## ◆ Toy Apps

email : store@toy-apps.com
asset store : https://www.assetstore.unity3d.com/en/#!/publisher/7834

## ◆ License



## ◆ Before Start..

This project is a simple example program using Unity 2D functions with c# language. Even though this information document helps you to make a simple game step by step, it does not contain any grammartical explanation of c# or basic Unity instructions.

If you need any help with c# or basic Unity, check out these websites prior to beginning.
+ 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

If you have any question, please don't hesitate to contact to store@toy-apps.com. I might not be able to answer all the questions since I'm still learning Unity programming, but I will try my best to answer your questions. (You can ask questions either in English or Korean)

You can also find other simple projects from here. We have more paid assets than free ones, but we are working on more free projects currently. So please Check out the page often for more free projects.

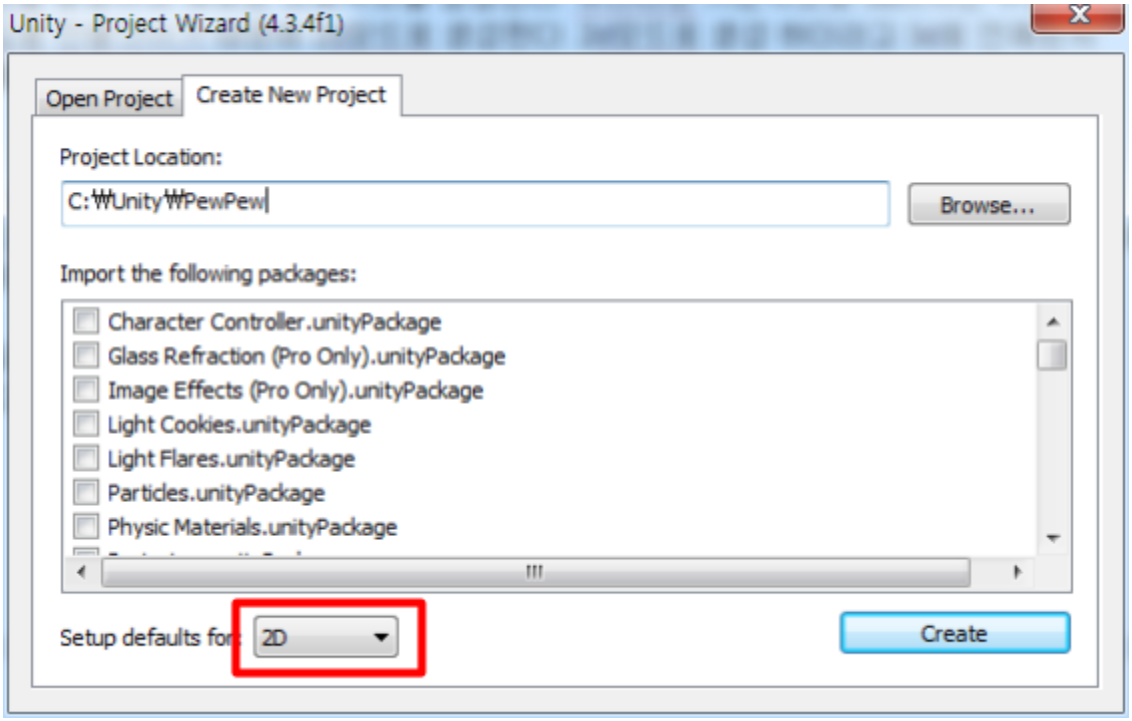
## ◆ Project preview

This is a tutorial for a simple defence shooting game.

Gameplay Demonstration : http://toyapps.github.io/PewPewPlayer/
Play video clip:
http://www.youtube.com/watch?v=lrp3it8Udxc&list=PLFXLeC2Hh_3db7XFynySEMJkkQ6umELpC&index=1

## ◆ Create a Project

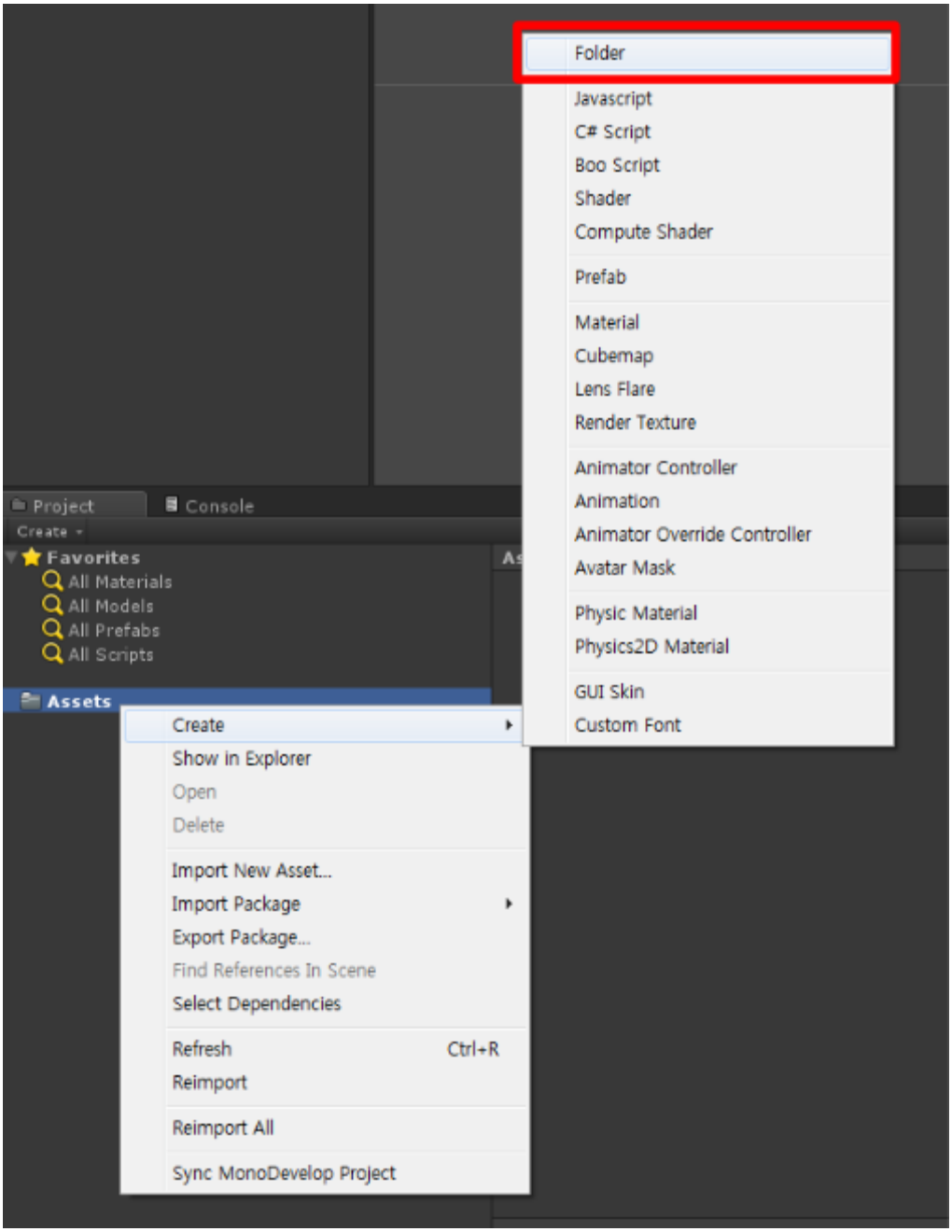
### Create a Project



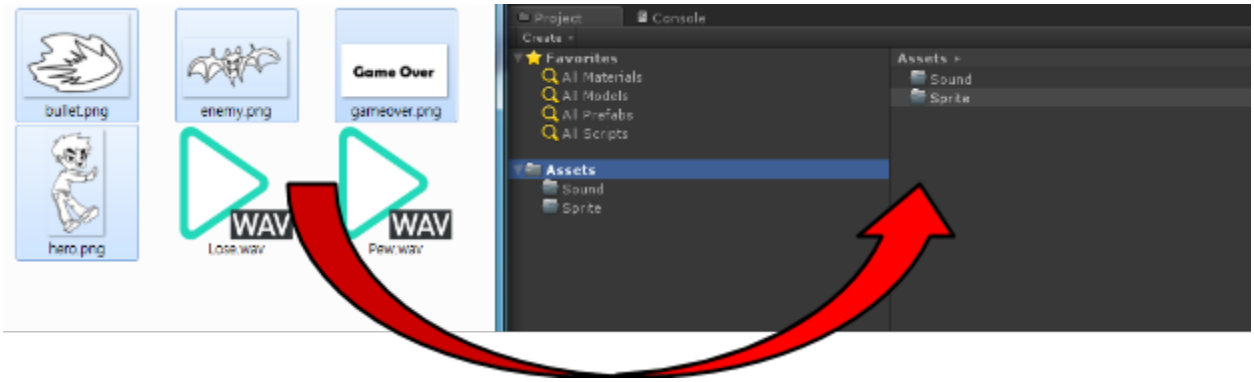

Run Unity and create a new project. Preferentially Unity offers 3-D programming, however we are going to make a 2-D game, select 2-D mode and click Create button to make a project.

### Resource import

Unlikely to other programs, Unity manages all the files through project. Resource files are managable in Unity project, but these must go through import step first.




First, select Create -> Folder from the inside of Assets folder then create Sprite and Sound Folder (You can choose any name you want).
When you unzip the resource.zip which you have downloaded, then you can see the image files and sound files.

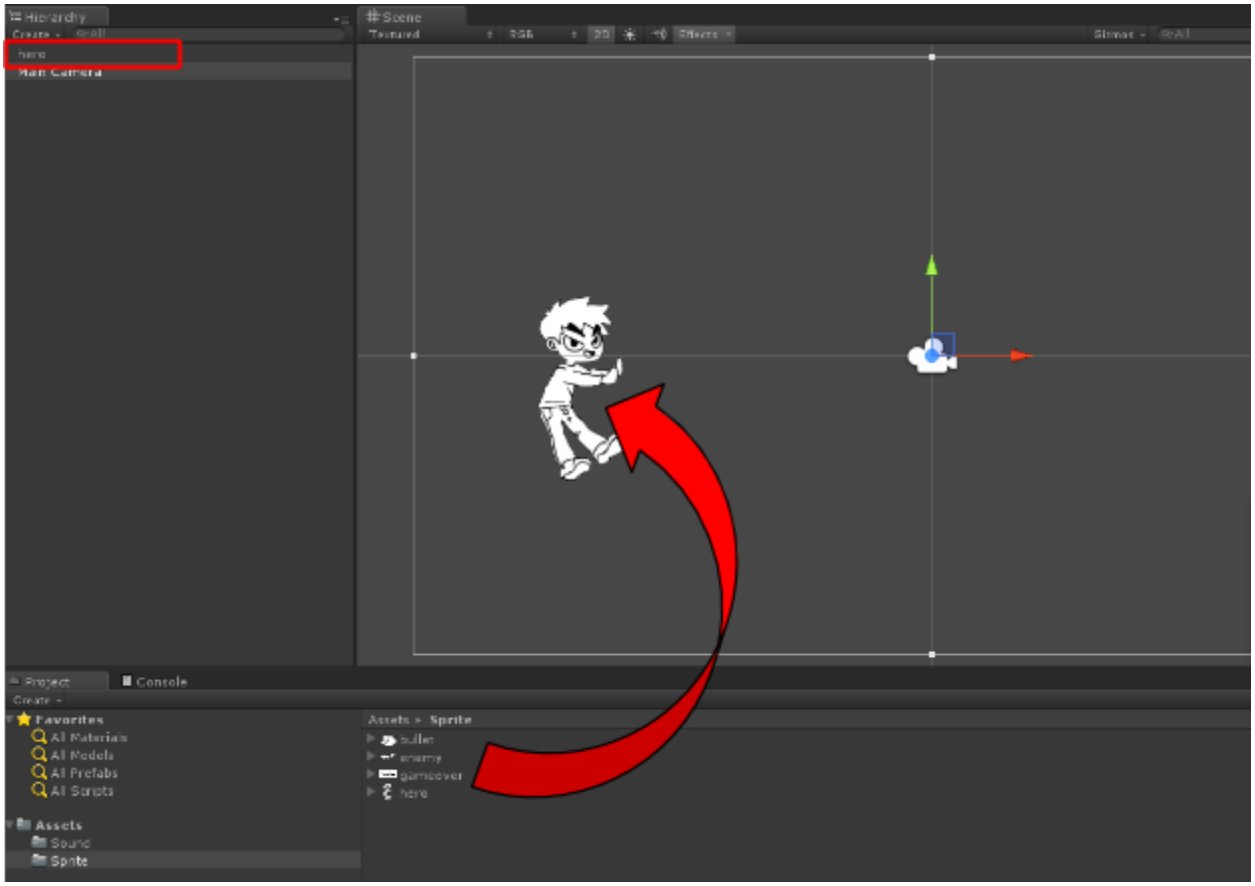



You can easily import them by Drag & Drop the image files (png) into Sprite folder and sound files (wav) into Sound folder. Never move or copy these files. Files in the Assets are managed separately in Unity system. If these files are moved or name change occures outside of Unity system, then the links between them will be all messed up and it will not be retrieved. So please do those works only within the Unity system.

