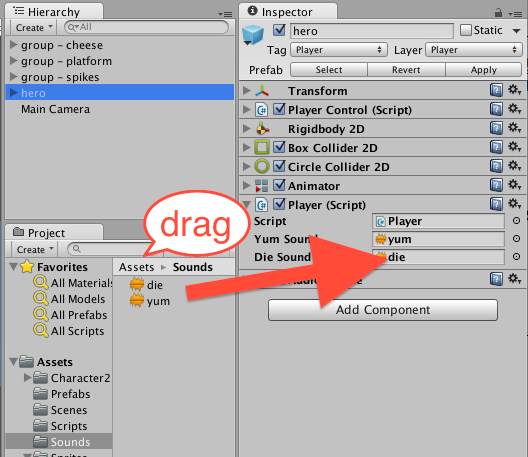
|  |  |
| --- | --- |
| ITB logo portrait B&W | INSTITUTE OF TECHNOLOGY BLANCHARDSTOWN |

A Taster of Computing

[[VERSION – Unity 2D – C# language]]

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

Part 6 – 800x600 size / playing different sounds / parenting



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

CONTENTS

1 Aims of this part of the tutorial 2

2 Tidy up hierarchy – child items to empty container objects 3

3 Set the ‘resolution’ of the application build to 800 x 600 pixels 5

4 Different sounds for different collisions 8

5 FULL LISTINGS 11

# Aims of this part of the tutorial

## 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:

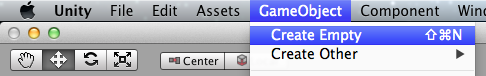
* Tidy up hierarchy by parenting child items to empty container objects
* Set the ‘resolution’ of the application build to 800 x 600 pixels
* Different sounds for different collisions

# Tidy up hierarchy – child items to empty container objects

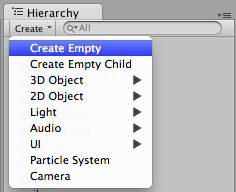
## Create ‘empty’ game object in Hierarchy, named ‘spikes – group’

Create a new empty gameObject in the Hiearchy:

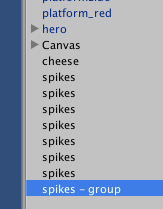
* Select application menu bar: GameObject | Create Empty



* Or from the **Hierarchy** choose menu: Create | Create Empty

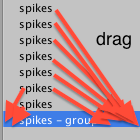


* Rename the new empty gameObject in the Hierarchy as ‘spikes - group

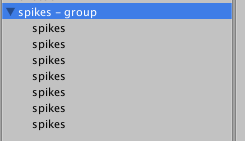


## In Hierarchy, drag all ‘spikes’ gameObjects into the new group object

In the **Hierarchy** drag to ‘child’ each spikes gameObject into the ‘spikes – group’ object:



A **spikes** gameObjects should now be children of the group gameObject:



## Repeat with ‘platforms - group’

In the **Hierarchy** repeat this procedure to child all the platform gameObjects into a new empty gameObject:



# Set the ‘resolution’ of the application build to 800 x 600 pixels

**WHAT are we making?**

Unity projects can be ‘built’ into different kinds of computer program:

* Stand along PC, Apple, Linux desktop applications
* Web-player games
* Games console games
* Mobile phone ‘apps’ for iOS, Android, Windows phone etc.

Once we know the destination ‘device’ that our game will be ‘deployed’ onto, we can set the **Game** panel dimensions to the device **width** and **height**

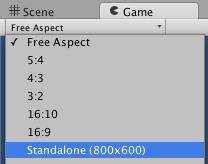
Even if we are creating a ‘proof of concept’ it can be very useful to FIX THE RESOLUTION of our game, so we can design the screen layout, and find / create 2D graphic assets of the correct size

As a rule of thumb, If in doubt, build to 800 x 600 pixels (width x height)

