|  |  |
| --- | --- |
| ITB logo portrait B&W | INSTITUTE OF TECHNOLOGY BLANCHARDSTOWN  A Taster of Computing  [[VERSION – Unity 2D – C# language]] |

Gravity Guy 2D (2015) - a little computer game...

Part 2 – the story continues …



Welcome to “Gravity Guy”. In this multimedia programming exercise you will create a little 2D computer game.

CONTENTS

1 Recap, and aims of this part of the tutorial 3

2 Play a sound when a piece of food is eaten 4

3 Respawn the hero character when it falls down too low … 8

# Recap, and aims of this part of the tutorial

## Features to date

In the first part of this tutorial you implemented the following:

* Hero guy controlled by the user
* Fixed blue platforms
  + Acting as solid ‘Ground’ which can be landed on / jumped from
* Moving red platform
  + Acting as solid ‘Ground’ which can be landed on / jumped from
  + Moving at a customisable speed, between customisable max/min Y values
* A piece of cheese, tagged ‘Food’
  + Which when hit adds to Player’s score
  + And also removes itself from the scene
* An integer score
  + Stored in the player
  + And displayed using a print() statement to the Console status panel

## Skills and knowledge learned

In the first part of this tutorial you learned the following:

* The roles of the different window panels in the Unity editor
  + Project
  + Inspector
  + Hierarchy and Scene
  + Game
* How to:
  + play / stop a game
  + add instances of a prefab into the current scene
  + add a copy of an image sprite to the scene, the copying becoming a gameObject
  + change the properties of a selected gameObject or Project asset
  + add a Box Collider 2D to a gameObject
  + create and add a new string ‘tag’ to a game object
  + activate collision ‘trigger’ event messages for collisions between colliders
  + create a new C# script and add an instance as a component to a gameObject
  + create a prefab, populated with the components and properties of a gameObject in the scene

## New features / skills to be learned in this part of the tutorial

In this part of the tutorial you will add the following features to our game:

* Play a sound each time we add to the score
* Respawn the user when they fall too far down the screen
* Separate the score ‘view’ display to the user, from the observed score property inside the Player object
* Add killer ‘spike’ gameObjects to the scene
  + And have these make the player lose lives and respawn each time they are hit
  + This will involve adding a ‘lives’ property to our Player, and adding corresponding UI display of this property
* Add a “Game Over” scene to our game
  + And have the user see this scene when they lose their last life

# Play a sound when a piece of food is eaten

**The 3 audio object types**

Unity has 3 special sounds related types of objects:

* **AudioClip** – a sound file, such as our ‘yum.mp3’
* **AudioSource** – a component of a game object, allowing a gameObject to play a sound clip file

- this can be empty in a gameObject

or it can be populated with an AudioClip

* **AudioListener** – this is like an electronic ‘ear’ – usually we just work with the default setup, which is that when a new Scene is created in Unity, the Main Camera automatically has an AudioListener

**Other media assets – import them just the same way**

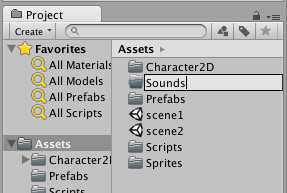
The methods shown in this section can be used to import 2D images, 3D models, audio files, text files, video files, C# scripts etc.

So know you know how to get NEW media/scripting assets into your Unity project …

## Add a sound clip file to a new Project folder “sounds”

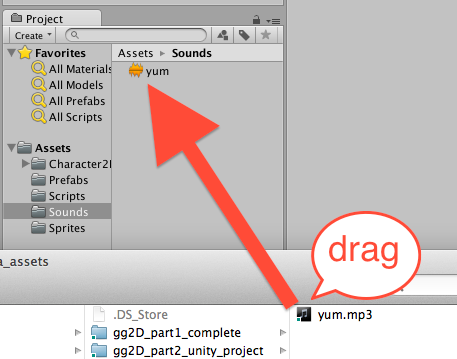
Create a folder “sounds” in the Project panel:

* In the Project panel select the Assets folder
* From the ‘Create’ dropdown menu choose ‘Folder’
* Rename the new folder ‘Sounds’