### Reference

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

## ◆ Create a Character

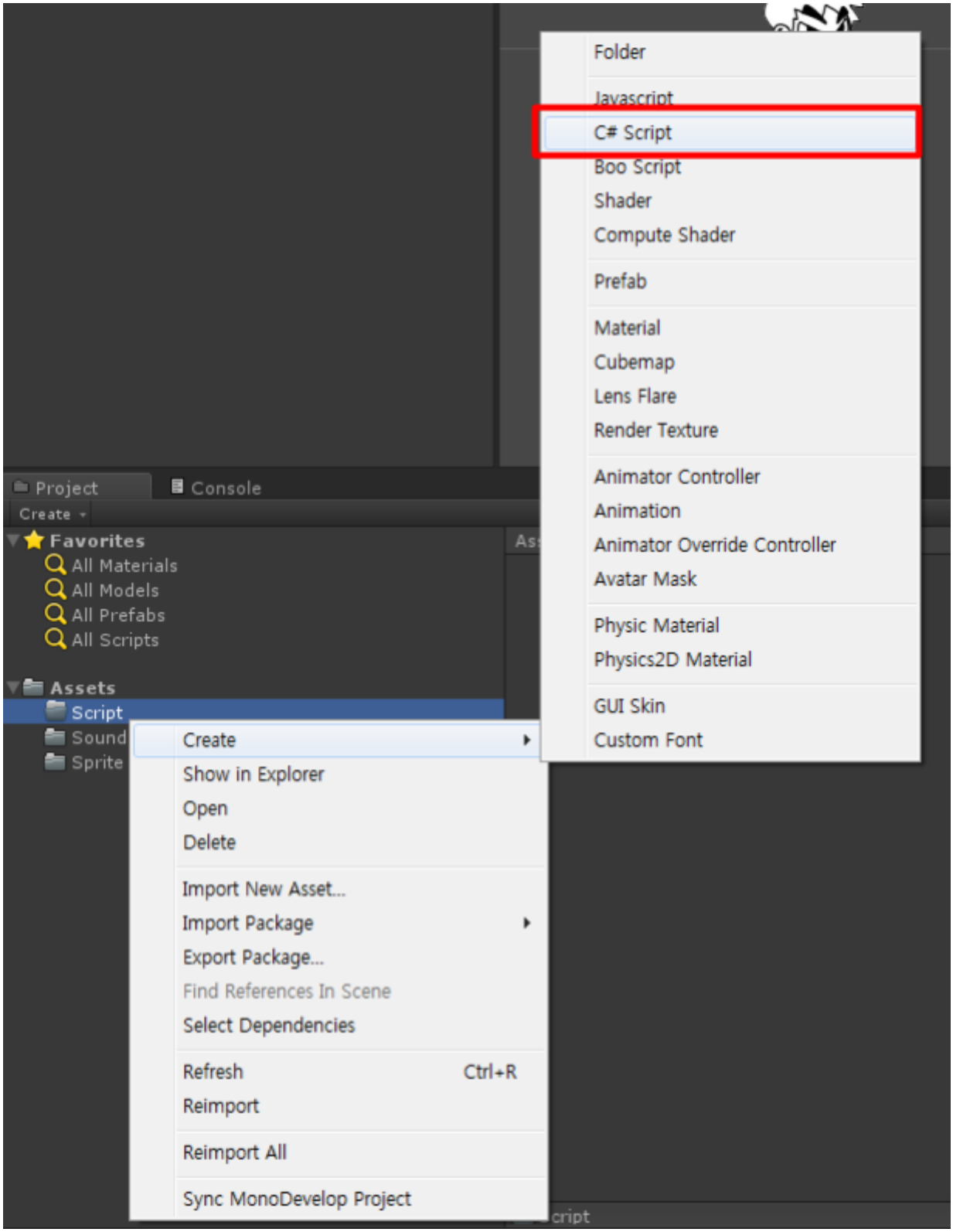



Among the previously imported image files, find hero.png and Drag & Drop on the Scene. Then you will see the Hero on the Scene and you can also find Hero on Hierarchy too.
Even though you are seeing the objects on the Scene, if the Camera is not facing the objects then they might not be shown on the screen while playing. So always check if objects are properly located infront of the Camera.

## ◆ Firing bullets

### Create a bullet on Scene

As beginning, we are going to make the main character shoots bullets. Similar as making a Hero in previous steps, Drag & Drop the bullet to insert on the Scene.


Folder
Javascript
C# Script
Boo Script
Shader
Compute Shader
Prefab
Material
Cubemap
Lens Flare
Render Texture
Animator Controller
Animation
Animator Override Controller
Avatar Mask
Physic Material
Physics2D Material
GUI Skin
Custom Font
Project
Console
Create
Favorites
All Materials
All Models
All Prefabs
All Scripts
Assets
Script
Sound
Sprite
Create
Show in Explorer
Open
Delete
Import New Asset...
Import Package
Export Package...
Find References In Scene
Select Dependencies
Refresh Ctrl+R
Reimport
Reimport All
Sync MonoDevelop Project

## Create Bullet Script

Make a Script folder. Select Create -> C# Script and name it as Bullet. When you double click this file, it will run MonoDevelop and you can see the script that you just created.

```
using UnityEngine;
using System.Collections;

public class Bullet : MonoBehaviour
{
	// 1
	void Start ()
	{

	}
	// 2
	void Update ()
	{

	}
}
```

1) Runs first when the script is executed.
2) Runs on every frames.

## Bullet Movement

We are going to insert steps here to make the bullet moves from left to right.

```
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;
	}
}
```

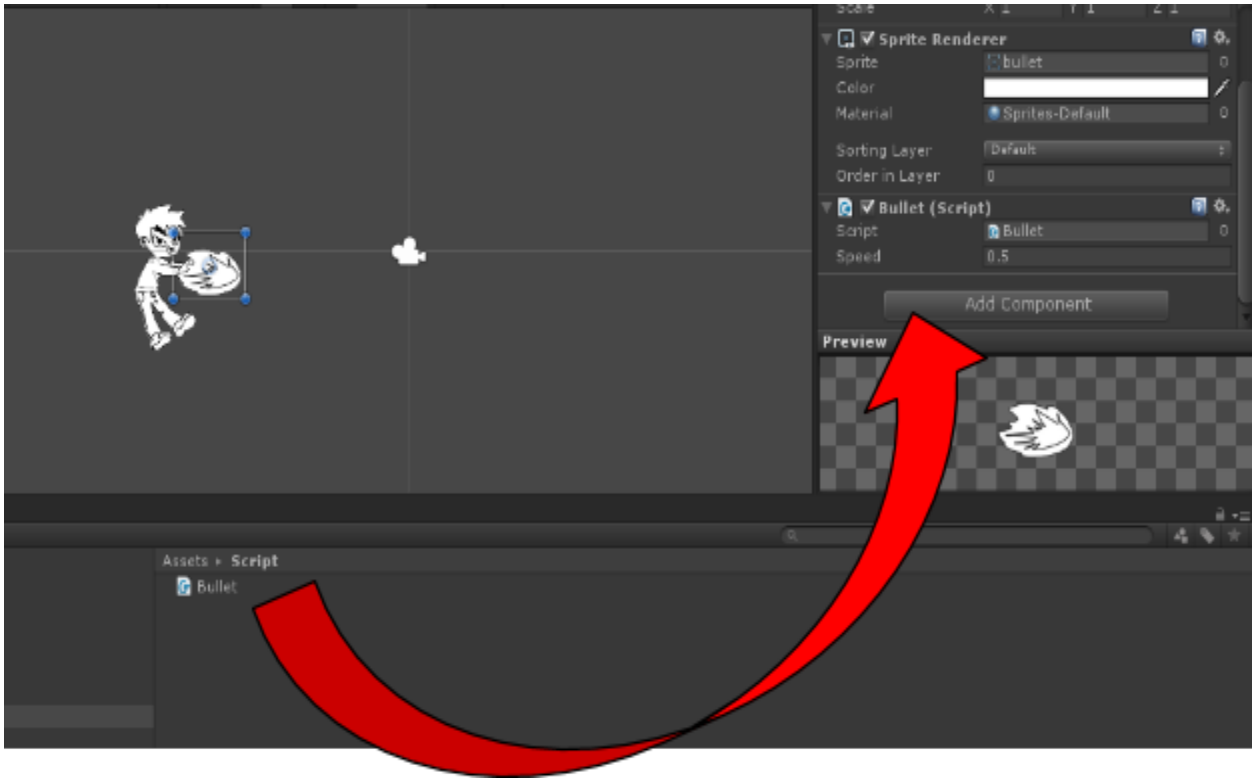
1) Set variable m_speed for bullet speed. It can be easily modified from Inspector of Scene since it is declared as public.
2) To avoid the infinite movement of bullets, use Destroy function to delete the bullet.
3) Set m_speed for the speed of the bullet to move to right side.

### Reference

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

## Applying a Script



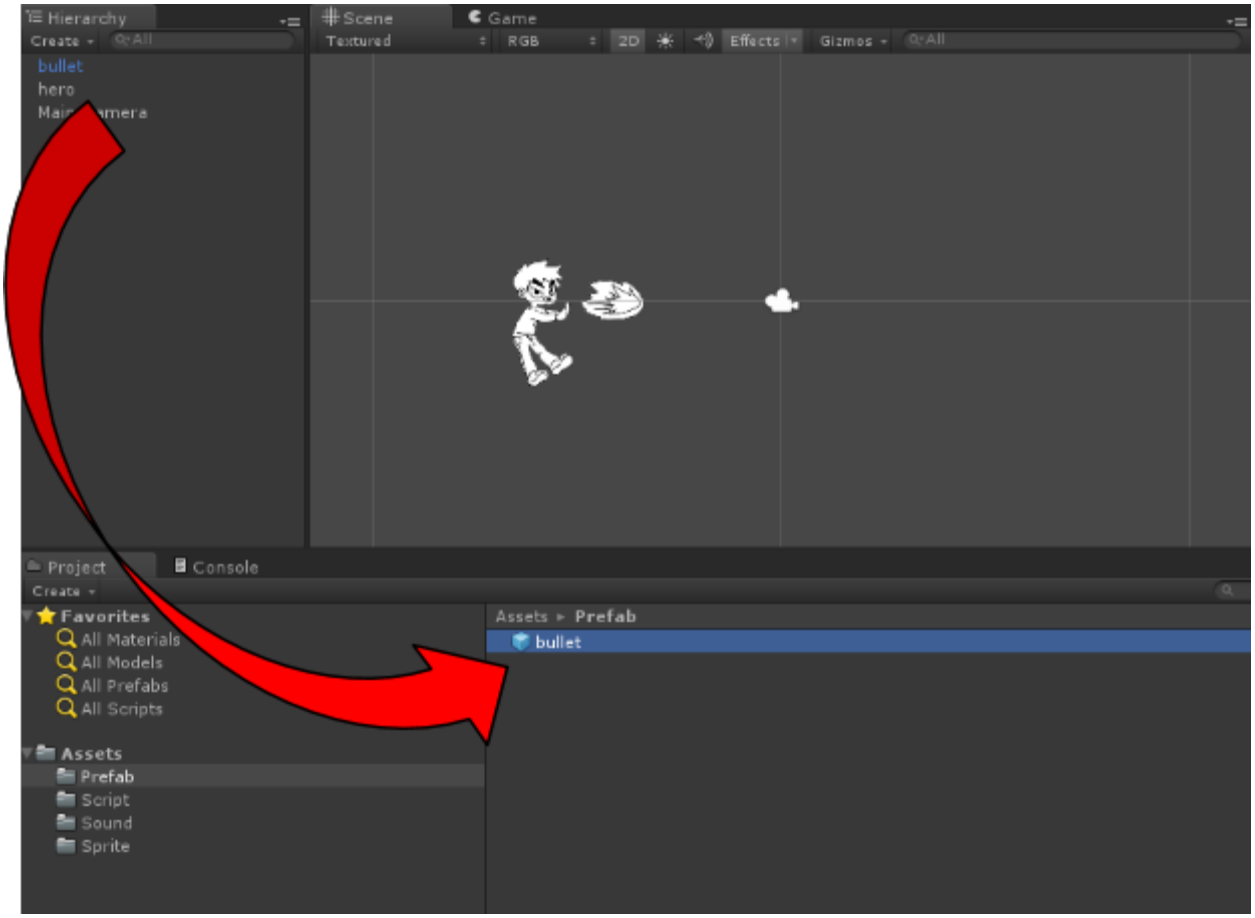

The script you just made will run if you apply it on the Sprite. To apply the Bullet Script, Drag & Drop the Script as shown on the picture above. You will see the change of speed of the bullet as you modify the Speed variable on the Inspector. Set this speed as fast as you want.

### Video Clip

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

## Make Bullet Prefab

It looks insufficient to call it as a game yet, since only a single bullet runs through the screen when the game begins. So, let's make the character shoots bullets when you click the screen.




First, you should make the Bullet you just made as Prefab to shoot multiple bullets.
Make Prefab folder, then Drag & Drop the Bullet from Hierarchy. It will create the Prefab. Then, you will be able to make copy of it as many as you want if you make Prefab.

If you Drag & Drop the bullet prefab on the Scene, you can see the creation of a copy of it.

