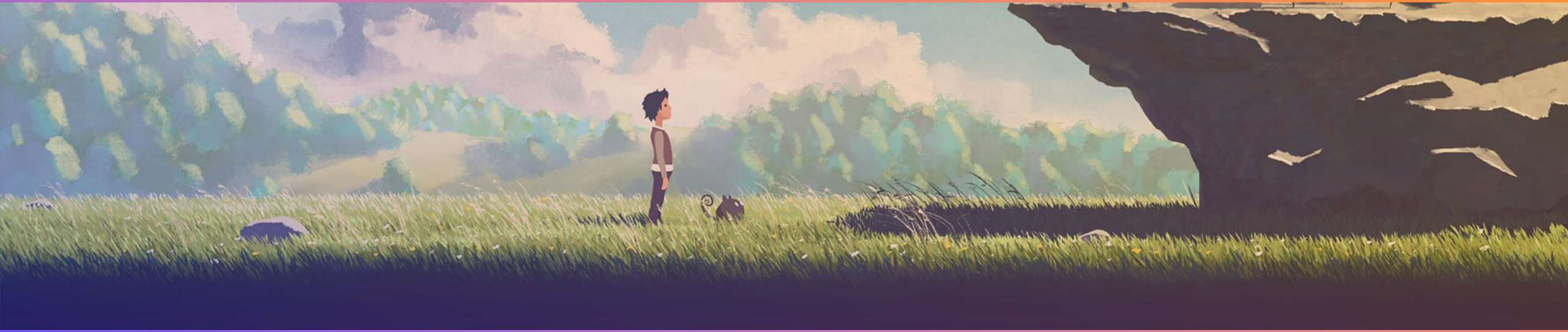
(Week 11)



การเขียนโปรแกรมคอมพิวเตอร์ขั้นสูงเพื่อควบคุมอุปกรณ์ ADVANCE COMPUTER PROGRAMMING

สอนโดย พงศธร เกียรติเจริญพร (มิว)

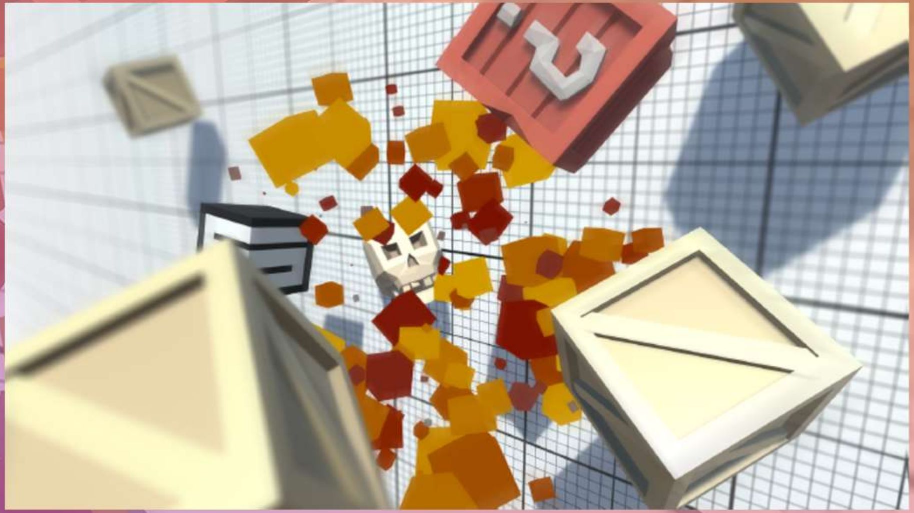
29/09/2021

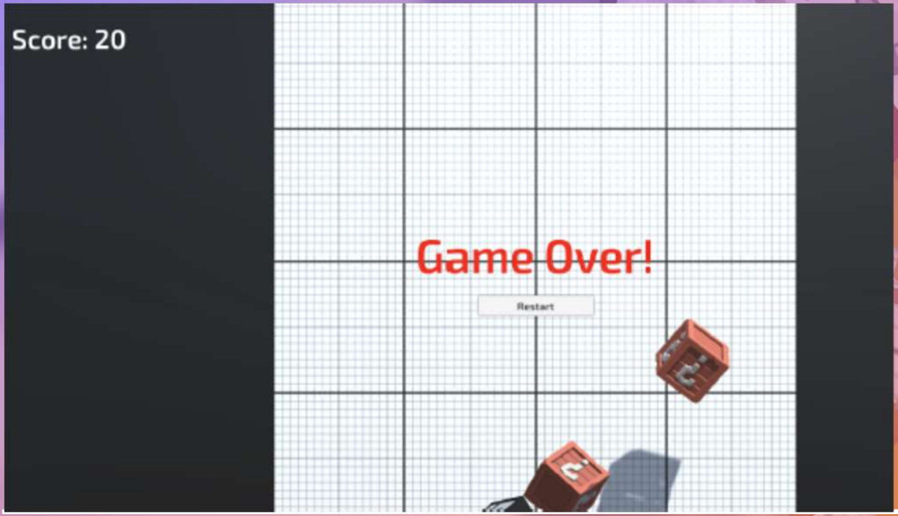
UNIT 5 – USER INTERFACE

(Quick Click Prototype)

