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

Gravity Guy 2D (2014) - a little computer game...

Part 6 – and there’s even more …

CONTENTS

1 TOPIC 2

2 Simple timers 3

# TOPIC

## ABOUT

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

TEMPORARY INVULUNERABILITY – only die when its time to die ….

# Simple timers

## In many games there is some form of countdown or countup timer …

Examples of timers in games include:

* Lose life if level not complete before countdown timer gets to zero
* Pick up a shield of invulnerability, that lasts for 2 seconds
* Drop a bomb that will explode in 5 seconds
* Survive as long as possible during a bonus mini-level, and for each 10 seconds you survive you get an extra life / more money etc.
* And so on…

## The CountdownTimer.cs script

Add an instance of the provided **CountdownTimer.cs** script to the Main Camera:

* Select the **Main Camera** in the **Hierarchy**
* Drag a copy of the **CountdownTime** script to the **Inspector** (or directly onto the **Main Camera**)

NOTE: just as with ButtonActions, if we need to have a runtime object of a class, and there isn’t a natural home (such as in the player’s character, or in a pickup object like a key), then the Main Camera is one handly place to add instances of classes to keep things tidy.

## The GameManager class – overview

We shall create a Game Manager class to handle the following responsibilities:

* Reset the timer at the beginning of the level
* Load the Game Over screen, if the timer goes below zero – checking every frame in method Update()
* Update the remaining seconds display in a UI Text object, again every frame from method Update()

## The GameManager class – reset timer when scene begins

Create a new C# script named **GameManager**. When the scene starts, we need to get a reference to our CountdownTimer object (also in GameManager), and reset it to the number of seconds for this level (let’s give the player 20 seconds for now, but if we make this public then we can change it later in the **Inspector** if we want to). We need a private variable to be a reference to our **CountdownTimer** object:

public int timeForLevel = 20;

private CountdownTimer countdownTimer;

void Start() {

countdownTimer = GetComponent<CountdownTimer>();

countdownTimer.ResetTimer(timeForLevel);

}

## The GameManager class – check each frame for game over condition (timer < 0)

Each frame we test the game over conditions, in this case if the seconds remaining in our timer go below zero. If this is the case it’s time to load the Game Over scene. Since we are going to also be wanting to update the timer display every frame, we’ll get the seconds left, and then call our game over test (in its own method) passing the remaining seconds, doing this every frame from Update():

void Update()

{

int secondsLeft = countdownTimer.GetSecondsRemaining();

CheckGameOver(secondsLeft);

}

private void CheckGameOver(int secondsLeft)

{

if(secondsLeft < 0)

{

int gameOverLevelIndex = 2;

Application.LoadLevel("scene1\_GameOver");

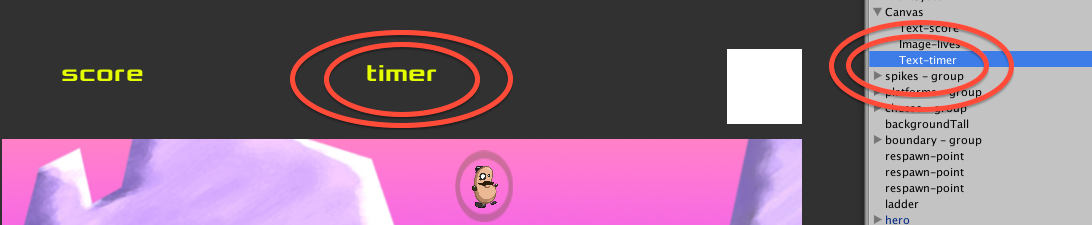
}

}

## The GameManager class – update display of seconds left to user (every frame)

Every frame let’s update the remaining seconds display in a UI Text object.

First, let’s create a new UI Text object to display the seconds remaining. The simplest way is to duplicate the **Text-score** UI text object, rename the copy **Text-timer**, position it to be at TOP – CENTER, and change its default text to something meaningful like ‘timer’ so we know which UI Text it is in the **Game** panel:



Next our **GameManager** class needs a public Text object reference variable, into which we can drag our Text-timer UI object:

using UnityEngine;

using System.Collections;

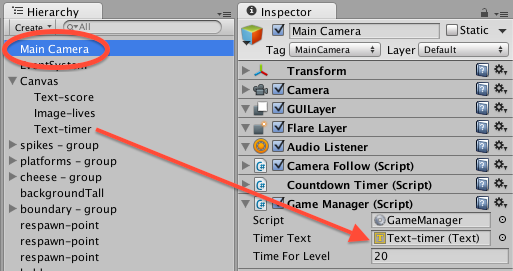
using UnityEngine.UI;

public class GameManager : MonoBehaviour

{

// public

public Text timerText;



When the scene starts, we need to get a reference to our CountdownTimer object (also in GameManager), and reset it to the number of seconds for this level (let’s give the player 20 seconds for now, but if we make this public then we can change it later in the **Inspector** if we want to). We need a private variable to be a reference to our **CountdownTimer** object:

public int timeForLevel = 20;

private CountdownTimer countdownTimer;

private void Start()

{

countdownTimer = GetComponent<CountdownTimer>();

countdownTimer.ResetTimer(timeForLevel);

print ("Timer started");

}

Each frame, from method Update() we wish to call method UpdateTimerDisplay():

private void Update()

{

int secondsLeft = myTimer.GetSecondsRemaining();

CheckGameOver(secondsLeft);

UpdateTimerDisplay(secondsLeft);

}

private void UpdateTimerDisplay(int secondsLeft)

{

string timerMessage = "Time left = " + secondsLeft;

timerText.text = timerMessage;

}

Write the following for your GameGUI script, so the user can see the seconds remaining for the countdown timer component:

## The GameManager

Our GAME GUI-HUD needs to read the number of seconds left, to display to the user

Sometimes there are responsibilities that are separate to the Player or GameGUI, for this we can place such logic into a special ‘GameManager’ script, (or LevelManager).

Create a new C# script named GameManager, containing the following code, and add a copy of this to the Main Camera:

**using UnityEngine;**

**using System.Collections;**

**public class GameManager : MonoBehaviour**

**{**

**public int timeForLevel = 20;**

**private CountdownTimer myTimer;**

**private void Start()**

**{**

**myTimer = GetComponent<CountdownTimer>();**

**myTimer.ResetTimer(timeForLevel);**

**print ("Timer started");**

**}**

**private void Update()**

**{**

**CheckGameOver();**

**}**

**// GAME OVER if seconds < 0 !!!!!**

**private void CheckGameOver()  
 {**

**int secondsLeft = myTimer.GetSecondsRemaining();**

**if(secondsLeft < 0)**

**{**

**int gameOverLevelIndex = 2;**

**Application.LoadLevel(gameOverLevelIndex);**

**}**

**}**

**}**