### Video Clip

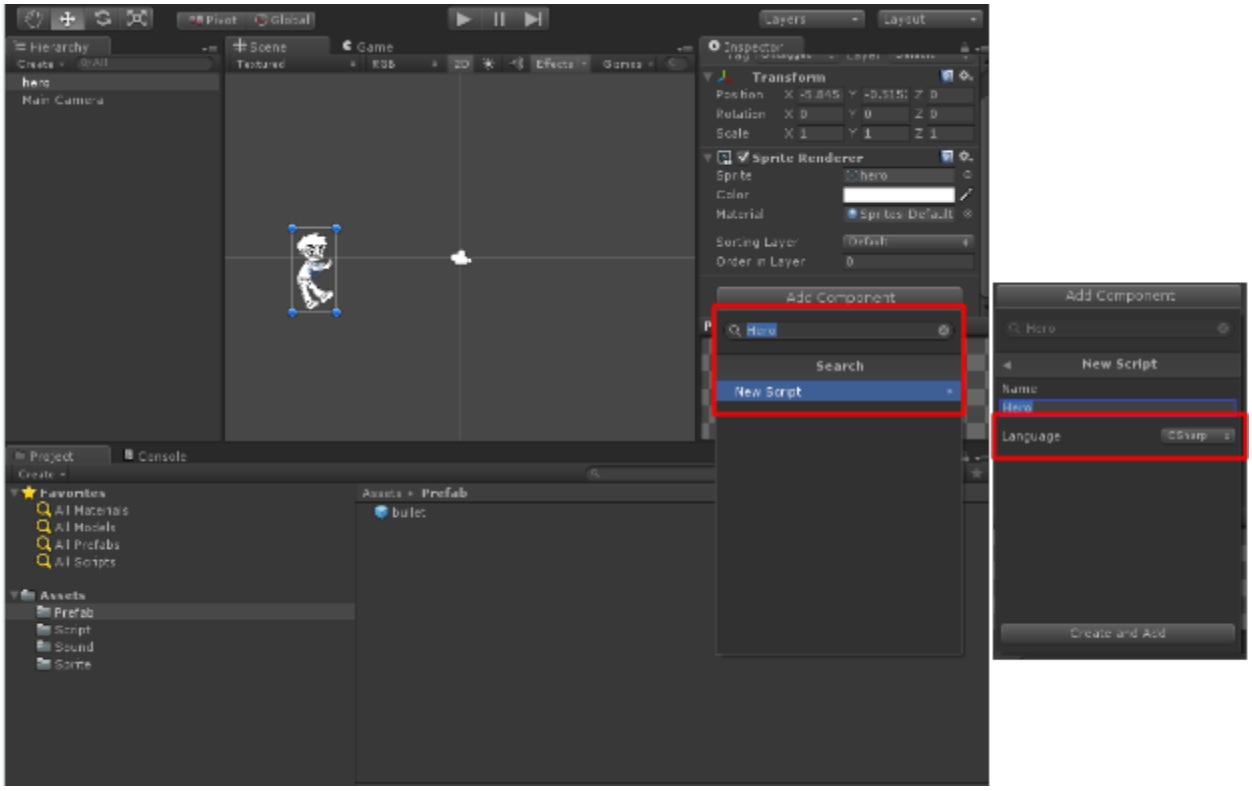
http://www.youtube.com/watch?v=2YFd0emILIw&index=3&list=PLFXLeC2Hh_3db7XFynySEMJkkQ6umELpC

### Reference

http://docs.unity3d.com/Manual/Prefabs.html
http://docs.unity3d.com/Manual/InstantiatingPrefabs.html

## Bullet shooting by mouse click

On this step, we are going to add Script on Hero to shoot a bullet by mouse. At this time, let's add Script differently.

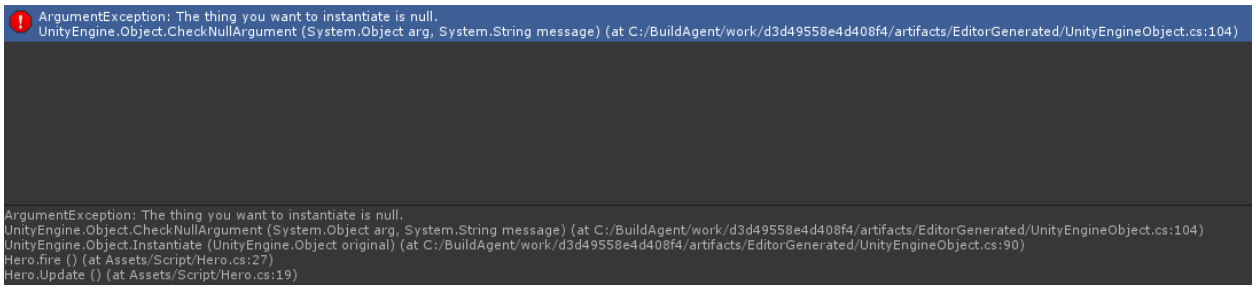



Select Hero first, then click Add Component from the Inspector on right side. When you type in a Script name you want to make, a New Script menu appears below. Click the New Script then click Create and Add again to create a Script. Then the Script will be created right under the Assets, so move the newly created Script into the Script folder.
As I mentioned before, Unity file movement must only happens in Unity program.
Let's double click the newly added Script to open the editor and add codes.

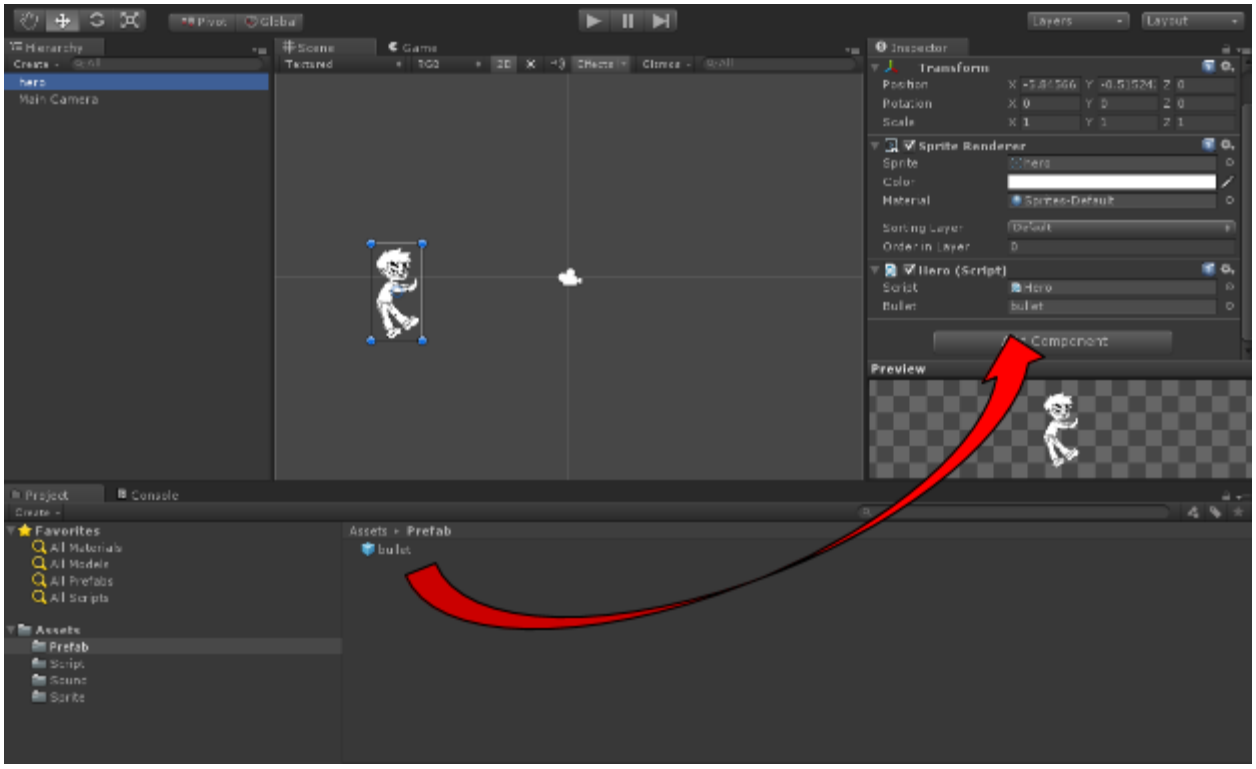
```
using UnityEngine;
using System.Collections;

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) Variable for setting properties of a bullet. Declare as public to make it possible to set it on the Scene.
2) Fire method will run when the mouse button is pushed.
3) Use m_bullet to create a new object and save it in the bullet variable. If you want to put data in a variable and you are using Instantiate, you should do Casting. It is casted as GameObject above.
4) Set the location of the newly created Object as the position of the Hero.




After you put in all the codes and run, you can see an error message as above. It is because the m_bullet variable we have created earlier has null value.



### Video Clip

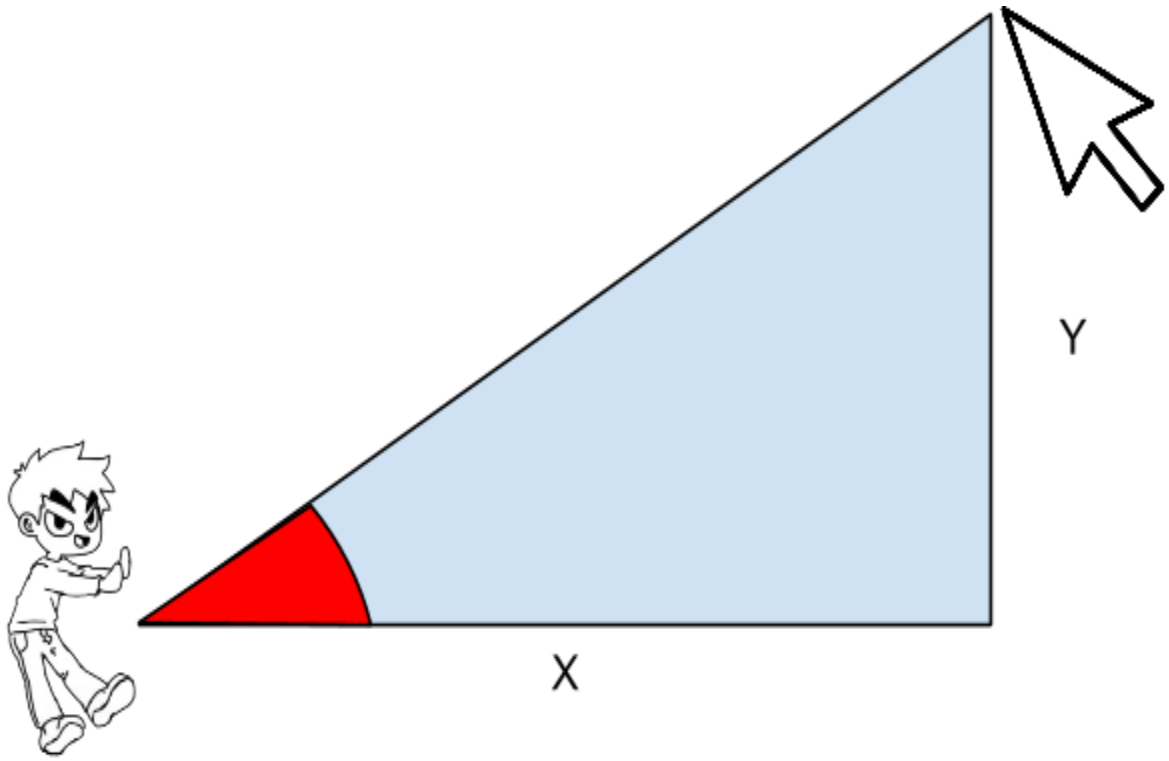
http://www.youtube.com/watch?v=2lnu05tEd18&index=4&list=PLFXLeC2Hh_3db7XFynySEMJkkQ6umELpC

### Reference

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

## Setting angle of a Bullet

Now, we can shoot a bullet, but it only moves straight to the right side and we cannot hit the enemy if they comes from above or below that straight line. This step, let's make the bullet moves to the direction of the position where mouse is pressed.
The bullet is moving to right direction already, so we only need to twist it a little bit to the mouse position to shoot it as I want.  You can easily find the angle out by using trigonometric function.




You need to find the angle of the red part in the picture above to shoot a bullet. First, find the length of x and y between the hero and the mouse position. Length of x can be easily found by [ x-coordinate of mouse position - x-coordinate of the Hero]. And same for the length of y. Now, we are able to use Atan2 function to find angle with the length of x and y.

The below is the example.

```
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) Use ScreenToWorldPoint function to get the coordinate of mouse.
2) Find the angle as explained previously.
3) Turn the created bullet as much as the angle we calculated above. The result from Atan2 function is in Radian, you have to use Rad2Deg function to change it to Degree.

### video Clip

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

### Reference

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

## ◆ Create Enemy

Enemy is similar to the motion of a bullet. As bullet, Drag & Drop the image of Enemy and create and apply an Enemy Script.

### Movement of Enemy

Unlike to bullets, enemy will move right to left. Remove x-coordinate to make it move to left every frame.

```
public class Enemy : MonoBehaviour
{
        void Start ()
        {
```

```
        }

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

1) Enemy moves to left by -0.1f at a time.

## Dynamic movements of Enemy

It will be boring if all the enemies come with same speed. To add a little more tension to the game, let's set the speed of enemy differently everytime they appears.