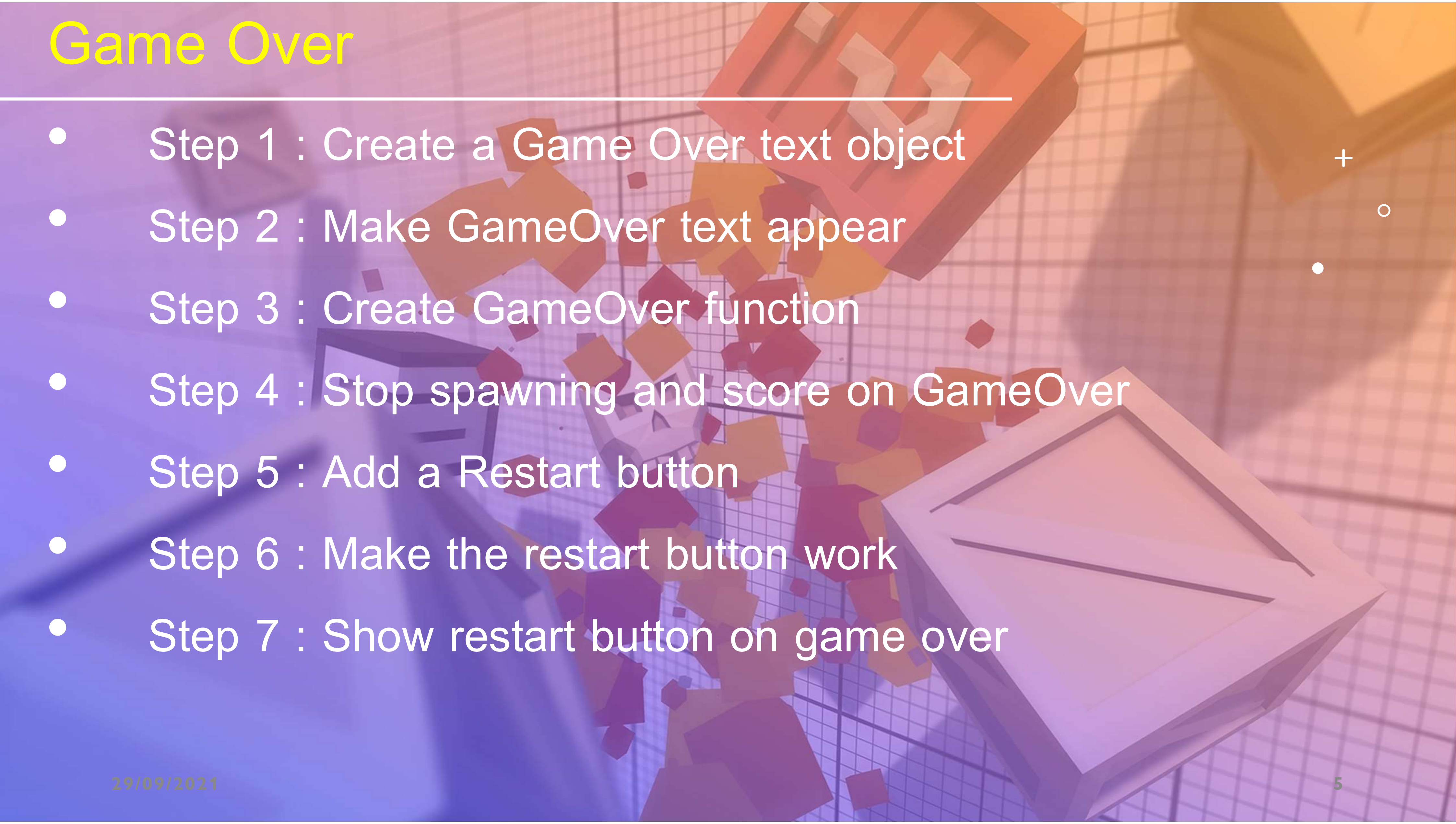
Unit 5 – User Interface

- Quick Click Prototype
  - Clicky Mouse
  - Keeping Score
  - Game Over
  - What’s the Difficulty?





Game Over – เจ้าน่ะ... ได้แพ้แล้ววว



Game Over

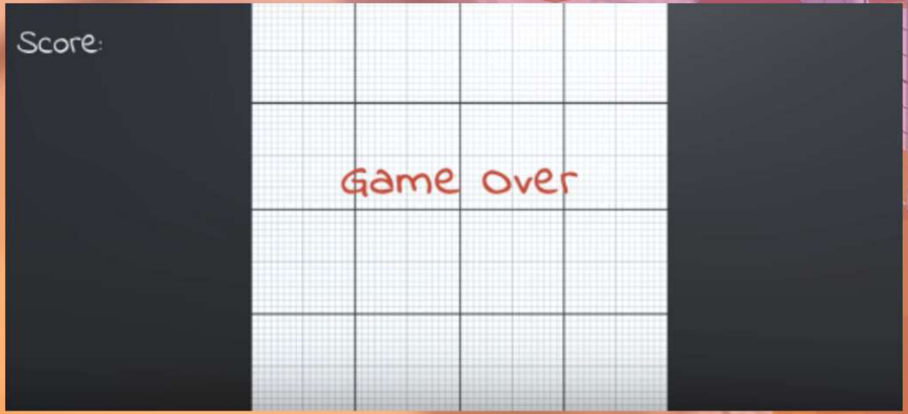
- Step 1 : Create a Game Over text object
- Step 2 : Make GameOver text appear
- Step 3 : Create GameOver function
- Step 4 : Stop spawning and score on GameOver
- Step 5 : Add a Restart button
- Step 6 : Make the restart button work
- Step 7 : Show restart button on game over

Game Over – Step 1 : Create a Game Over text object

If we want some "Game Over" text to appear when the game ends, the first thing we'll do is create and customize a new UI text element that says "Game Over".

1. Right-click on the Canvas, create a new UI > TextMeshPro - Text object, and rename it "Game Over Text"
2. In the inspector, edit its Text, Pos X, Pos Y, Font Asset, Size, Style, Color, and Alignment
3. Set the "Wrapping" setting to "Disabled"

Tip : The center of the screen is the best place for this Game Over message - it grabs the player's attention

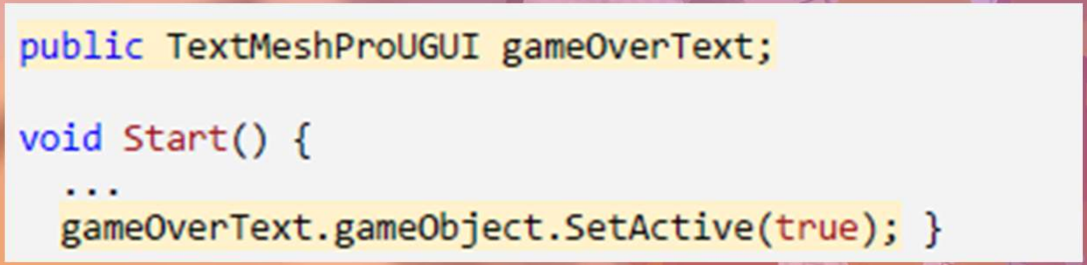


Game Over – Step 2 : Make GameOver text appear

We've got some beautiful Game Over text on the screen, but it's just sitting and blocking our view right now. We should deactivate it, so it can reappear when the game ends.

