|  |  |
| --- | --- |
| 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 4 – lives & killer spikes …



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 Add lives property to player, and its display to the user 2

3 Add killer ‘spike’ objects to the scene 8

4 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:

* 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 lives property to player, and its display to the user

## Add a new property ‘lives’ to our Player script class

Let’s start the player off with 3 lives. We need to add a property ‘lives’ to our Player script class, and call the method to update its display when the scene starts.

Add a ‘lives’ integer property to script class Player:

* 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 a new private integer property ‘lives’ initialised to 3

using UnityEngine;

using System.Collections;

public class Player : MonoBehaviour {

private PlayerDisplay playerDisplay;

private int lives = 3;

... as before  

In method Start() we need to call the method to update the lives display:

* Add a statement to the end of method Start():

*void Start(){*

*playerDisplay = GetComponent<PlayerDisplay>();*

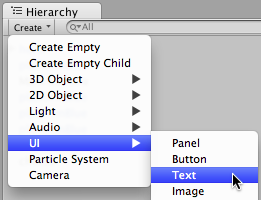
*playerDisplay.UpdateScoreText(score);*

playerDisplay.UpdateLivesText(lives);

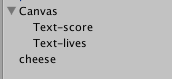
*}*

## Create a new UI Text object on screen to display lives value to user

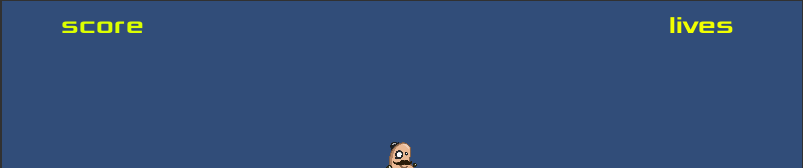
Create a new UI Text object. In the Hierarchy panel choose first select the **Canvas** object, then choose menu: Create | UI | Text, and type “lives” in property Text of component Text (Script) (to know which is which):



Renamed your new Text object **Text-score**. Your hierarchy should now look as follows:



Since we have 2 UI texts, lets arrange them in the two top corners:



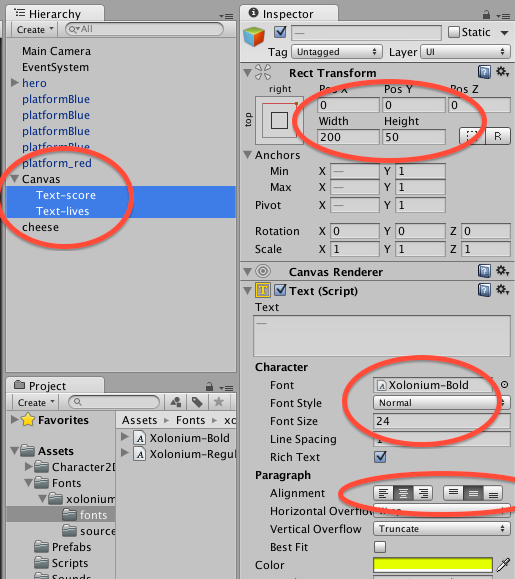
First let’s select BOTH text objects and set all the properties that will be the same.:

1. Using the SHIFT key select BOTH Text-score and Text-lives in the Hierarchy
2. In the Inspector ensure all the following are set for the **Rect Transform** component:

* Font is Xolonium-Bold and size 24
* Pos X = 0, Pos Y = 0, Pos Z = 0;
* Width = 200, Height = 50

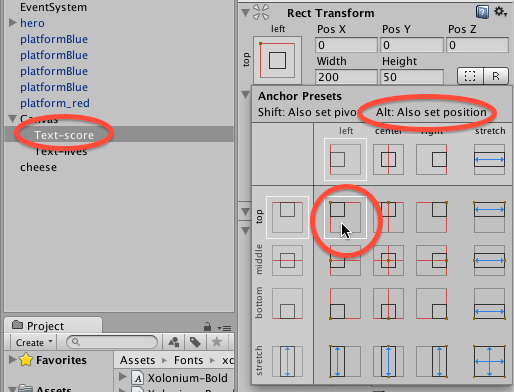
1. In the Inspector ensure all the following are set for the Text (Script) component:

* Font is Xolonium-Bold and size 24
* Paragraph Alignment is centered both horizontally and vertically
* Color is yellowish

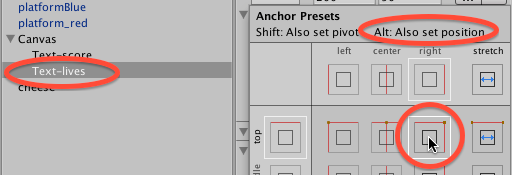


Now we need to position the 2 text objects into each corner ….