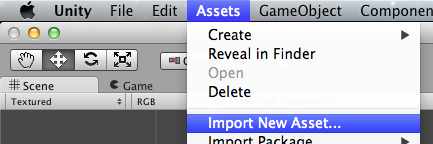


Add a copy of file sounds ‘yum.mp3’ into this new Sounds folder. You can either use the drag-and-drop method, or the Assets| Import menu approach …

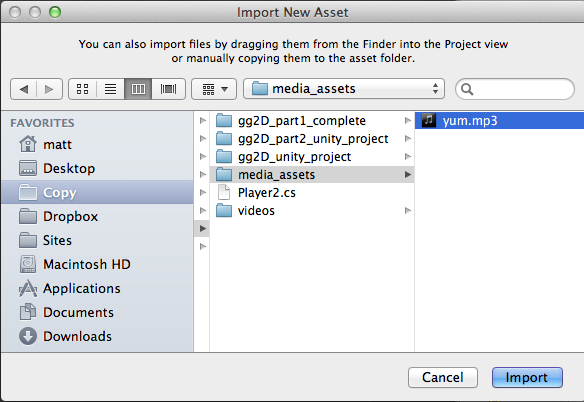
* **Drag-and-drop method:**
  + Ensure your **Sounds** folder is selected in the **Project** panel
  + Drag from an OS file window the file into the **Sounds** folder area (right hand side of **Project** panel)
  + You should now see the ‘yum’ sound clip file (with audio wave icon) in the **Project** panel **Sounds** folder



* **Assets MENU method:**
  + Ensure your Sounds folder is selected
  + Choose menu: Assets | Import New Asset …



* + Use the file dialog to navigate to and import the sounds file



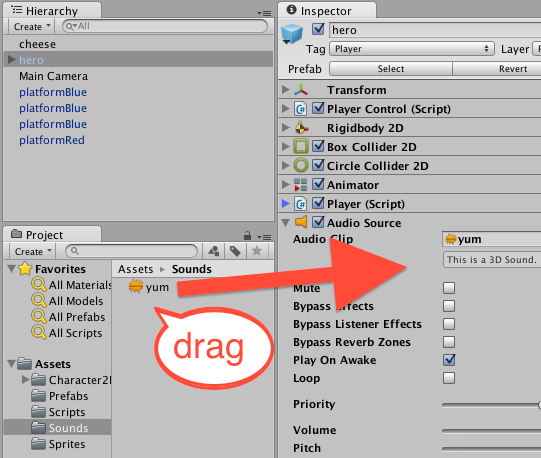
* + You should now see the ‘yum’ sound file (with audio wave icon) in the **Project** panel **Sounds** folder

## Add an AudioSource component, containing our ‘yum’ sound, to the ‘hero’ gameObject

Any Unity gameObject that wants to play a sound must have an ‘AudioSource’ component. The AudioSource component can be pre-loaded with a sound clip file (such as ‘yum’ …), or at run time a particular sound clip file can be loaded into it and then played.

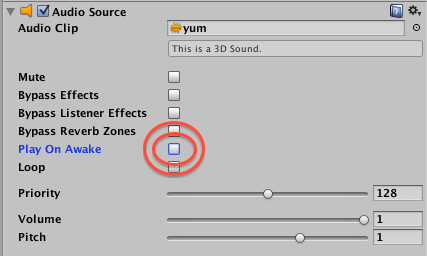
Unity makes things easy – if we drag an AudioClip sound clip file from the Project panel into a gameObject it will automatically add an AudioSource component to the gameObject, and populate it with the sound clip file that was dragged:

* In the **Project** panel select the **Sounds** folder
* In the **Hierarchy** select the **hero** gameObject
* Drag the **yum** audioClip sound file into the Inspector (which should be showing us the components and properties of gameObject **hero)**
* You should now see that the **hero** gameObject contains an **AudioSource** component, and its **AudioClip** property is our **yum** sound clip file



Finally, we do NOT want this sound to play at the beginning of the scene when the **hero** gameObject is instantiated, so we need to DESELET the “Play On Awake” option in the AudioSource component:

* In the **Hierarchy** select the **hero** gameObject
* In the Inspector un-tick the “Play on Awake” property of the AudioSource component