1. In GameManager.cs, create a new public TextMeshProUGUI gameOverText; and assign the Game Over object to it in the inspector
2. Uncheck the Active checkbox to deactivate the Game Over text by default
3. In Start(), activate the Game Over text

Don't worry : We're just doing this temporarily to make sure it works

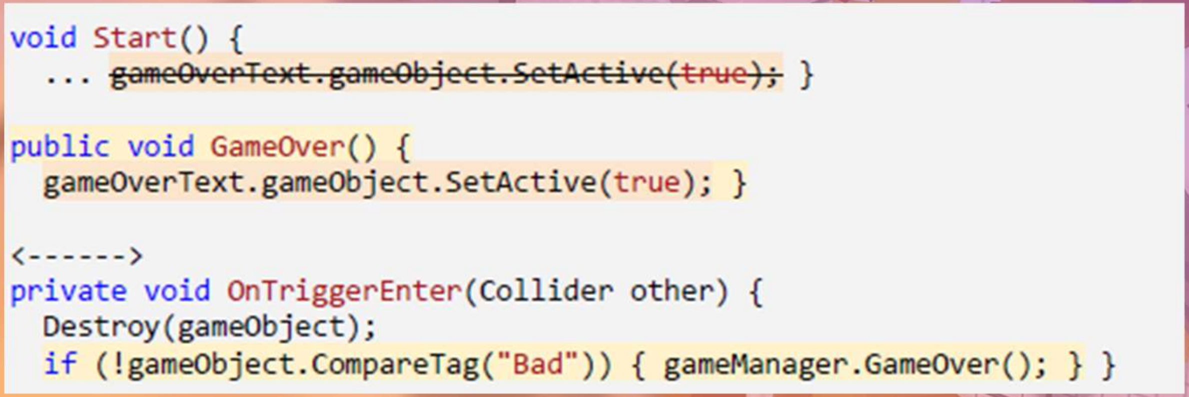


```
public TextMeshProUGUI gameOverText;

void Start() {
  ...
  gameOverText.gameObject.SetActive(true); }
```

Game Over – Step 3 : Create GameOver function

We've temporarily made the "Game Over" text appear at the start of the game, but we actually want to trigger it when one of the "Good" objects is missed and falls.

1. Create a new public void GameOver() function, and move the code that activates the game over text inside it
2. In Target.cs, call gameManager.GameOver() if a target collides with the sensor
3. Add a new "Bad" tag to the Bad object, add a condition that will only trigger game over if it's not a bad object

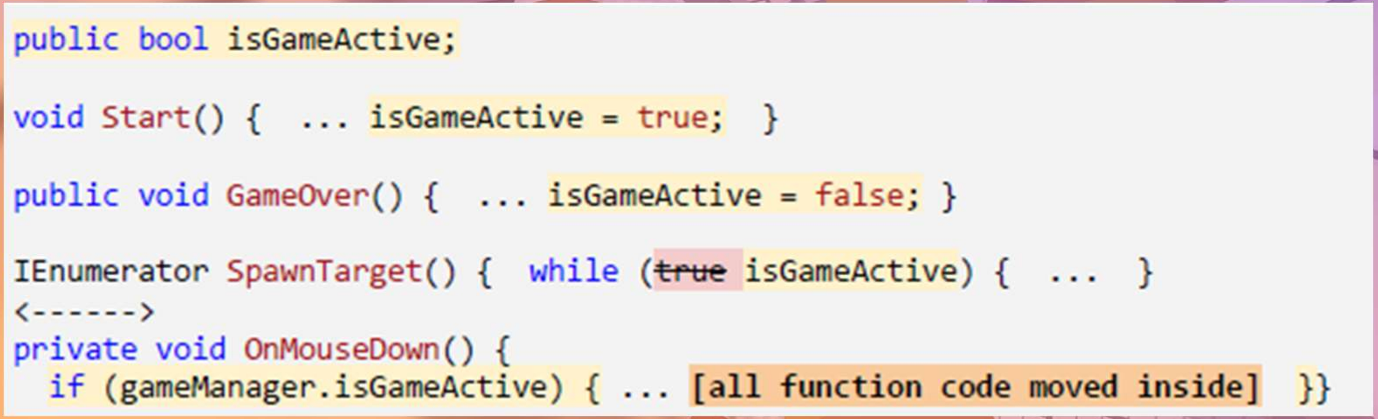


```
void Start() {
  ... ~~gameOverText.gameObject.SetActive(true);~~ }

public void GameOver() {
  gameOverText.gameObject.SetActive(true); }

<------>
private void OnTriggerEnter(Collider other) {
  Destroy(gameObject);
  if (!gameObject.CompareTag("Bad")) { gameManager.GameOver(); } }
```

GameOver

The "Game Over" message appears exactly when we want it to, but the game itself continues to play. In order to truly halt the game and call this a "Game Over', we need to stop spawning targets and stop generating score for the player.

1. Create a new public bool isGameActive;
2. As the first line in Start(), set isGameActive = true; and in GameOver(), set isGameActive = false;
3. To prevent spawning, in the SpawnTarget() coroutine, change while (true) to while (isGameActive)
4. To prevent scoring, in Target.cs, in the OnMouseDown() function, add the condition if (gameManager.isGameActive) {



```
public bool isGameActive;

void Start() {  ... isGameActive = true;  }

public void GameOver() {  ... isGameActive = false; }

IEnumerator SpawnTarget() {  while (~~true~~ isGameActive) {  ...  }
<------>
private void OnMouseDown() {
  if (gameManager.isGameActive) { ... [all function code moved inside]  }}
```

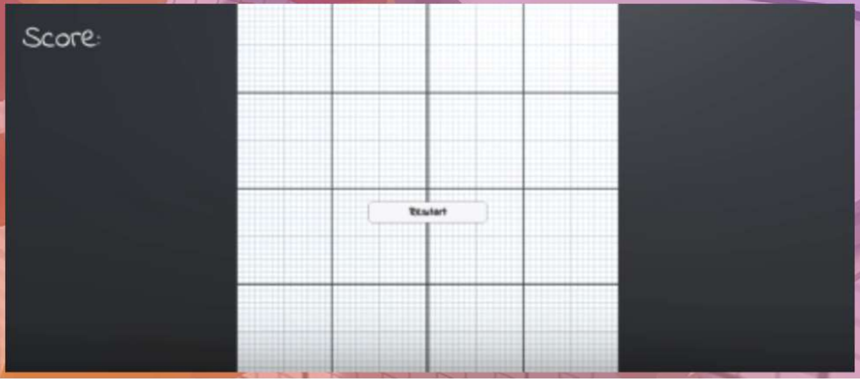
Game Over – Step 5 : Add a Restart button

Our Game Over mechanics are working like a charm, but there's no way to replay the game. In order to let the player restart the game, we will create our ﬁrst UI button