```
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) Minimum speed of the enemy. Declare as public to be able to modified on the Scene.
2) Maximum speed of the enemy. Declare as public to be able to modified on the Scene.
3) Variable to store the speed of the enemy.
4) Use Range function to get the random value between m_minSpeed and m_maxSpeed, then store it in m_speed variable.
5) Use Log message to check if the speed is changing properly.

6) Move as fast as the randomly chosen speed.
Check out the video clip below to see the speed change occures on every execution. You can modify the speed as fast as you want.

### Video Clip

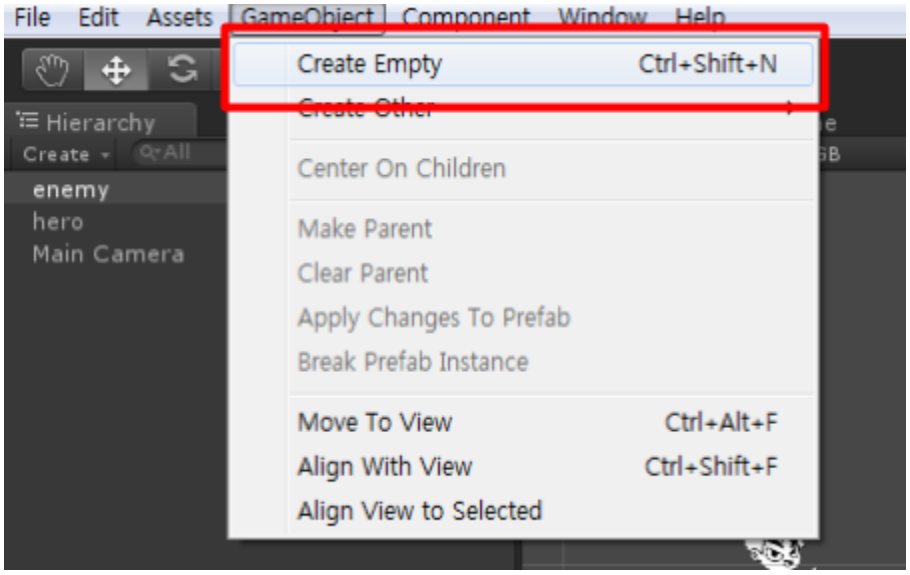
http://www.youtube.com/watch?v=kRig2Q71BIc&index=6&list=PLFXLeC2Hh_3db7XFynySEMJkkQ6umELpC

### Reference

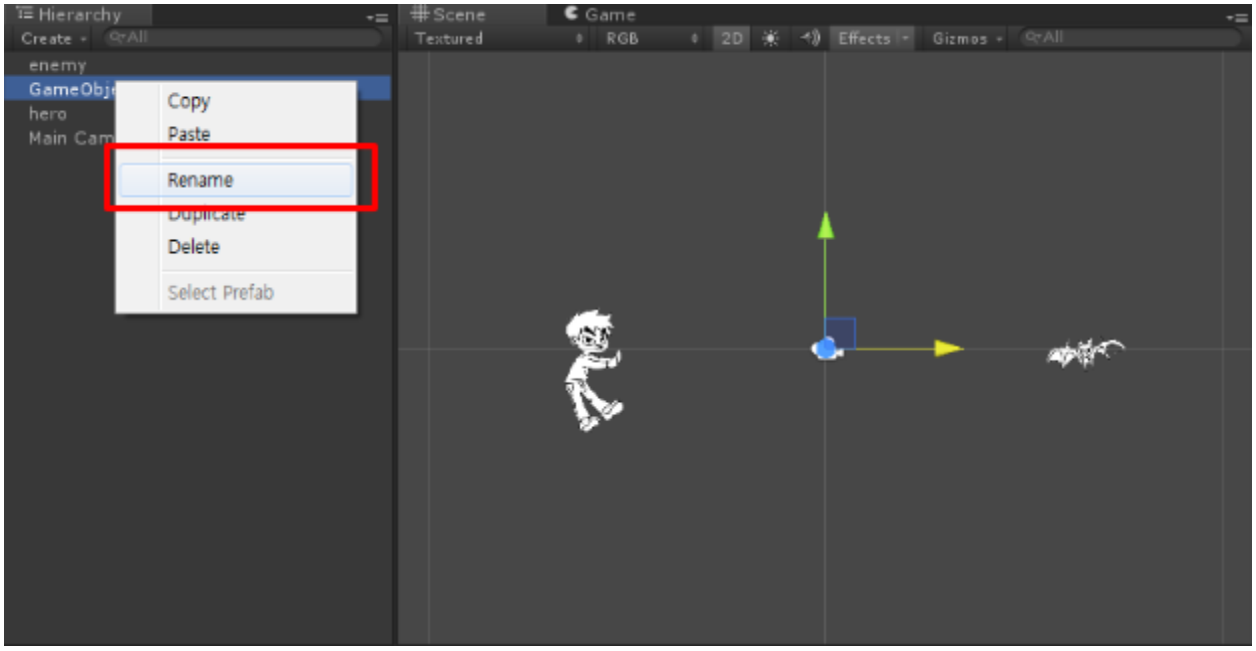
http://docs.unity3d.com/ScriptReference/Random.Range.html
http://docs.unity3d.com/ScriptReference/Debug.Log.html

## Dynamic position of enemies

After setting the speed, make enemy to Prefab. As we made a bullet before, just Drag & Drop in the Prefab folder locating right below the Assets. Once again, we will create multiple number of enemies as we did for bullets.




To make a script for creating an enemy, select GameObject -> Create Empty to create an empty object.



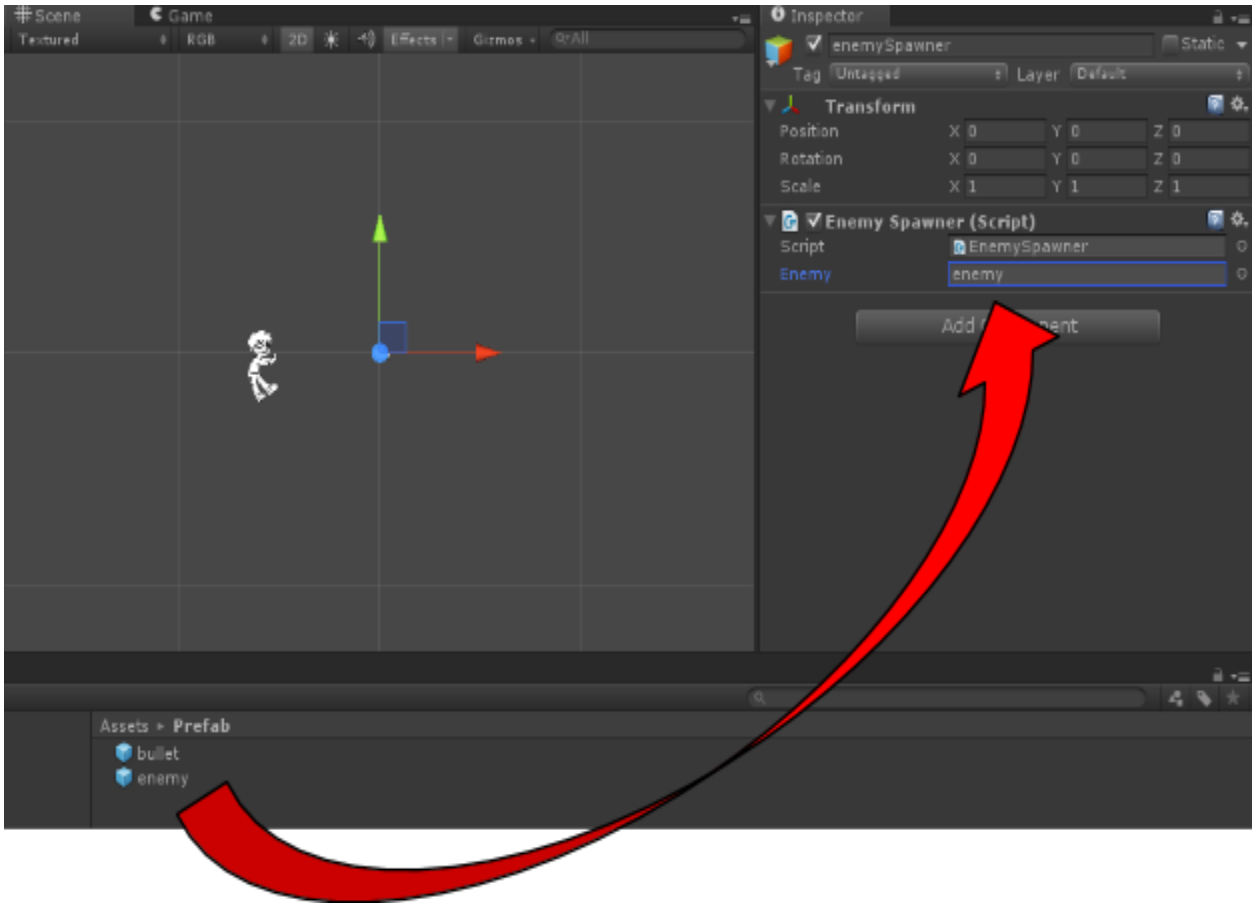

Now You will see a new object named GameObject on Hierarchy, but you might get confused with the other objects. Rename the newly formed GameObject by right click on the name.

Then create EnemySpawner Script and apply it on the Object you created earlier.

```
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) Declare a variable to be able to modify on the Scene
2) Run Coroutine for creating an enemy
3) Coroutine function to create an enemy
4) Use while-loop to create enemies infinitly
5) Pause Coroutine for 1 sec
6) Use m_enemy to create a new Object
7) Set coordinate of the newly created Obejct. We will set it with the coordinate of current Object temporarily.




Finally, Drag & Drop the enemy Prefab that was created earlier.

You can see the new enemies appearing every seconds when you run the game. However, it is very monotonous and boring since enemies are only coming from a point.

### Video Clip

http://www.youtube.com/watch?v=rpLZMfbeY40&index=7&list=PLFXLeC2Hh_3db7XFynySEMJkkQ6umELpC

### Reference

http://docs.unity3d.com/Manual/Coroutines.html
http://docs.unity3d.com/ScriptReference/MonoBehaviour.StartCoroutine.html
http://docs.unity3d.com/ScriptReference/WaitForSeconds.html

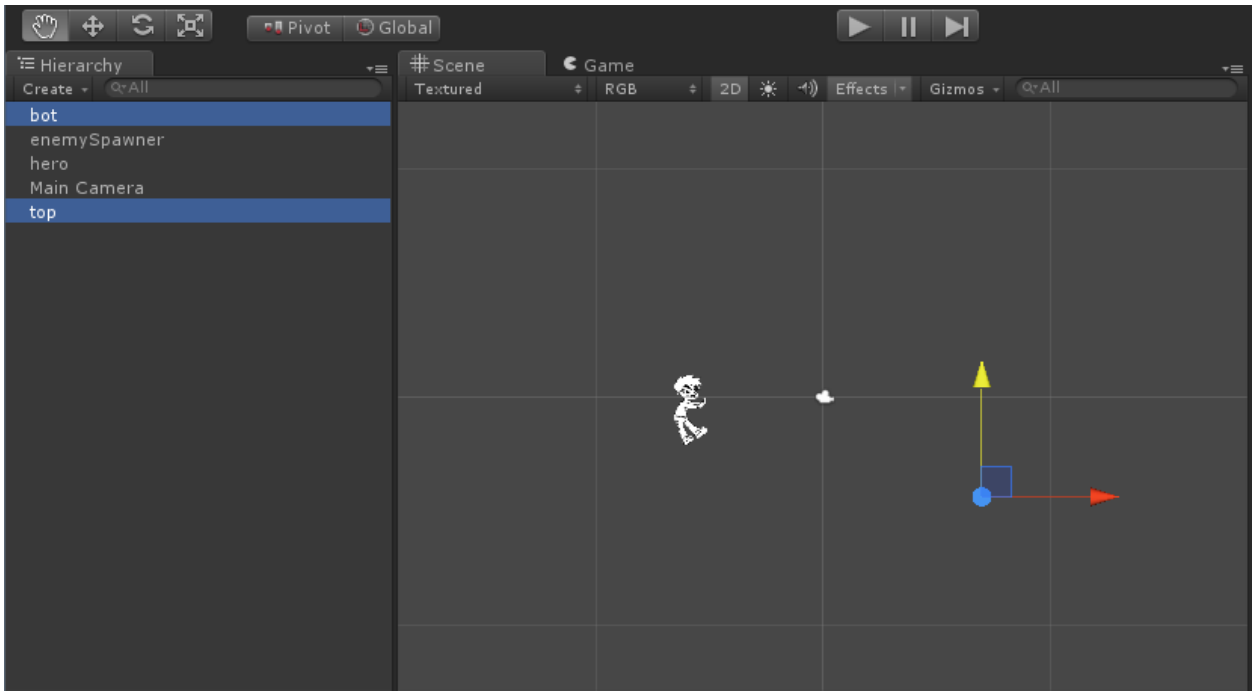
## Setting Enemy Position

In this step, we will make the enemies to form randomly from some range of positions. We can just force to insert some numbers in it, but we will create transform to set number for the future. In this way, we will easily change positions even later on.

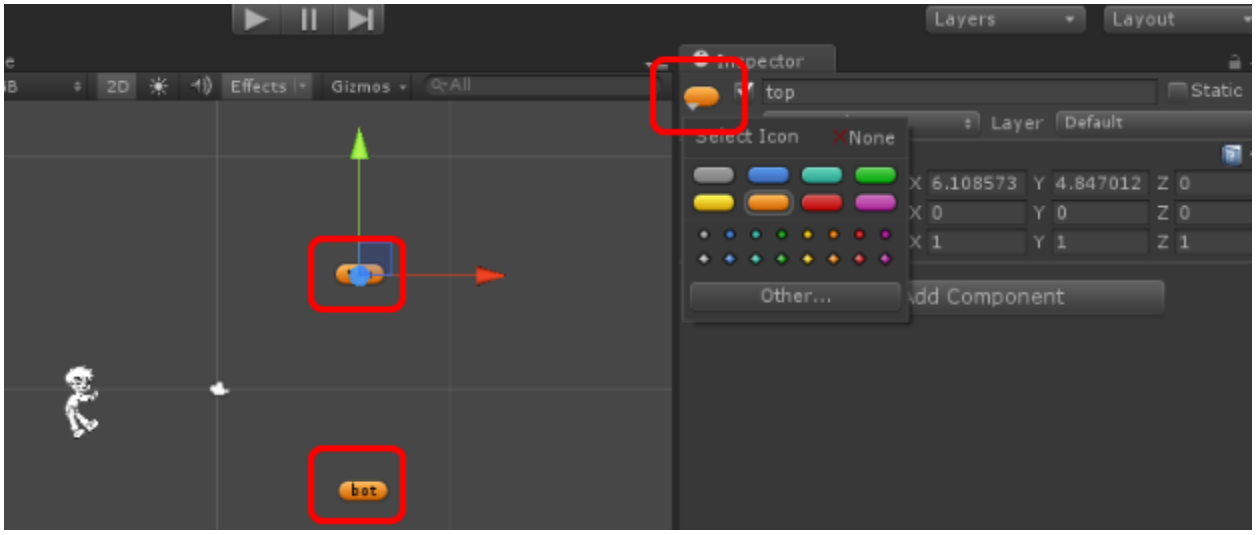
```
using UnityEngine;
using System.Collections;