In the **Hierarchy** select **Text-score**. In the Inspector open up the preset window for all the positions of anchors and pivot points, and set the position (hold down **ALT**) and click TOP LEFT:



In the **Hierarchy** select **Text-lives**. In the Inspector open up the preset window for all the positions of anchors and pivot points, and set the position (hold down **ALT**) and click TOP RIGHT:



Try this too (to get to know UI layout techniques):

* Play around with arranging these text objects in different parts of the screen, and their alignment and stretching (add more text – the quick brown fox jumps over the lazy dog / Lipsum etc.)
* Play around with the width / height / font size – and then play with the Paragraph Horizontal / Vertical OVERFLOW …

## Add code to class PlayerDisplay to update our new UI ‘lives’ text object

We need to add a new **public** variable that will be a reference to the UI Text object **Text-lives**:

public Text livesText;

We need a new method, that when passed a new integer lives value, will build a string message and then update the text property of the UI Text object **Text-lives**:

public void UpdateLivesText(int newLives){

string livesMessage = "Lives = " + newLives;

livesText.text = lives;

}

Full listing of PlayerDisplay:

using UnityEngine;

using System.Collections;

using UnityEngine.UI;

public class PlayerDisplay : MonoBehaviour {

public Text scoreText;

public Text livesText;

public void UpdateScoreText(int newScore){

string scoreMessage = "Score = " + newScore;

scoreText.text = scoreMessage;

}

public void UpdateLivesText(int newLives){

string livesMessage = "Lives = " + newLives;

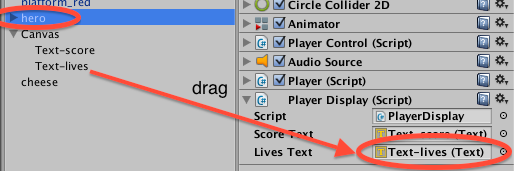
livesText.text = livesMessage;

}

}

## Link UI Text-lives component to the public variable in hero

Let’s set up the reference in our PlayerDisplay scripted component in hero to our UI Text object **Text-lives**. Select hero in the Hierarchy, and drag into its PlayerDisplay (Script) component property livesText the UI Text object **Text-lives**



# Add killer ‘spike’ objects to the scene

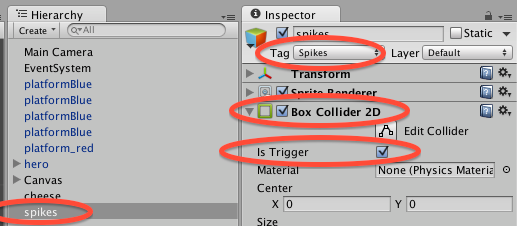
## Create a ‘spikes’ prefab

Let’s use the ‘spikes’ image to create a tagged prefab that we can duplicated underneath the platforms.

First drag the spikes image from folder Project / Sprites onto the scene below one of the platforms. You should now see a new object on the Scene, and in the hierarchy named ‘spikes.

Create a new string tag “Spikes”, and assign it to gameObject **spikes** in the Hierarchy.

With **spikes** selectedin the Hierarchy, add a Physics2D | Collider2D as a component, and tick its “**Is Trigger**” option.



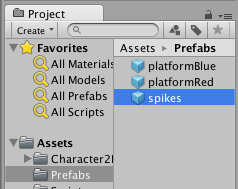
In folder **Project / Prefabs** create a new empty prefab named ‘spikes’. Drag from the Hierarchy the spikes object over your new empty prefab – it should turn blue and become ‘populated’ with a copy of all the components and properties of the gameObject.