1. Right-click on the Canvas and Create > UI > Button
   Note: You could also use Button - TextMeshPro for more control over the button's text.
2. Rename the button "Restart Button"
3. Temporarily reactivate the Game Over text in order to reposition the Restart Button nicely with the text, then deactivate it again
4. Select the Text child object, then edit its Text to say "Restart", its Font, Style, and Size

New Concept :

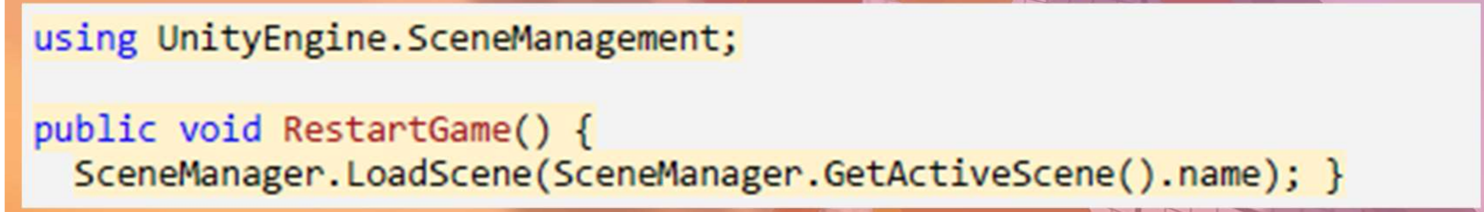
Buttons



Game Over – Step 6 : Make the restart button work

We've added the Restart button to the scene and it LOOKS good, but now we need to make it actually work and restart the game.

1. In GameManager.cs, add using UnityEngine.SceneManagement;
2. Create a new public void RestartGame() function that reloads the current scene
3. In the Button's inspector, click + to add a new On Click event, drag it in the Game Manager object and select the GameManager.RestartGame function



```
using UnityEngine.SceneManagement;

public void RestartGame() {
  SceneManager.LoadScene(SceneManager.GetActiveScene().name); }
```

New Concept : Scene Management

New Concept : On Click Event

Don't worry : The restart button is just sitting there for now, but we will fix it later

Game Over – Step 7 : Show restart button on game over

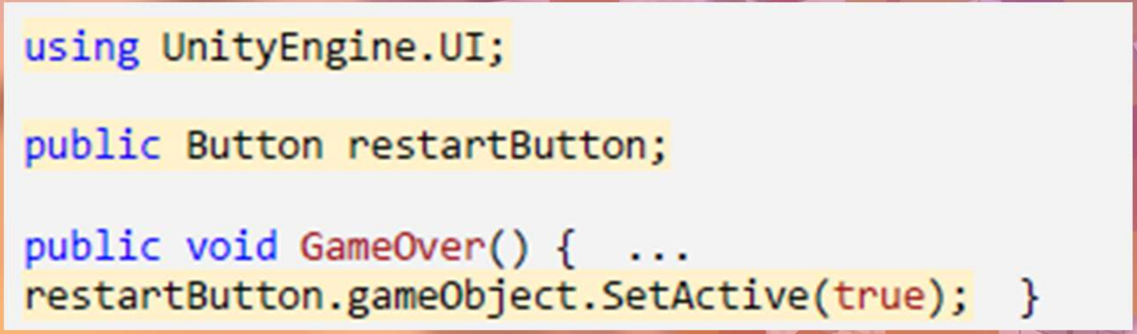
The Restart Button looks great, but we don't want it in our faces throughout the entire game. Similar to the "Game Over" message, we will turn off the Restart Button while the game is active.

1. At the top of GameManager.cs add using UnityEngine.UI;
2. Declare a new public Button restartButton; and assign the Restart Button to it in the inspector
3. Uncheck the "Active" checkbox for the Restart Button in the inspector
4. In the GameOver function, activate the Restart Button

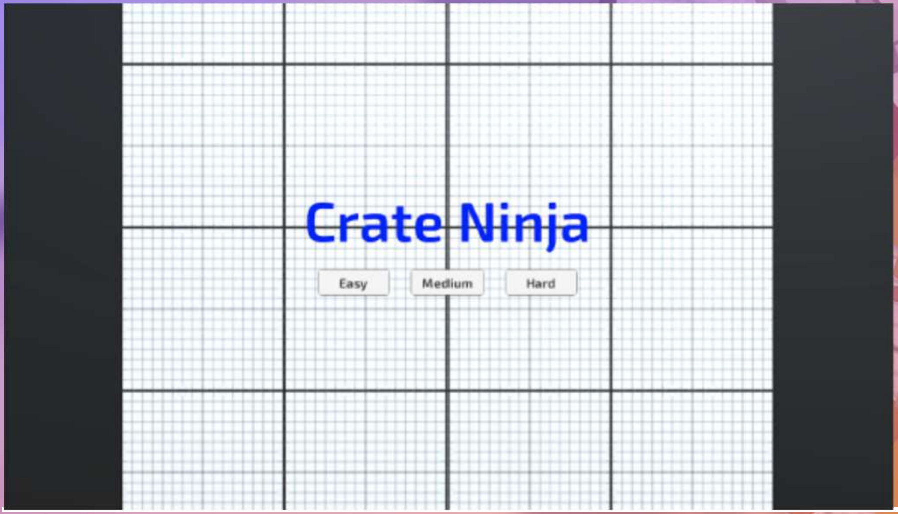
New Concept : Scene Management

New Concept : On Click Event

Don't worry : The restart button is just sitting there for now, but we will fix it later



```
using UnityEngine.UI;

public Button restartButton;

public void GameOver() {  ...
restartButton.gameObject.SetActive(true);  }
```



What's the Difficulty? – ง่ายจังเลยยย...มาปรับความยากกัน

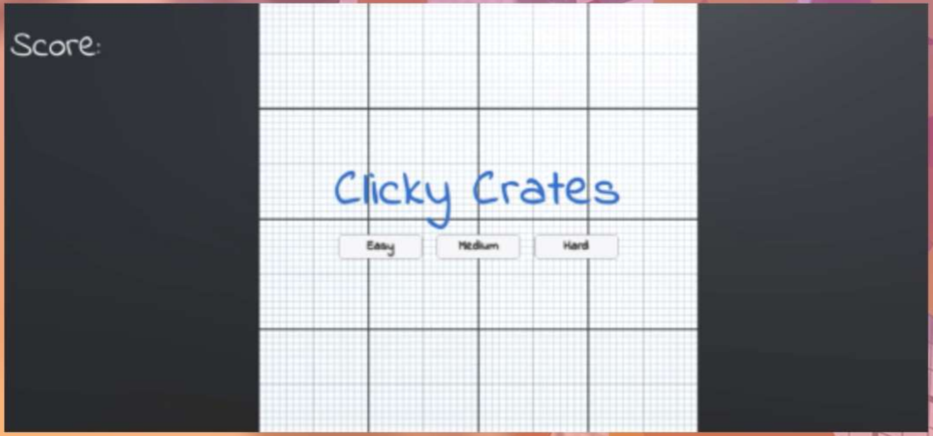
What’s the Difficulty?