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) );
		}
	}
```

```
}
```

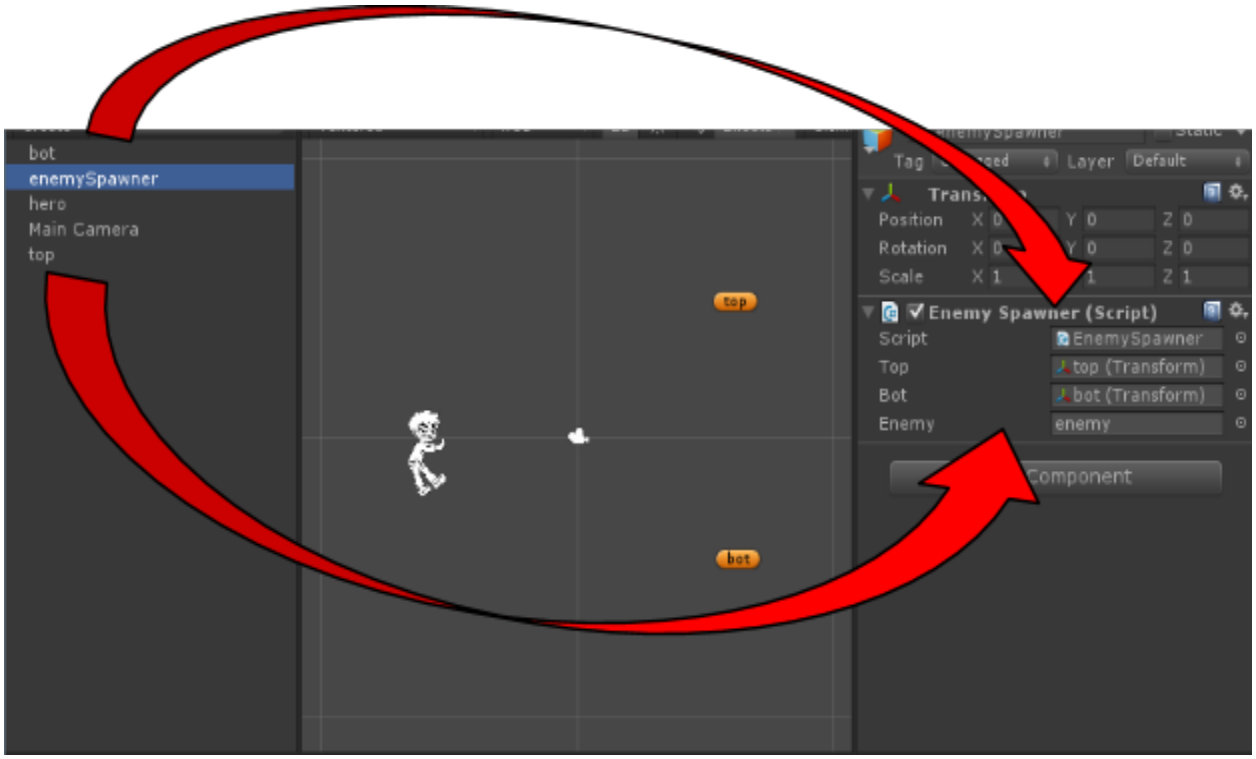
1) Make two public transforms so we can set it on the Scene. We will let the enemies to be created between these two transforms.
2) Randomly set y-coordinate within these two transforms.




Create transforms that will become the boundary. It can be formed by clicking GameObject -> Create Empty.
Create two of them and name them as top and bot. Locate them on adequate position. It can be awkward to setting them since these two objects are invisible.




If you use this function (red box above) on the Inspector, it is easy to modify the position since they become visible. This is not going to be shown on the actual game.




(Drag & Dropping)
Lastly, Drag & Drop the top and bot on Script. You will be able to see the enemies forming from the setted position when you run the game.

### Video Clip

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

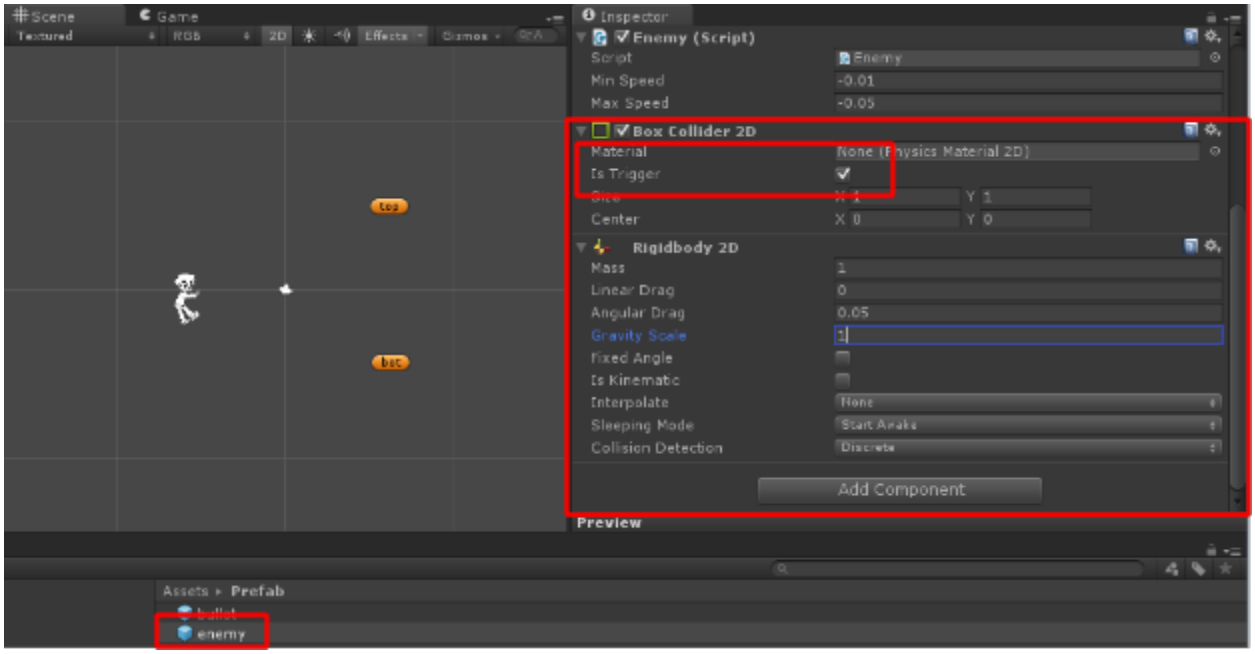
### Reference

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

## ◆ Removing Enemy

### Modifying Enemy

No matter how many bullets you are shooting to the enemies, they are invulnerable currently. So we will remove them after being shotted.



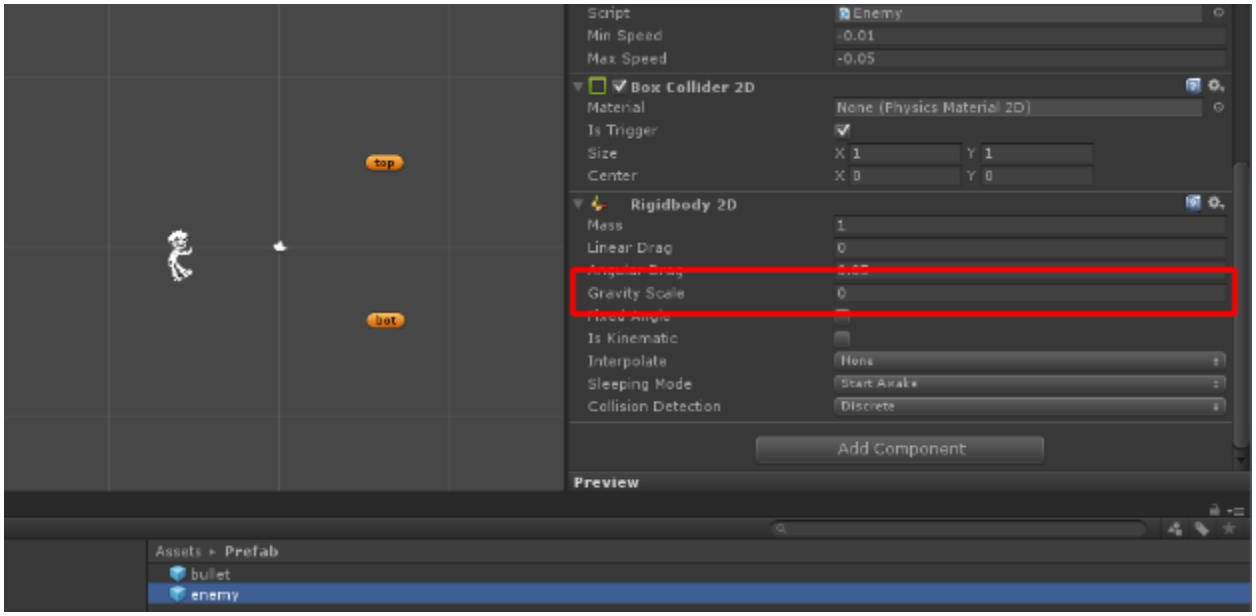

First, we need to add Collider and rigidbody to check collision. Since we are working on 2-D project, add 2-D as above image. Then check Is Trigger box, which will only check whether two objects are touching each other or not without any physical action.

However, when you run it, you will see the enemy is falling to the ground.

### Video Clip

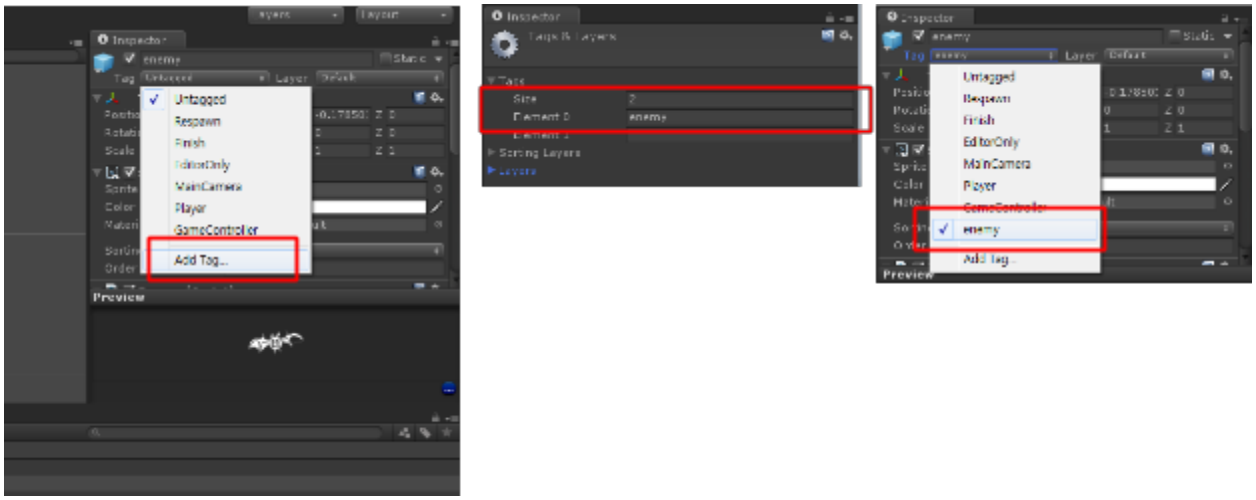
http://www.youtube.com/watch?v=B_xu1nGMARg&list=PLFXLeC2Hh_3db7XFynySEMJkkQ6umELpC&index=9

The reason is that the object became physical object by inserting rigidbody and it is falling down because of the gravity.




Since what we want is that the enemy moves to the left constant, let's set gravity to 0 so the object is not affected by the gravity.

One way to find out which object is collided with the object is to add a tag.




Let's select enemy after adding enemy tag through Add tag on the Inspector.

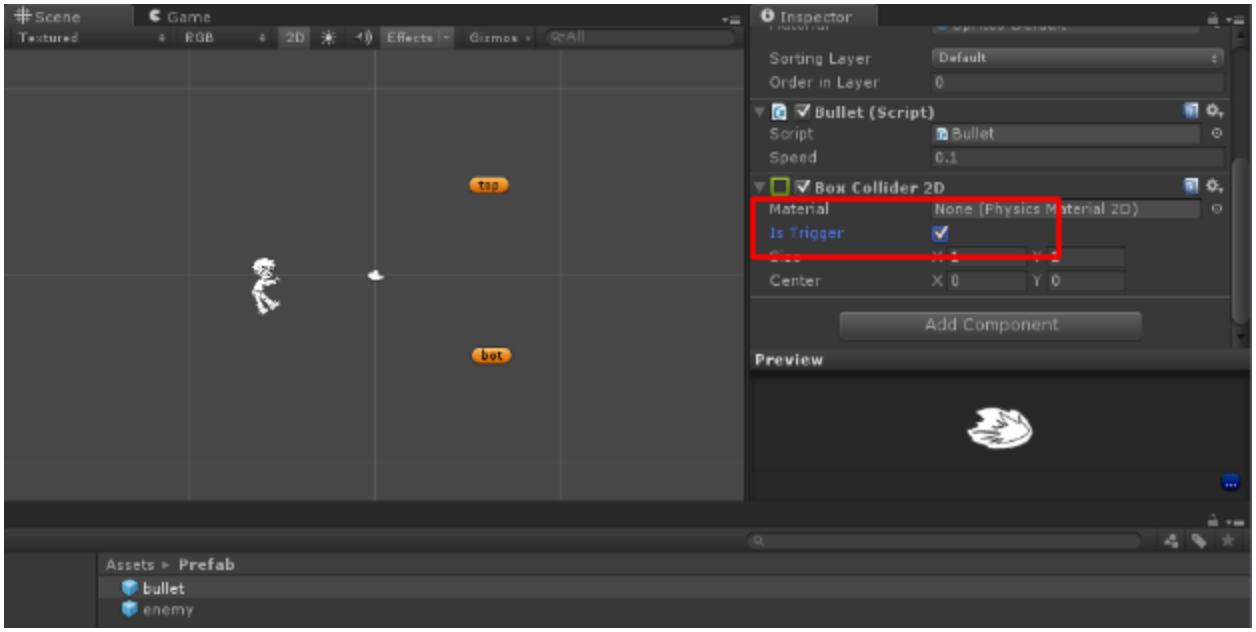
## Reference


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

## Modifying Bullet

Add Box Collider 2D only for the Bullet Prefab. You should add rigitbody if you want to make them move like physical objects, but we don't need that since we only need to check collision.




Check Is Trigger box after adding it. Then you will be able to check collision with OnTriggerEnter2D function.