## If listed, choose the desired Game resolution from Game panel drop down

If the build settings have been set up correctly, or they are inherited since we started our game on a pre-ade Unity project (like Gravity Guy), then the **Game panel** resolution can be set very simply:

* Choose the screen resolution (800 x 600) from the drop down menu at the top of the **Game panel**



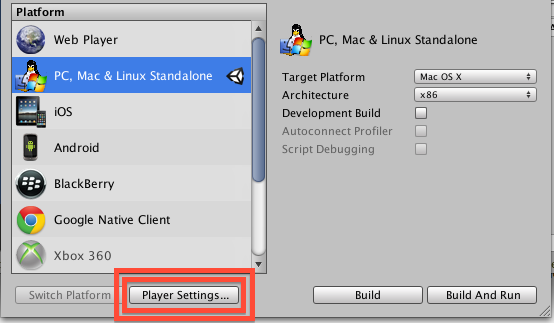
Once chosen, you’ll see that the **Game panel** now always scales its display with the correct width/height proportion:



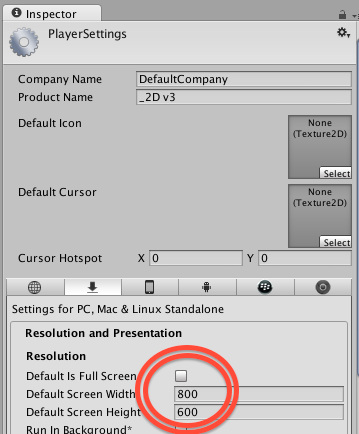
## If desired resolution not listed, set it via the Build or Player settings option

To change/set a resolution to have added to the Game panel menu, do the following:

* Open the Build settings dialog, by choosing menu: **File | Build Settings …**
* Click the button ‘**Player Settings …**”



* Alternatively you can also get the Player settings in the Inspector by choosing application menu: **Edit | Project Settings | Player**
* You can set the default screen width and height in the **Inspector**, which is now showing the Player Settings properties:



# Different sounds for different collisions

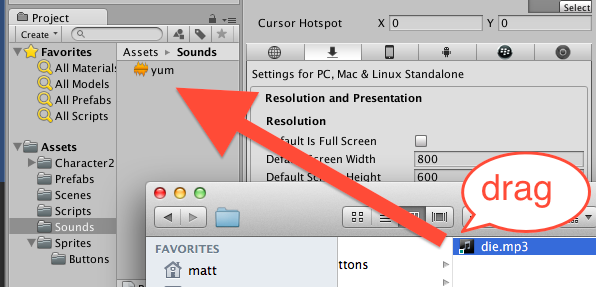
## Add new audio sound clip file to your project

Most games play one of several different sounds, depending on what even has occurred. We are going to enhance our game to do the following:

* If “Food” hit, play the “yum” audio sound clip
* If “Spikes” hit, play the “die” audio sound clip

First, add the “die” audio sound clip to our project:

* Drag the “zombieDie.mp3” file into the **Project panel** folder “Sounds”



## Change our “Player” script class to play different sounds when appropriate

Once we are dealing with more than one audio clip, our script needs a variable for each sound clip file.

So first add public variables for the 2 sound clips to our Player script class:

* In the **Project** panel select the **Scripts** folder
* Double click the **Player** script class file to load it into the **Monodevelop** editor
* Edit the code as follows:
  + Add variables “yumSound” and “dieSound”

*using UnityEngine;*

*using System.Collections;*

*public class Player : MonoBehaviour*

*{*

**public AudioClip yumSound;**

**public AudioClip dieSound;**

*... as before ...*

**One reason to make a variable ‘public’ – can be set in the Inspector**

Once an instance of a script class has been added as a component of a gameObject, the public variables for that instance can be set in the **Inspector**

* Numeric variables can just be typed in
* Media objects (images, sounds, 3D models, materials, prefabs) can be drag-and-dropped
* Drag-and-drop can also be used when variables are gameObjects or script instances as well

This makes linking different parts of our game, and our media files, very straightforward and natural – once you’ve got the hand of it …

Note – rather annoyingly, in the Inspector the first letter of variables are capitalised ….