- Step 1 : Create Title text and menu buttons
- Step 2 : Add a DifficultyButton script
- Step 3 : Call SetDifficulty on button click
- Step 4 : Make your buttons start the game
- Step 5 : Deactivate Title Screen on StartGame
- Step 6 : Use a parameter to change difficulty

menu buttons

The first thing we should do is create all of the UI elements we're going to need. This includes a big title, as well as three difficulty buttons.

1. Duplicate your Game Over text to create your Title Text, editing its name, text and all of its attributes
2. Duplicate your Restart Button and edit its attributes to create an "Easy Button" button
3. Edit and duplicate the new Easy button to create a"Medium Button" and a "Hard Button"

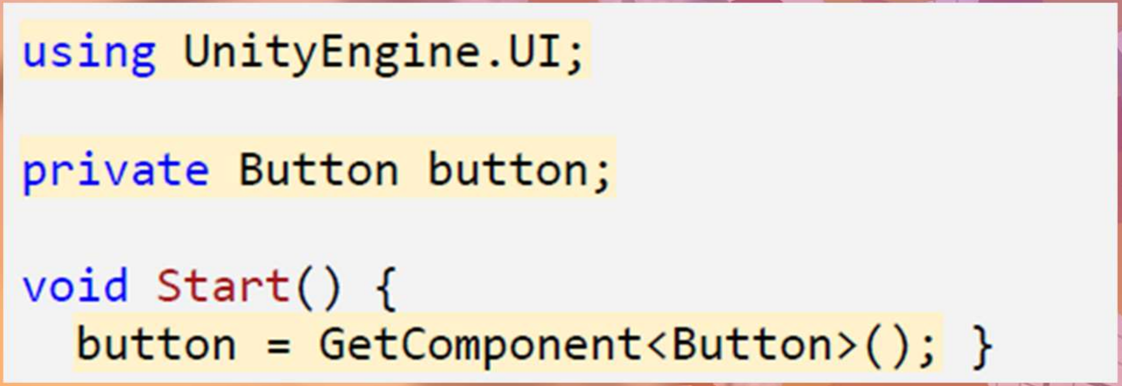


Tip : You can position the title and buttons however you want, but you should try to keep them central and visible to the player

script

Our difficulty buttons look great, but they don't actually do anything. If they're going to have custom functionality, we first need to give them a new script.

1. For all 3 new buttons, in the Button component, in the **On Click ()** section, click the minus (-) button to remove the RestartGame functionality
2. Create a new DifficultyButton.cs script and attach it to all 3 buttons
3. Add using UnityEngine.UI to your imports
4. Create a new private Button button; variable and initialize it in Start()

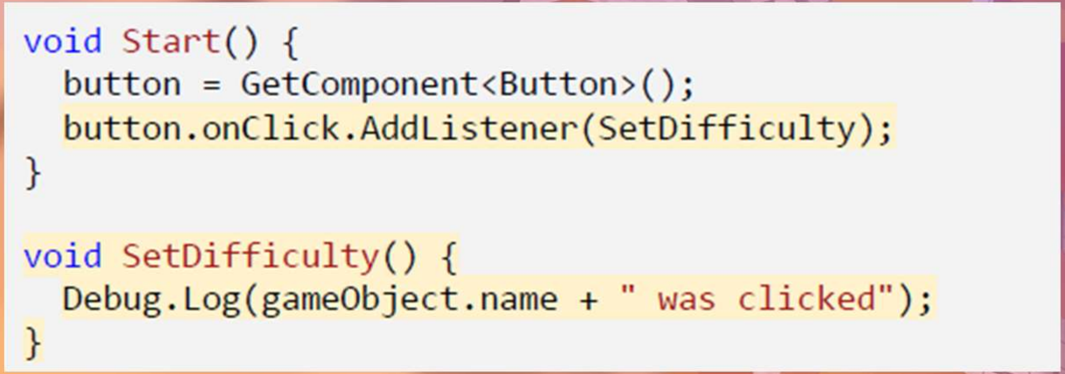


```
using UnityEngine.UI;

private Button button;

void Start() {
  button = GetComponent<Button>(); }
```

button click

Now that we have a script for our buttons, we can create a SetDifficulty method and tie that method to the click of those buttons

1. Create a new void SetDifficulty function, and inside it, Debug.Log(gameObject.name + " was clicked");
2. Add the button listener to call the SetDifficulty function



```
void Start() {
  button = GetComponent<Button>();
  button.onClick.AddListener(SetDifficulty);
}

void SetDifficulty() {
  Debug.Log(gameObject.name + " was clicked");
}
```

New Function : AddListener

Don't worry : onClick.AddListener is similar what we did in the inspector with the Restart button

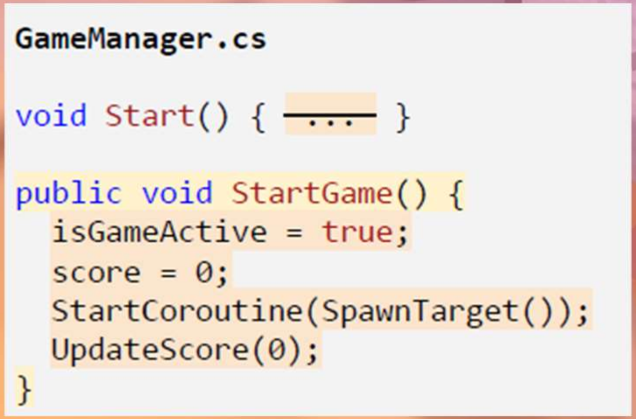
Don't worry : We're just using Debug for testing, to make sure the buttons are working