```
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) OnTriggerEnter2D gets executed when collision happens between two Colliders after checking Is Trigger. Other means the collided object.
2) Check if the collided object's tag is an enemy
3) Remove both bullet and the enemy

### Video Clip

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

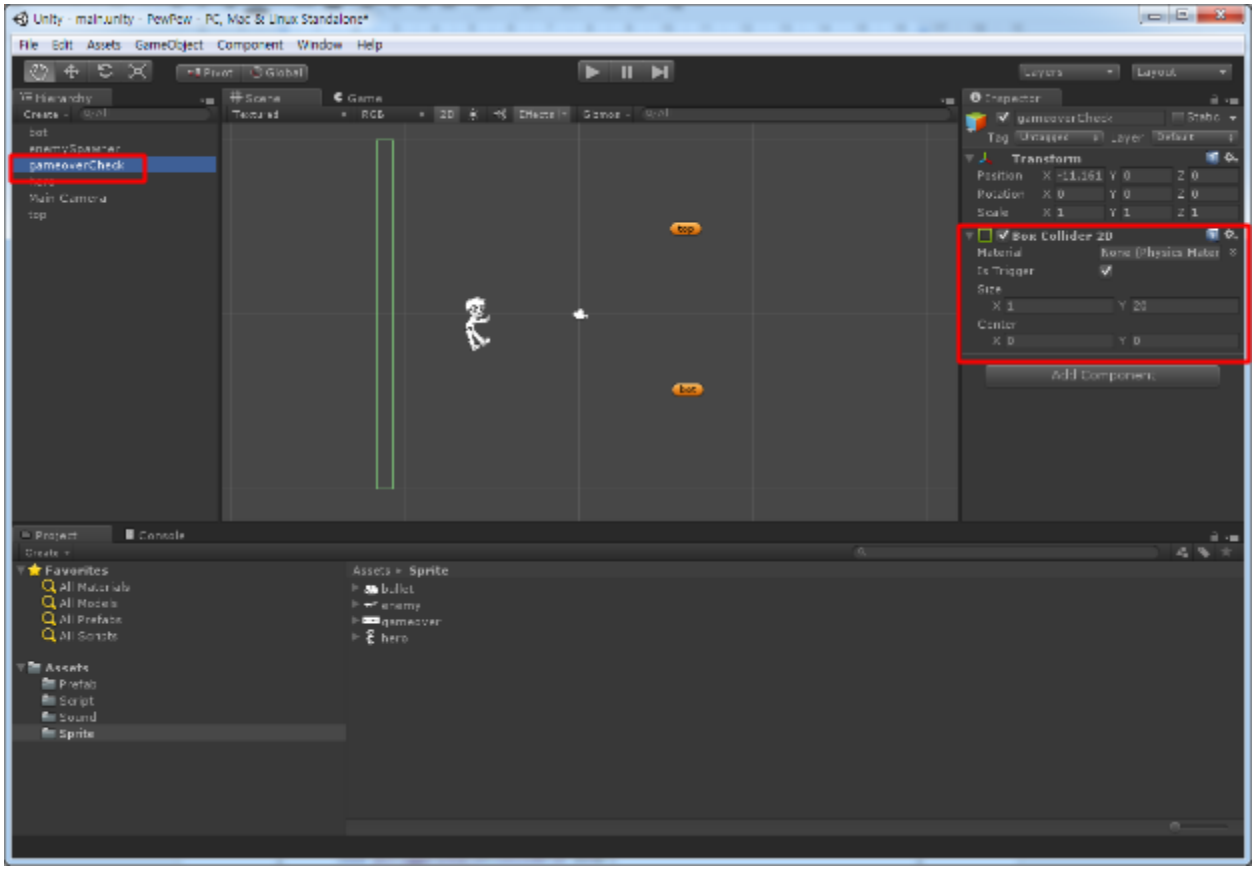
### Reference

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

## ◆ Game Over

### Remove Enemy when it Reaches the End of the Screen

Now we made the enemy disappear by shooting them, but no case of game over yet. So let's make the game ends when the enemy hits the end of the screen on the left side.




Create a GameObject through GameObject -> Create Empty, rename it, then add Box Collider 2D. Check Is Trigger box and increase the size as big as needed to check enemy.
Add a c# code called GameOverChecker then insert the codes below.

```
using UnityEngine;
using System.Collections;

public class GameOverChecker : MonoBehaviour
{
        void Start ()
        {

        }

        void Update ()
        {

        }

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

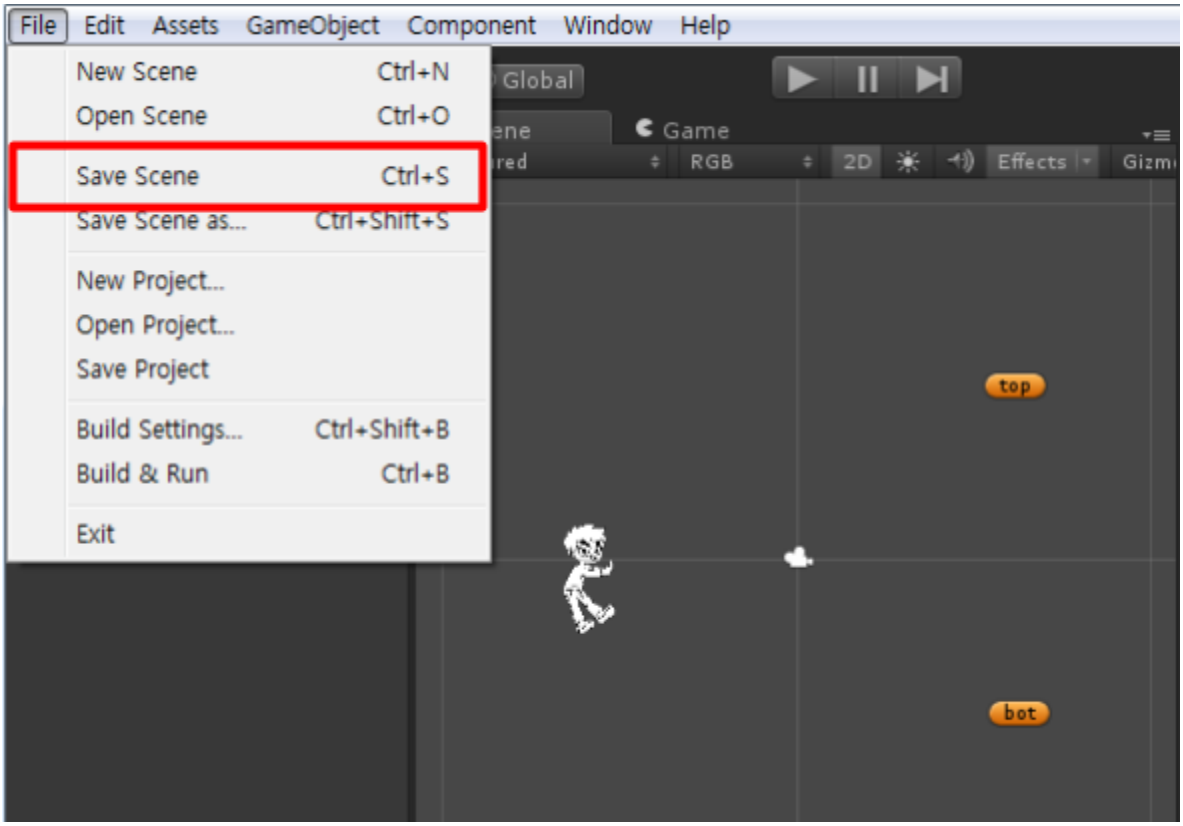
```
}
```

1) Remove enemy when it approaches the end of this screen.

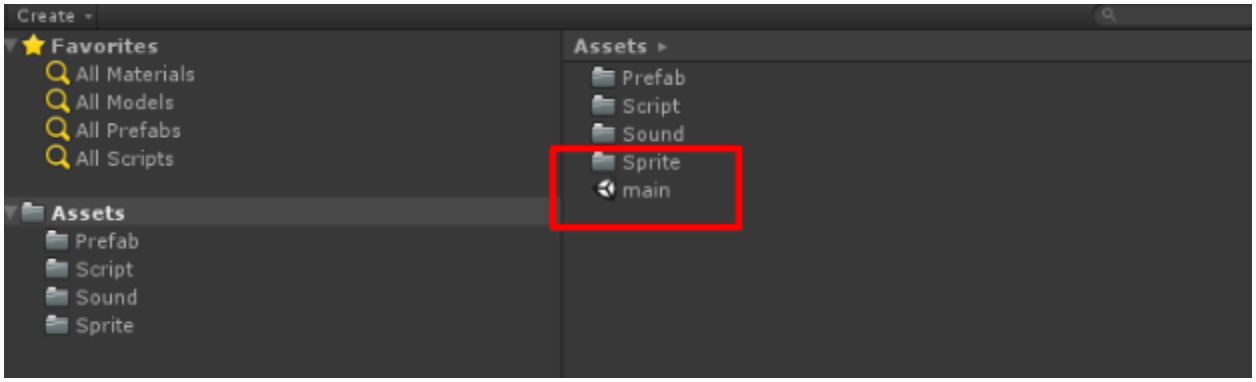
### Video Clip

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

## ◆ Game Over




Before going to the next step, save the Scene we are currently working on as ‘main’. It is good to have a habit of saving as frequent as possible, even though this is the first saving on this tutorial.




When you save a Scene, you will see it is locating in the Assets. As you can see in the image above, we have saved the Scene right in the Assets, but you can also make a separate folder for Scenes.

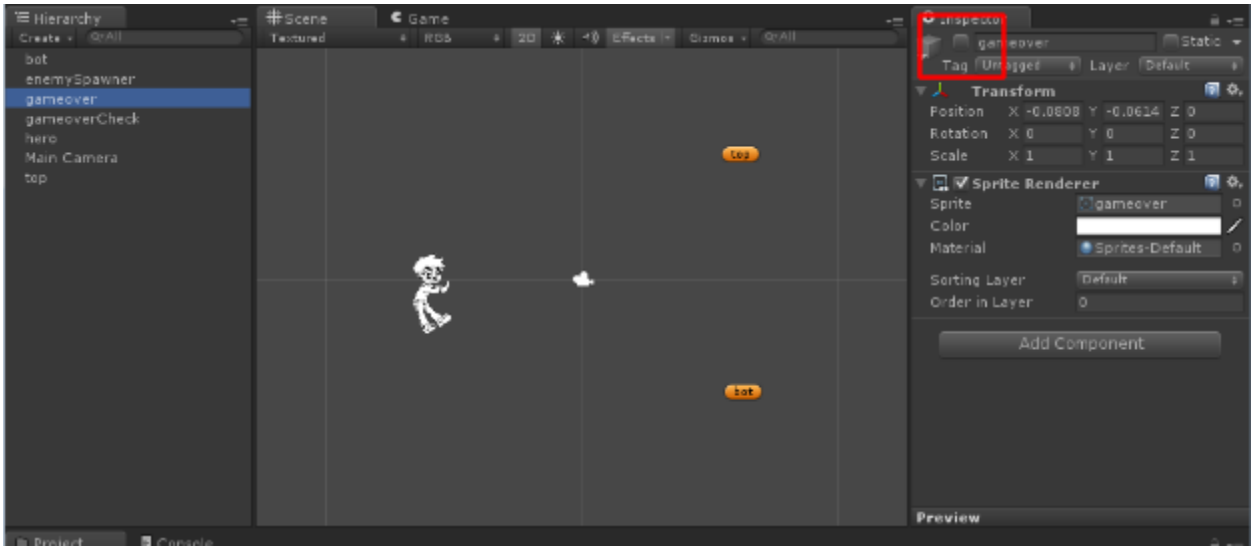
Next, we are going to remove enemy when it reaches the end of the screen.
Modify the GameOverChecker Script as below.