Next we are going to change our code. Rather than sending a “**Play**()” message to the “audio” component, we need to send a “**PlayOneShot**” messages AND pass as a parameter the sound clip variable, so the Audio Source component knows which sound clip to play:

Change method **OnTriggerEnter2D()** so that it looks as follows:

*void OnTriggerEnter2D(Collider2D hit){*

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

*score++;*

*playerDisplay.UpdateScoreText(score);*

*Destroy (hit.gameObject);*

**audio.PlayOneShot(yumSound);**

*}*

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

*LoseLife();*

*}*

*}*

Now add a final statement to method **LoseLife()** so that our die sound is player:

*private void LoseLife(){*

*lives--;*

*if(lives < 0){*

*Application.LoadLevel("scene1\_GameOver");*

*}*

*playerDisplay.UpdateLivesText(lives);*

*MoveToStartPosition();*

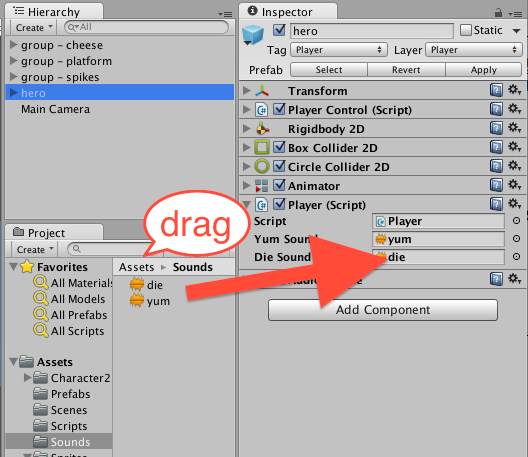
**audio.PlayOneShot(dieSound);**

*}*

## Associate the audio sound clip files with our yumSound and dieSound variables

Now we need to link the sound clip files in the **Project panel** with the yumSound and dieSound public variables in the instance of our Player script in our ‘hero’ guy in scene1:

* Ensue you have SAVED your Player script changes
  + And fix any erorrs !
* Ensure you are editing scene “**scene1”**
* Select the ‘hero’ gameObject in the **Hierarchy**
* Select the “Sounds” folder in the **Project panel**
* You can now drag-and-drop the “yum” sound from the Sounds folder over the “Yum Sound” variable in the Player script component in the Inspector
  + And do the same with the “die” sound clip for variable “Die Sound”



## Playtest your game

Start on the Welcome scene and click the button to play the game.

When you die you should hear a different sound than when you pick up a piece of cheese!

**Congratulations**

**You have now created part 6 of the tutorial !**

# FULL LISTINGS

## Player

using UnityEngine;

using System.Collections;

public class Player : MonoBehaviour {

public AudioClip yumSound;

public AudioClip dieSound;

private PlayerDisplay playerDisplay;

private int lives = 3;

private int score = 0;

private float deathY = -15;

//-----------------------------

void Start(){

playerDisplay = GetComponent<PlayerDisplay>();

playerDisplay.UpdateScoreText(score);

playerDisplay.UpdateLivesText(lives);

}

//-----------------------------

void Update(){

float y = transform.position.y;

if(y < deathY){

LoseLife();

}

}

//-----------------------------

private void LoseLife(){

lives--;

if(lives < 0){

Application.LoadLevel("scene1\_GameOver");

}

playerDisplay.UpdateLivesText(lives);

MoveToStartPosition();

audio.PlayOneShot(dieSound);

}

//-----------------------------

private void MoveToStartPosition(){

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

transform.position = startPosition;

}

//-----------------------------

void OnTriggerEnter2D(Collider2D hit){

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

score++;

playerDisplay.UpdateScoreText(score);

Destroy (hit.gameObject);

audio.PlayOneShot(yumSound);

}

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

LoseLife();

}

}

}