start the game

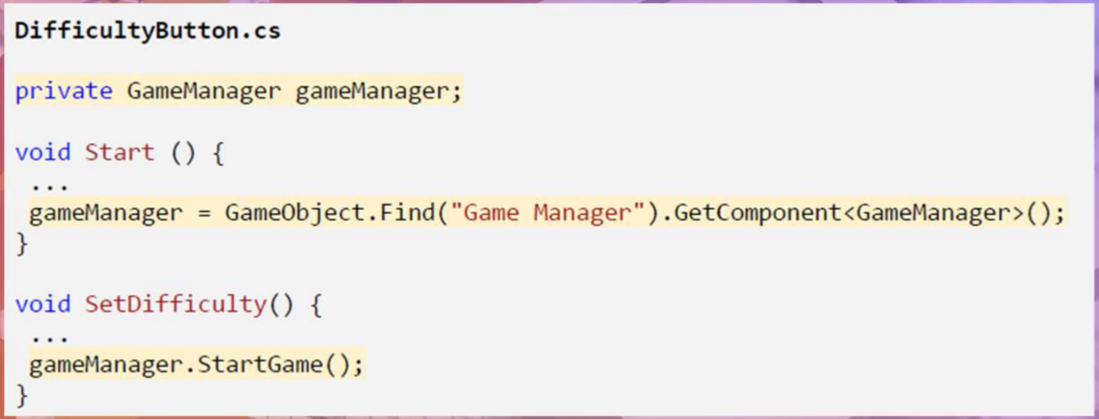
The Title Screen looks great if you ignore the target objects bouncing around, but we have no way of actually starting the game. We need a StartGame function that can communicate with SetDifficulty.

1. In GameManager.cs, create a new public void StartGame() function and move everything from Start() into it
2. In DifficultyButton.cs, create a new private GameManager gameManager; and initialize it in Start()
3. In the SetDifficulty() function, call gameManager.startGame();

Don't worry : Title objects don't disappear yet - we'll do that next



```
GameManager.cs

void Start() { ~~...~~ }

public void StartGame() {
  isGameActive = true;
  score = 0;
  StartCoroutine(SpawnTarget());
  UpdateScore(0);
}
```

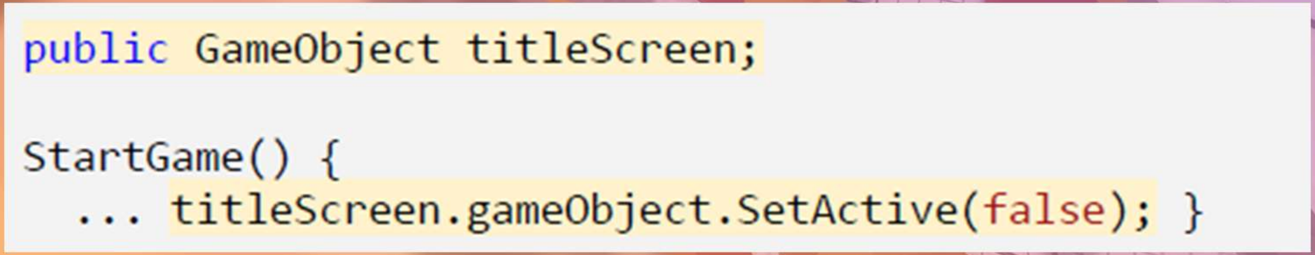


```
DifficultyButton.cs

private GameManager gameManager;

void Start () {
 ...
 gameManager = GameObject.Find("Game Manager").GetComponent<GameManager>();
}

void SetDifficulty() {
 ...
 gameManager.StartGame();
}
```

on StartGame

If we want the title screen to disappear when the game starts, we should store them in an empty object rather than turning them off individually. Simply deactivating the single empty parent object makes for a lot less work.

1. Right-click on the Canvas and Create > Empty Object, rename it “Title Screen”, and drag the 3 buttons and title onto it
2. In GameManager.cs, create a new public GameObject titleScreen; and assign it in the inspector
3. In StartGame(), deactivate the title screen object



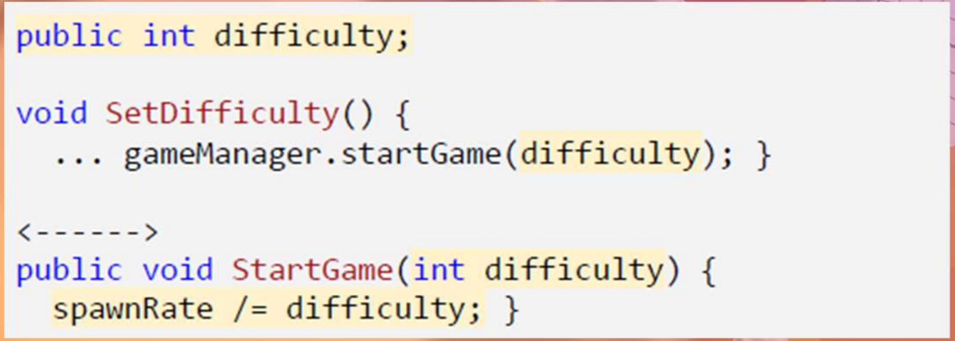
```
public GameObject titleScreen;

StartGame() {
  ... titleScreen.gameObject.SetActive(false); }
```

change difficulty

The difficulty buttons start the game, but they still don't change the game's difficulty. The last thing we have to do is actually make the difficulty buttons affect the rate that target objects spawn.

1. In DifficultyButton.cs, create a new public int difficulty variable, then in the Inspector, assign the Easy difficulty as 1, Medium as 2, and Hard as 3
2. Add an int difficulty parameter to the StartGame() function
3. In StartGame(), set spawnRate /= difficulty;
4. Fix the error in DifficultyButton.cs by passing the difficulty parameter to StartGame(difficulty)

New Concept : /= operator



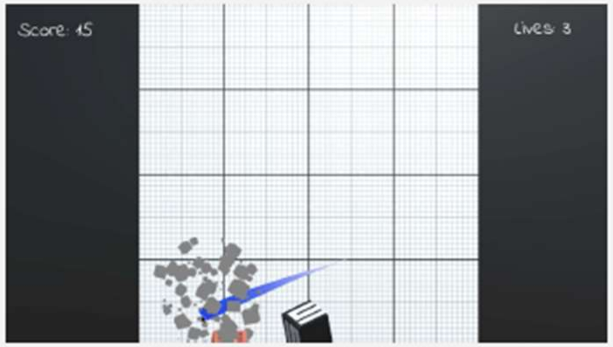
```
public int difficulty;

void SetDifficulty() {
  ... gameManager.startGame(difficulty); }

<------>
public void StartGame(int difficulty) {
  spawnRate /= difficulty; }
```