```
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");
```

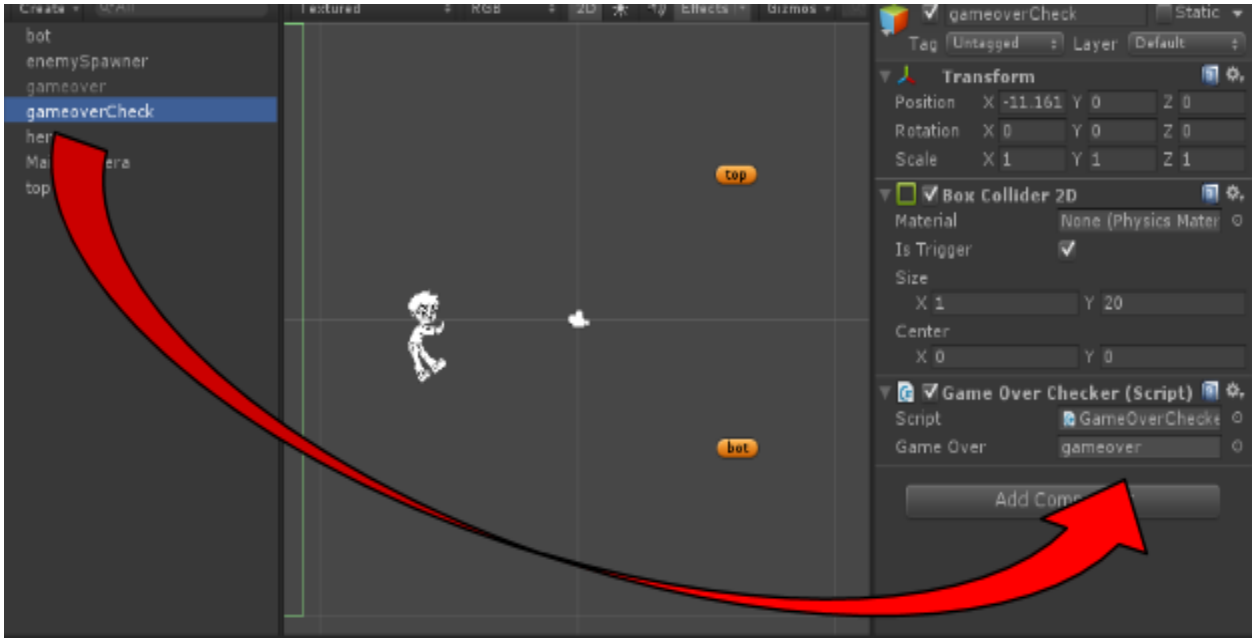
```
                        }
                }
                Destroy (other.gameObject);
        }

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

1) GameObject to launch the game over image
2) Flag to prevent the multiple check of game over
3) Check only if the tag is enemy
4) Runs only when m_isOver is false
5) Set m_isOver to true to prevent from running this if-statement again
6) Use setActive function to show game over state on screen
7) Run restartGame function by using StarCoroutine function
8) Wait for 3 sec.
9) Reload the Scene with LoadLevel function

Next, let's create GameObject to set m_gameOver. Drag & Drop gameover.png on Scene to create game over image.




Remove check mark in red box above to hide Gameover object. Removing check is same as setting setActive to false as we used setActive function on Script earlier.




Then Drag & Drop to apply the gameover object.

### Video Clip

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

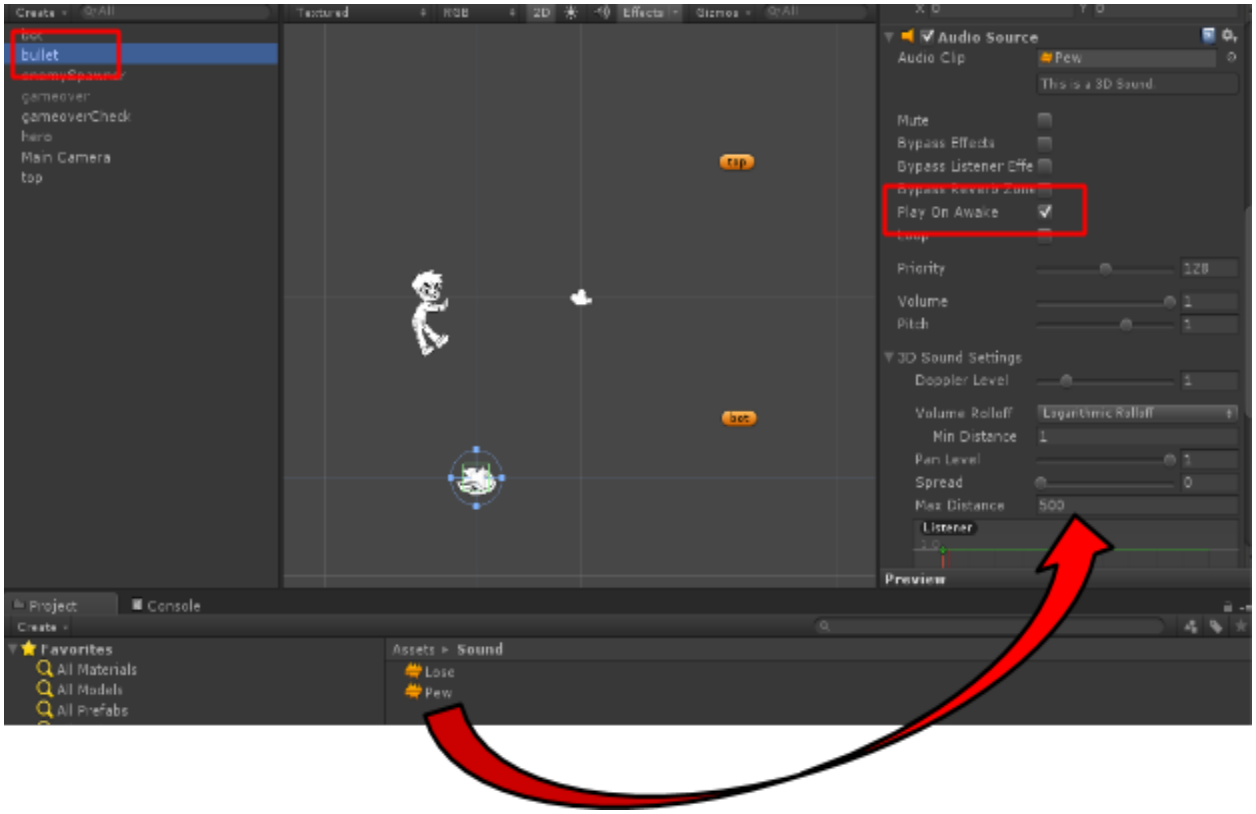
### Reference

http://docs.unity3d.com/ScriptReference/GameObject.SetActive.html
http://docs.unity3d.com/ScriptReference/Application.LoadLevel.html

## ◆ Applying Sound

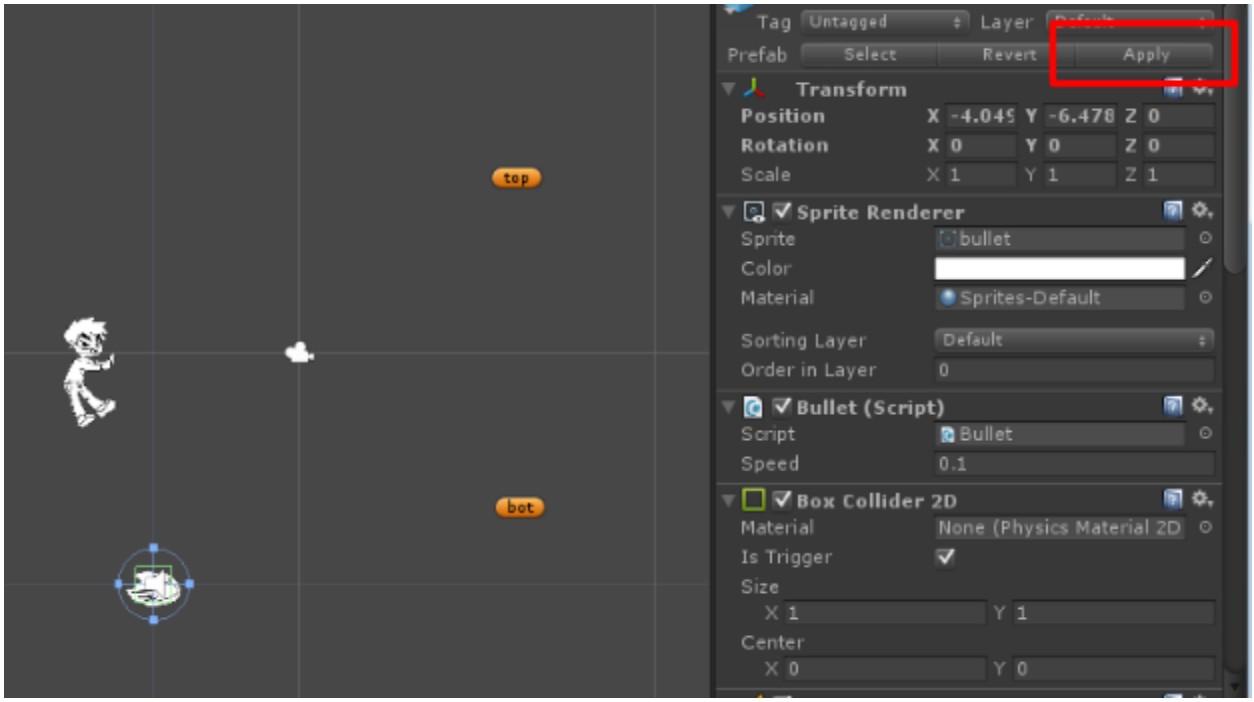
### Shooting Sound

If you want to insert sound when you are shooting bullets, play sound on the point of creation of a bullet. You can do it by modifying Bullet Prefab. We are going to modify the Prefab with different way as before. First, Drag & Drop Bullet Prefab on Scene to make another one.




Select Bullet and Drag & Drop the Pew Sound. Then Audio Source will appear. If Play On Awake box is checked, you will hear the sound on it's creation.

Though when you run the game, you will not hear any sound except for the bullet that is already created on the screen. It is because even though how many changes you make on Prefab, it does not affect on the original.




Click apply button if you want to apply changes on original as the changes you made on Prefab. Remove the bullet on the Scene after pressing apply button.