## Add to our Player script class, so that it plays a sound when “Food” items are collided with

Start editing the Player script class in Monodevelop

* In the **Project** panel select the **Scripts** folder
* Double click the **Player** script class file to load it into the **Monodevelop** editor

Recap – your Player script class code should look as follows:

using UnityEngine;

using System.Collections;

public class Player : MonoBehaviour {

private int score = 0;

void Update(){

string scoreMessage = "Score = " + score;

print(scoreMessage);

}

void OnTriggerEnter2D(Collider2D hit){

if(hit.CompareTag("Food")){

score++;

Destroy (hit.gameObject);

}

}

}

AFTER the statement where we destroy the object hit which was tagged with tag “Food”, we are going to add another statement, to tell the gameObject (**hero**) to send a “Play()” message to its AudioSource component (i.e. play the **yum** AudioClip in its AudioSource component). So each time we hit something tagged “Food” we play that sound:

*using UnityEngine;*

*using System.Collections;*

*public class Player : MonoBehaviour {*

*private int score = 0;*

*void Update(){*

*string scoreMessage = "Score = " + score;*

*print(scoreMessage);*

*}*

*void OnTriggerEnter2D(Collider2D hit){*

*if(hit.CompareTag("Food")){*

*score++;*

*Destroy (hit.gameObject);*

audio.Play();

*}*

*}*

*}*

## Playtest your game

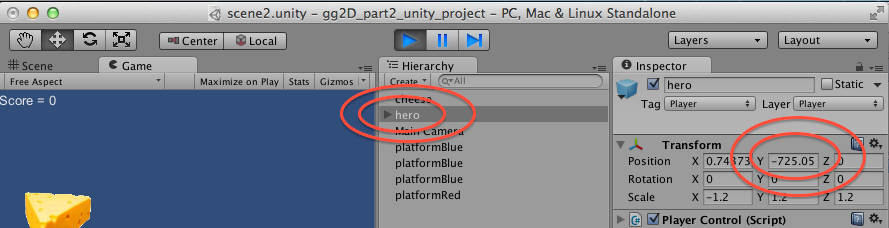
You should now hear the **yum** sound when you ‘eat’ a piece of cheese

# Respawn the hero character when it falls down too low …

## Add to our Player script class, to respawn when Y value too low

If our hero player character falls off a platform, it just keeps on falling …

Let’s see the numbers for this – deselect the Game panel option “Maximise on play”, select **hero** in the Hierarchy (so we can see that gameObject’s properties in the Inspector), and run the game. When your character walks off a platform you can see its Y-position keep increasing as a larger and larger negative (downwards) value …



The Update() message is sent to all scene gameObjects EVERY FRAME. So any code in method Update() will be executed every frame. Let’s add to our Update() method some logic to return our **hero** character gameObject back to its starting position (0,5,0) if its Y-position is less than -5 (below where all platforms should be).

* In the **Project** panel select the **Scripts** folder
* Double click the **Player** script class file to load it into the **Monodevelop** editor
* Add a new float property **deathY** set to -15
  + This is the Y-position below which the **hero** character will be moved back to (0,5,0)
* Add the new C# code to method Update(), and its related method MoveToStartPosition():

*using UnityEngine;*

*using System.Collections;*

*public class Player : MonoBehaviour {*

*private int score = 0;*

private float deathY = -15;

*void Update(){*

*string scoreMessage = "Score = " + score;*

*print(scoreMessage);*

float y = transform.position.y;

if(y < deathY){

MoveToStartPosition();

}

*}*

private void MoveToStartPosition(){

Vector3 startPosition = new Vector3(0,5,0);

transform.position = startPosition;

}

*... the rest of the code is unchanged …*

## Playtest your game

When your **hero** character falls off a platform, after a short time it should reappear back at (0,5,0) – its start position.

**Note – watch our for speed / angle velocity …**

Our current code moves the **hero** back to the start position, but he will still be falling, and moving left/right with the same velocity as he was when off the platform.

A more advanced re-spawning of a character would also require us to remove any downward speed, or sideways motion, when the character is moved back to its starting position.

But we’ll leave that ‘tweak’ for another tutorial …

**Congratulations**

**You have now created part 2 of the tutorial !**