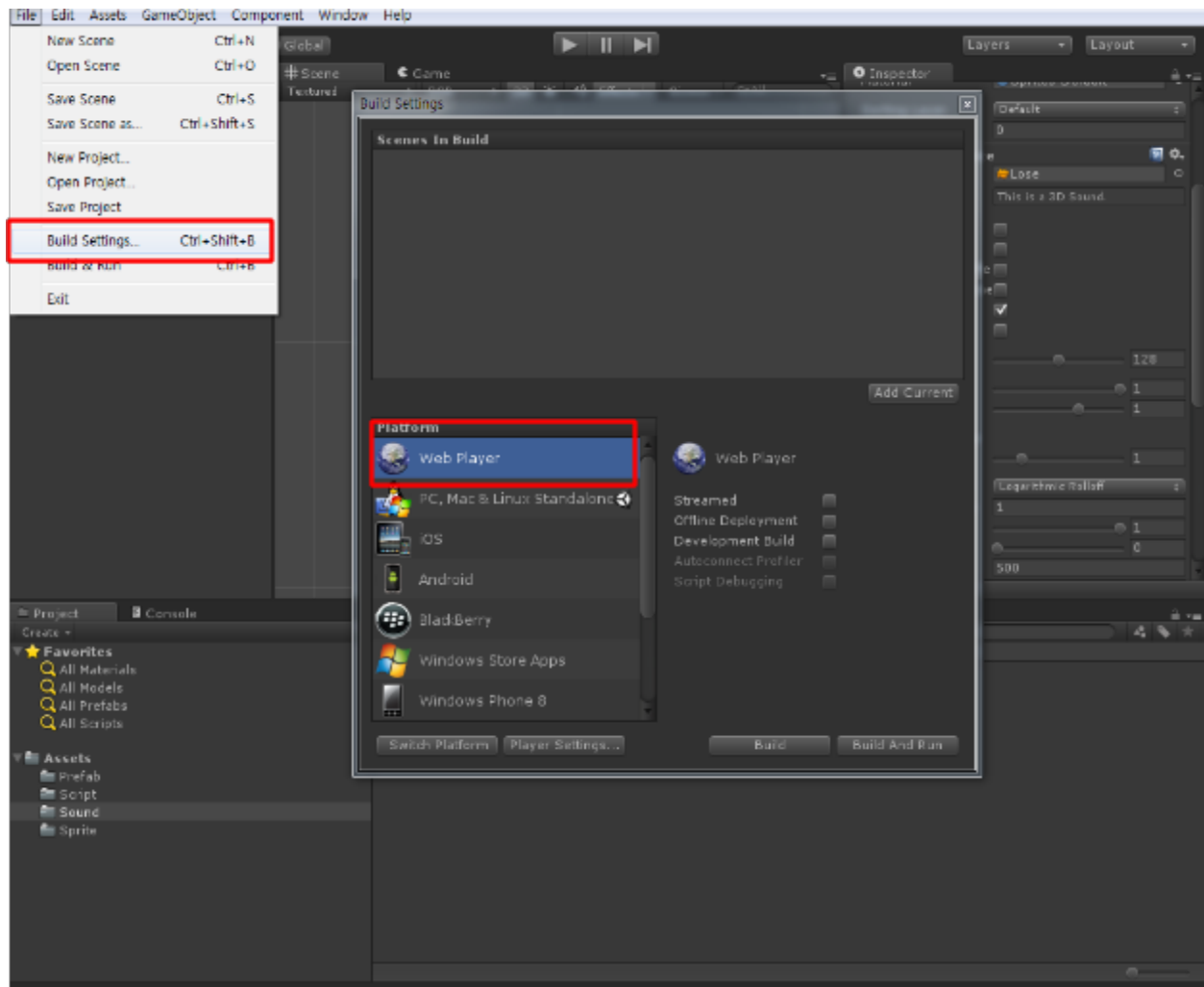
게임오버 소리



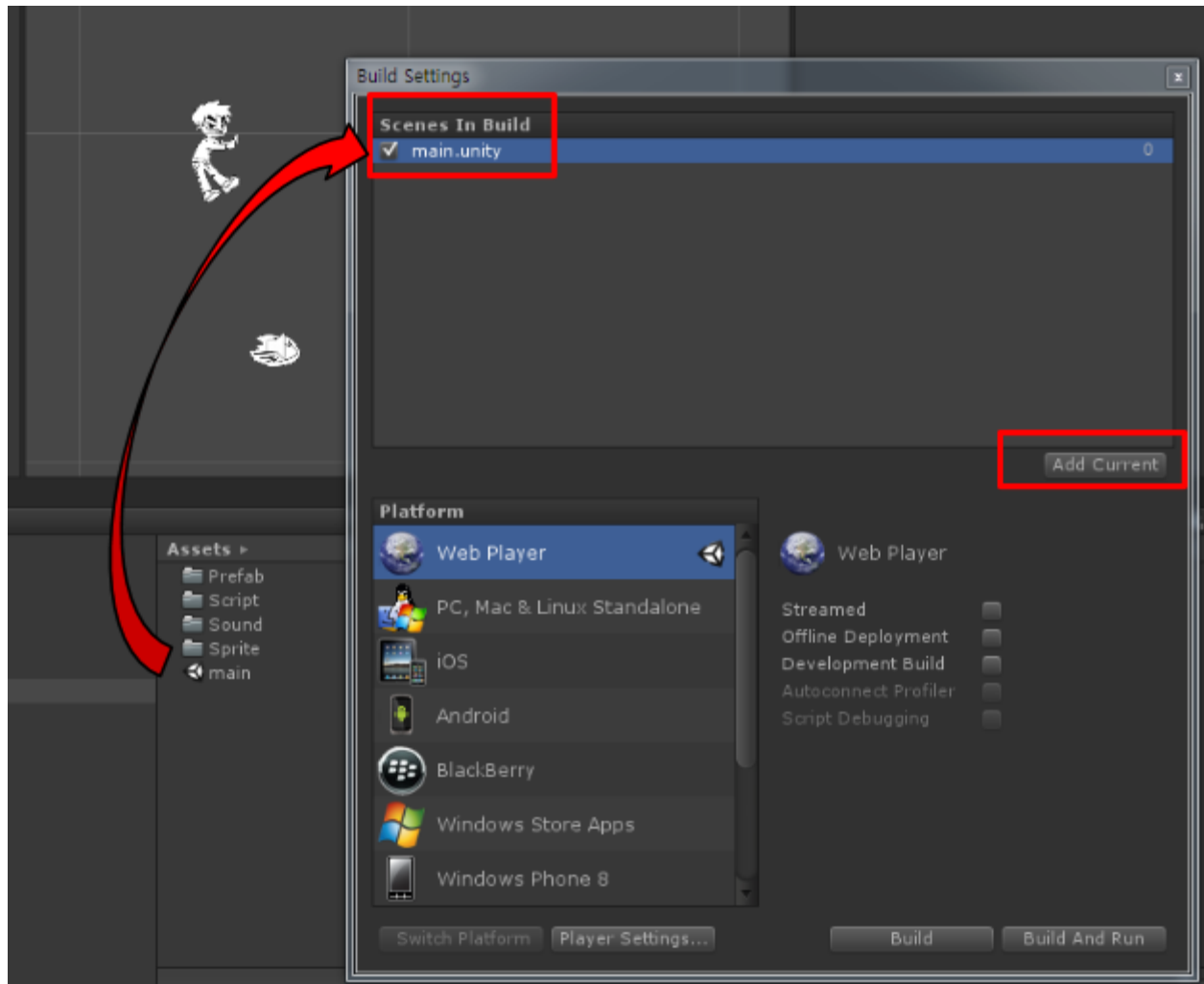
gameover 오브젝트에도 총을 쏠 때와 마찬가지로 Lose 소리를 적용해준다.

◆ 배포

이것으로 게임이 완성되었고 마지막으로 게임을 배포해 보도록 하자.



File -> Build Settings를 누르면 위와 같은 창이 뜨게 된다. 여러가지 플랫폼으로 배포가 가능한데 이 튜토리얼에서는 Web으로 Build 해보도록 하겠다.



게임이 정상적으로 실행되도록 하려면 포함된 Scene을 포함시켜 줘야 한다. 현재 그 Scene을 작업하고 있다면 Add Current 를 클릭하면 되고 다른 Scene들을 추가 시켜주고 싶다면 Scene을 Drag & Drop 하면된다.



Build가 정상적으로 뒀다면 브라우저에서 게임이 작동하는것을 볼 수 있다.

참고

<http://docs.unity3d.com/Manual/PublishingBuilds.html>