## Add some copies of the ‘spikes’ prefab as gameObjects to the scene

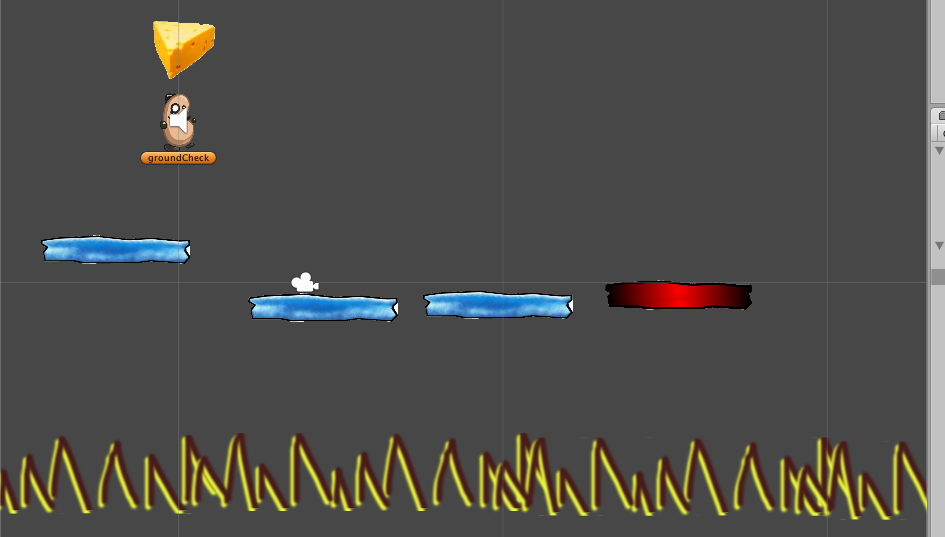
Rather than our player falling into some invisible minimum Y position, let’s actually display some ‘spikes’, whereby the player loses a life when they fall onto them.

Add some copies of the ‘spikes’ prefab as gameObjects to our scene, in a row BELOW the platforms:

* In the **Project** panel select the **Prefabs** folder
* Drag copies of the **spikes** prefab onto the scene, in a row below the platforms:



Your scene should now look something like the following:



­­­ ­­

## Edit Player code, to decrement lives when hit something tagged ‘spikes’

We now need to edit our Player script class, so that when it collides with something tagged ‘spikes’, we decrement the lives, and move it back to the start position. In fact since we now have 2 situations where a player dies and loses a life, let’s organise things into a **LoseLife()** method:

    private void LoseLife(){  
        lives--;  
        playerDisplay.UpdateLivesText(lives);  
        MoveToStartPosition();  
    }

There are 2 places where we will call this – when we hit something tagged ‘Spikes’ or when our Y-value becomes too low. Our test for Y-value is in Update():

void Update(){

float y = transform.position.y;

if(y < deathY){

LoseLife();

}

}

Our test for collision with something tagged ‘Spikes’ is in OnTriggerEnter2D():

*void OnTriggerEnter2D(Collider2D hit){*

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

*score++;*

*playerDisplay.UpdateScoreText(score);*

*Destroy (hit.gameObject);*

*audio.Play();*

*}*

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

LoseLife();

}

*}*

## Playtest your game

Close the Build Settings, and reload **scene1**, then run your game. Then keep falling off platforms and going low or hitting spikes to make the number of lives less than zero

**Congratulations**

**You have now created part 4 of the tutorial !**

# FULL LISTINGS

## Player

using UnityEngine;

using System.Collections;

public class Player : MonoBehaviour {

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--;

playerDisplay.UpdateLivesText(lives);

MoveToStartPosition();

}

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.Play();

}

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

LoseLife();

}

}

}

## PlayerDisplay

using UnityEngine;

using System.Collections;

using UnityEngine.UI;

public class PlayerDisplay : MonoBehaviour {

public Text scoreText;

public Text livesText;

public void UpdateScoreText(int newScore){

string scoreMessage = "Score = " + newScore;

scoreText.text = scoreMessage;

}

public void UpdateLivesText(int newLives){

string livesMessage = "Lives = " + newLives;

livesText.text = livesMessage;

}

}