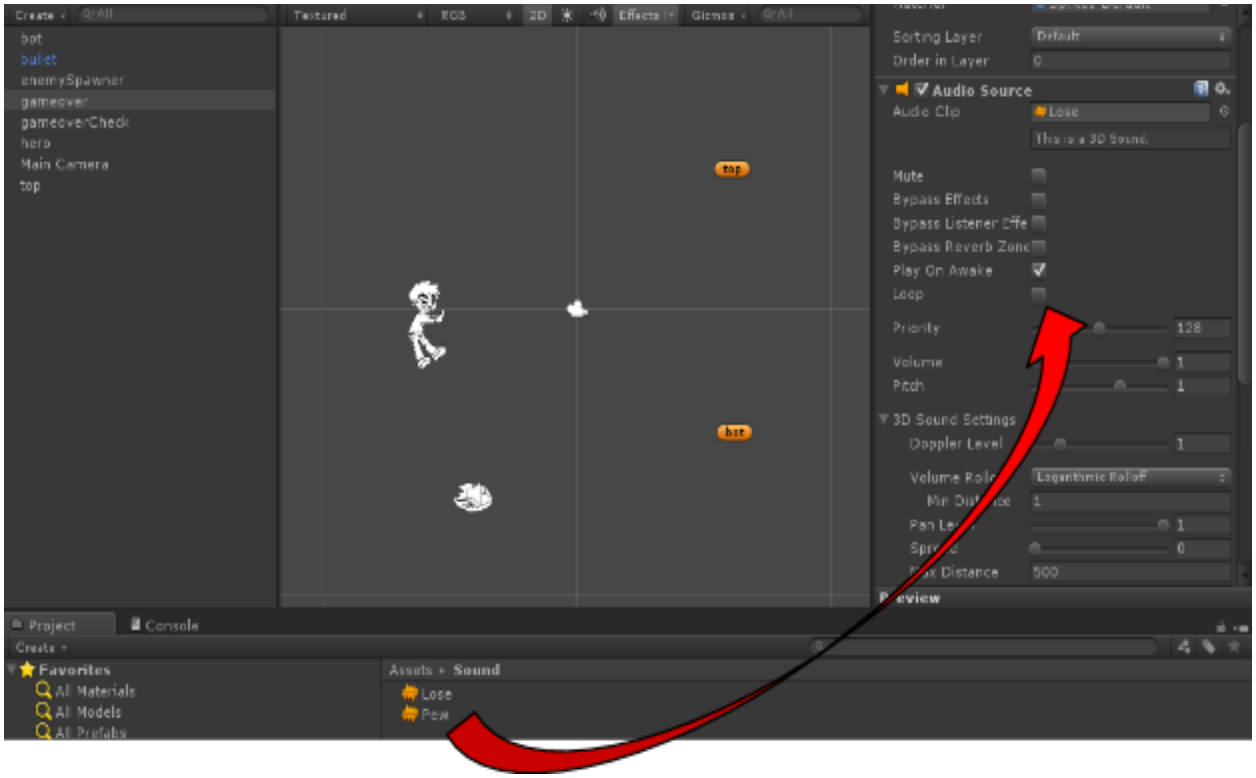
### Video Clip

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

### Reference

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

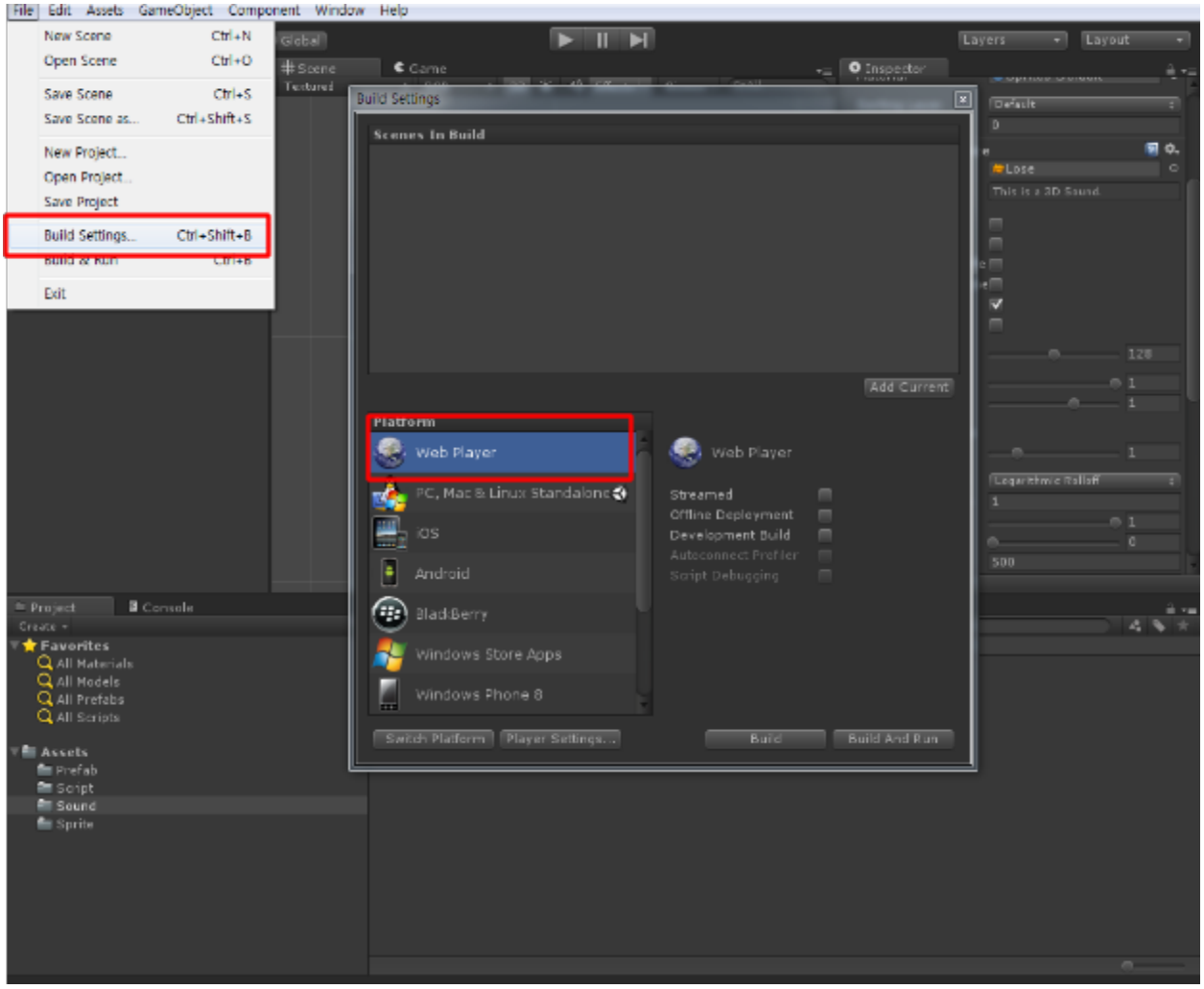
## Game Over Sound



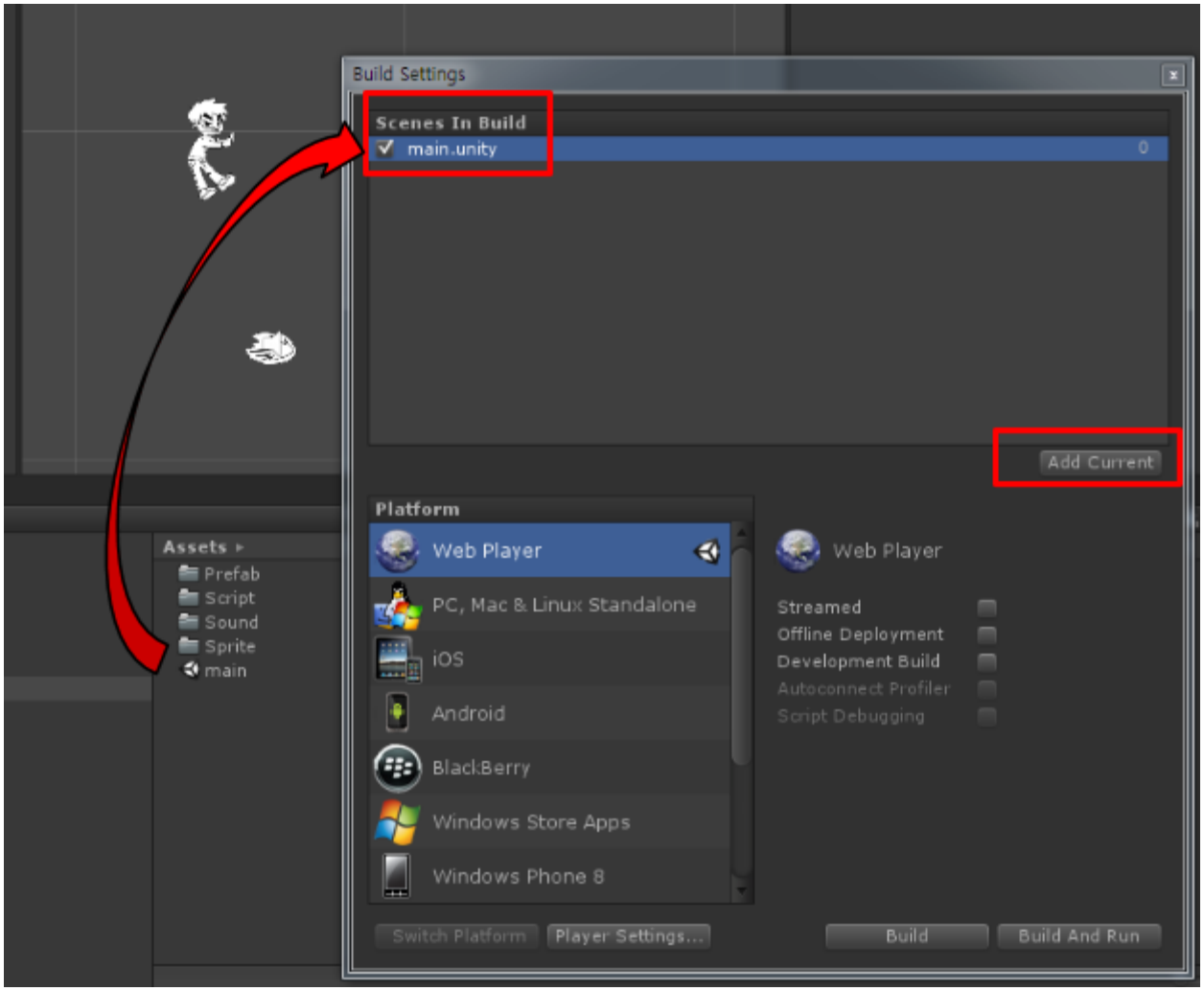

Similar as shooting, apply Lose sound on gameover object.

## ◆ Release

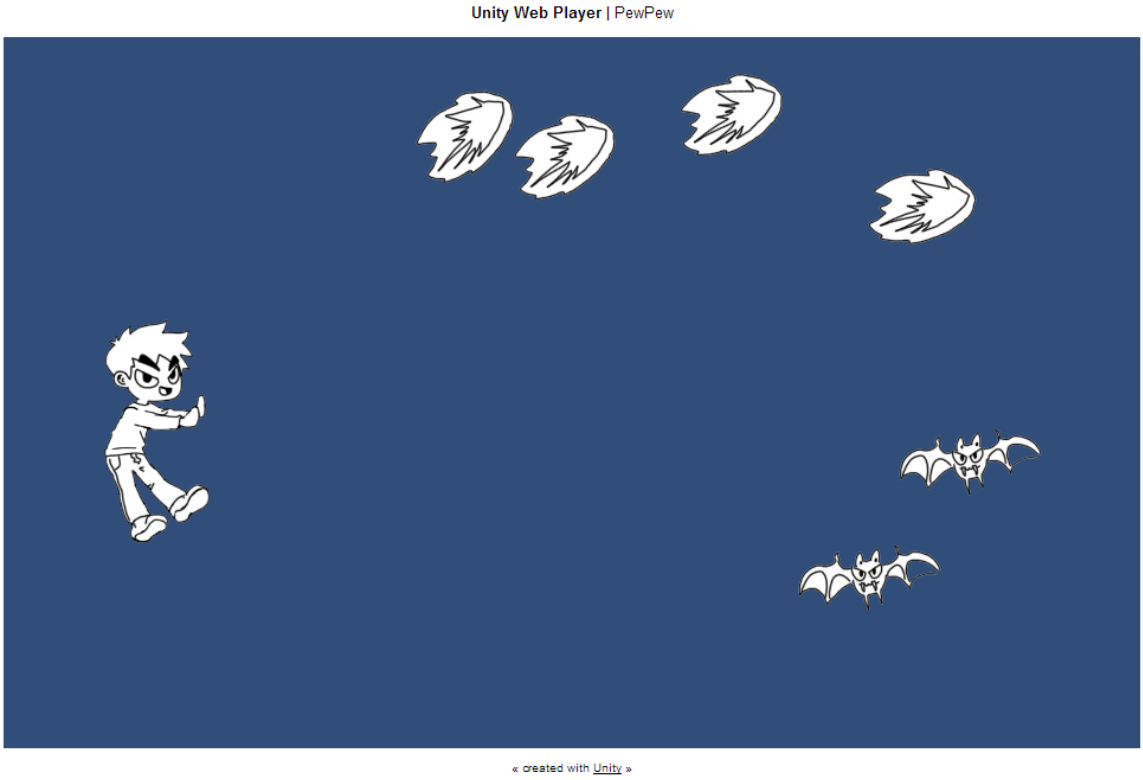
This is the completion of a game and finally, let’s release the game.




A pop-up window as above will appear when you click File -> Build Settings. It can be released in various platforms. In this tutorial, we are going to build it as Web.




Include the Scene to make the game runs properly. If you are currently working on the Scene, click Add Current button, or if you want to add other Scenes then Drag & Drop the Scene you want to include.




If the Build is succeed, then you can see the game working on browser.

## Reference

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