Step 1: Overview

This tutorial outlines four potential bonus features for the Quick Click Prototype at varying levels of difficulty:

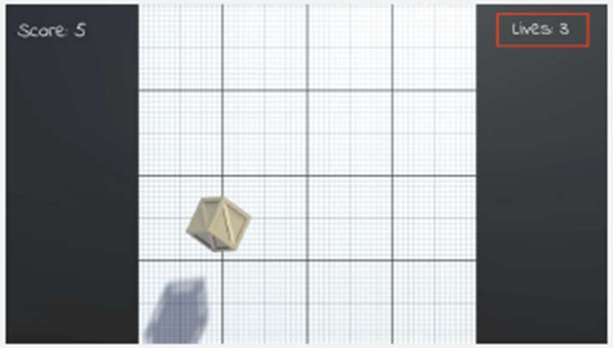
- **Easy**: Lives UI
- **Medium**: Music volume
- **Hard**: Pause menu
- **Expert**: Click-and-swipe

Here's what the prototype could look like if you complete all four features:



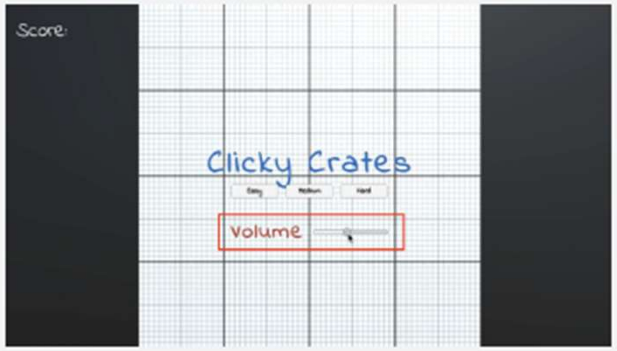
Step 2: Easy: Lives UI

Create a "Lives" UI element that counts down by 1 when an object leaves the bottom of the screen and triggers Game Over when Lives reaches 0.



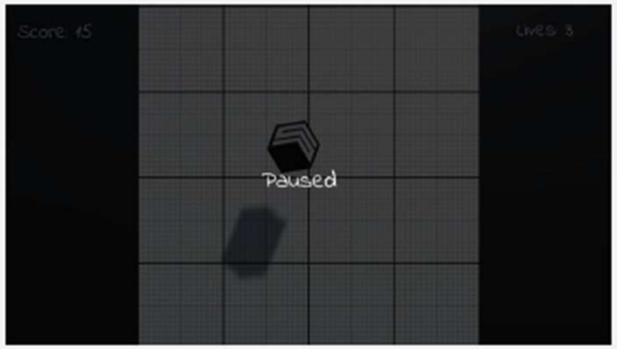
Step 3: Medium: Music volume

Add background music and a UI Slider element to adjust the volume.
Background music adds a lot of energy to a game, but not everyone likes it, so it's good to give people the option to lower the volume.



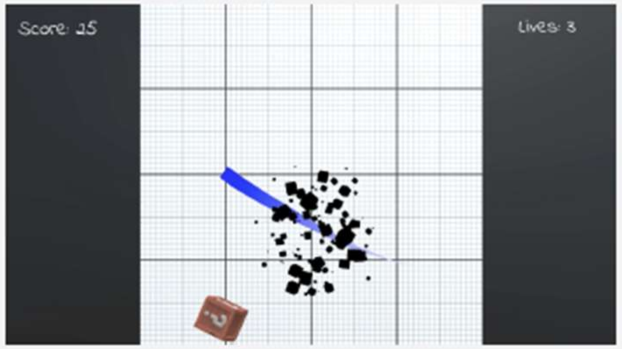
Step 5: Hard: Pause menu

During gameplay, allow the user to press a key to toggle between pausing and resuming the game, where a pause screen comes up while the game is paused.



Step 6: Expert: Click-and-swipe

Program click-and-swipe functionality instead of clicking, generating a trail where the mouse has been dragged. This does make the game easier, so you might also want to increase the gameplay difficulty on all levels if you implement this.



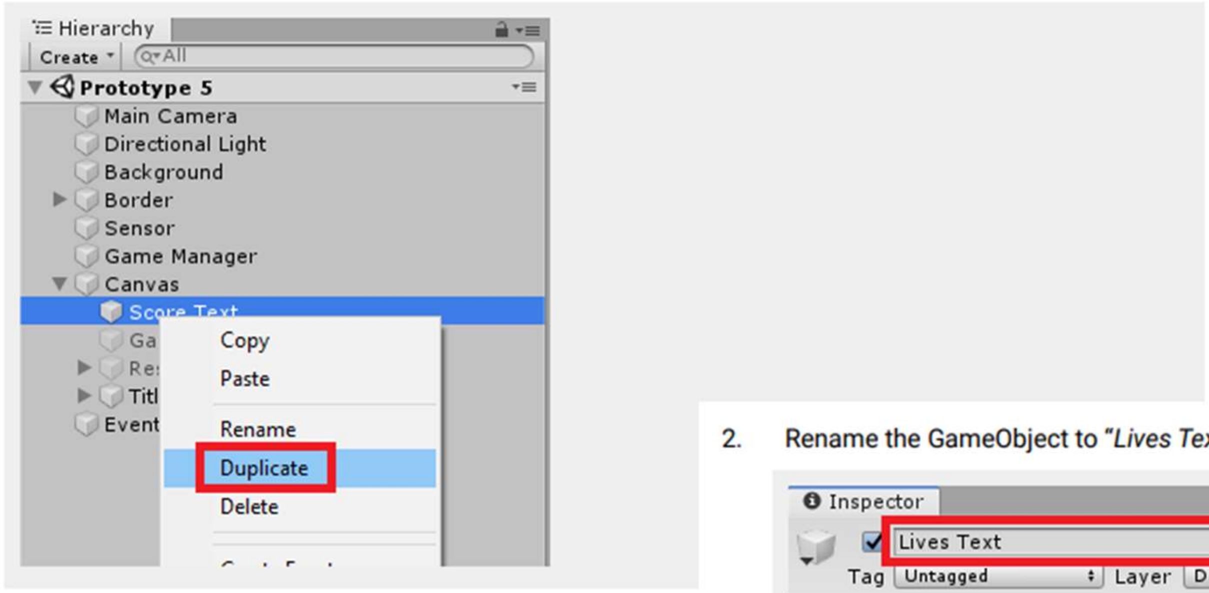
Step 7: Hints and solution walkthrough

Hints:

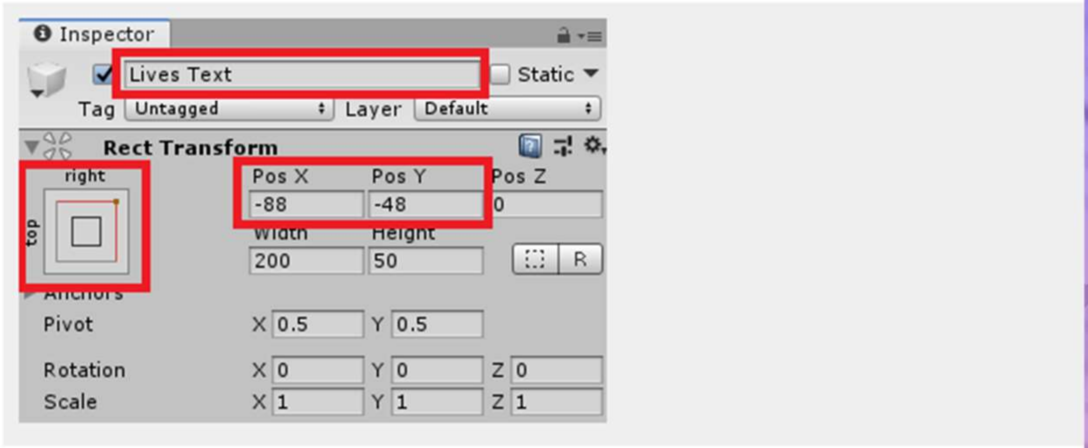
- Easy: Lives UI
  - Try using a Text GameObject like we did for the score
- Medium: Music volume
  - Try using the event on the Slider element
- Hard: Pause menu
  - Try using `Time.timeScale`
- Expert: Click-and-swipe
  - Camera.ScreenToWorldPoint will help convert a screen space position to world position

Easy - Lives UI

1. Duplicate the *Score Text* GameObject by right-clicking on it in the Hierarchy and selecting **Duplicate**



2. Rename the GameObject to *"Lives Text"* and align it to the Top-right of the screen.



Inspector

Lives Text

Static

Tag: Untagged

Layer: Default

Rect Transform

right / top

Pos X	Pos Y	Pos Z
-88	-48	0

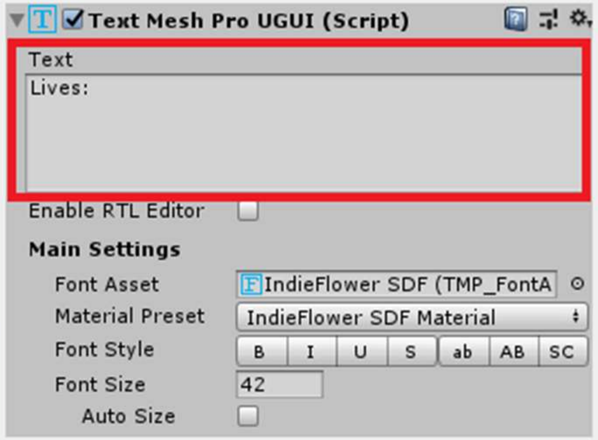
Width	Height
200	50

Pivot: X 0.5, Y 0.5

Rotation: X 0, Y 0, Z 0

Scale: X 1, Y 1, Z 1

3. Change the **Text** field in the **TextMeshPro - Text (UI)** component to “Lives:”


Text Mesh Pro UGUI (Script)

Text
Lives:

Enable RTL Editor ☐

Main Settings

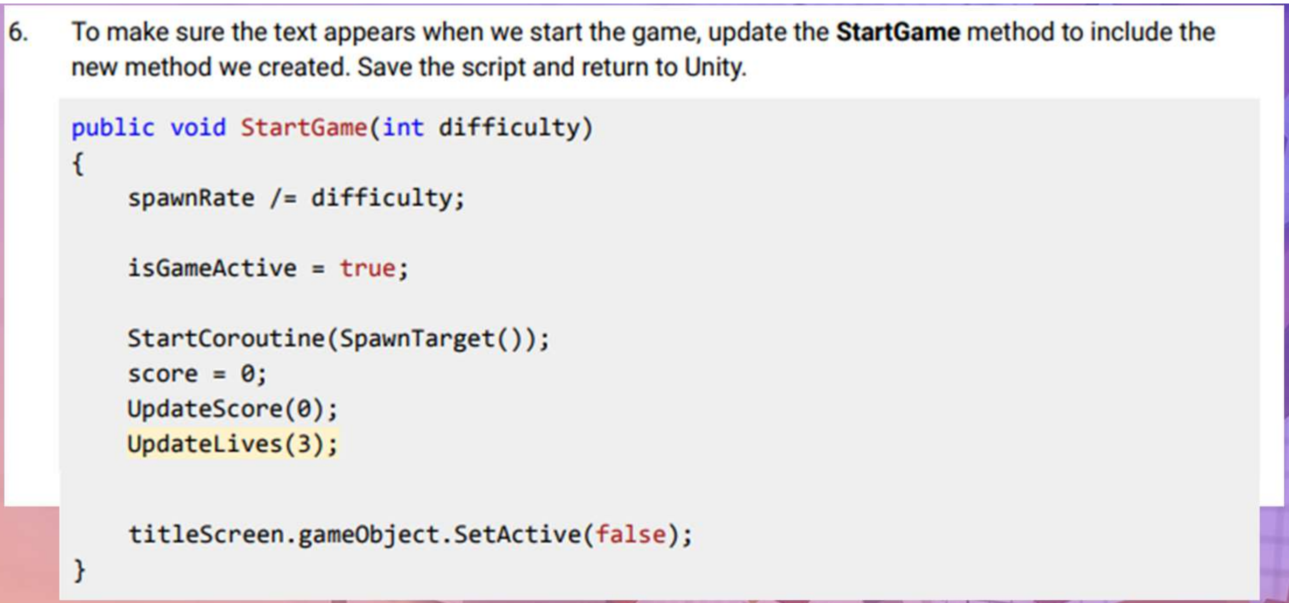
Font Asset	IndieFlower SDF (TMP_FontA
Material Preset	IndieFlower SDF Material
Font Style	B I U S ab AB SC
Font Size	42
Auto Size	☐

4. Open up “GameManager.cs” from the scripts folder and add two new variables - one for the text and the other for the amount of lives available.

```
public TextMeshProUGUI livesText;
private int lives;
```

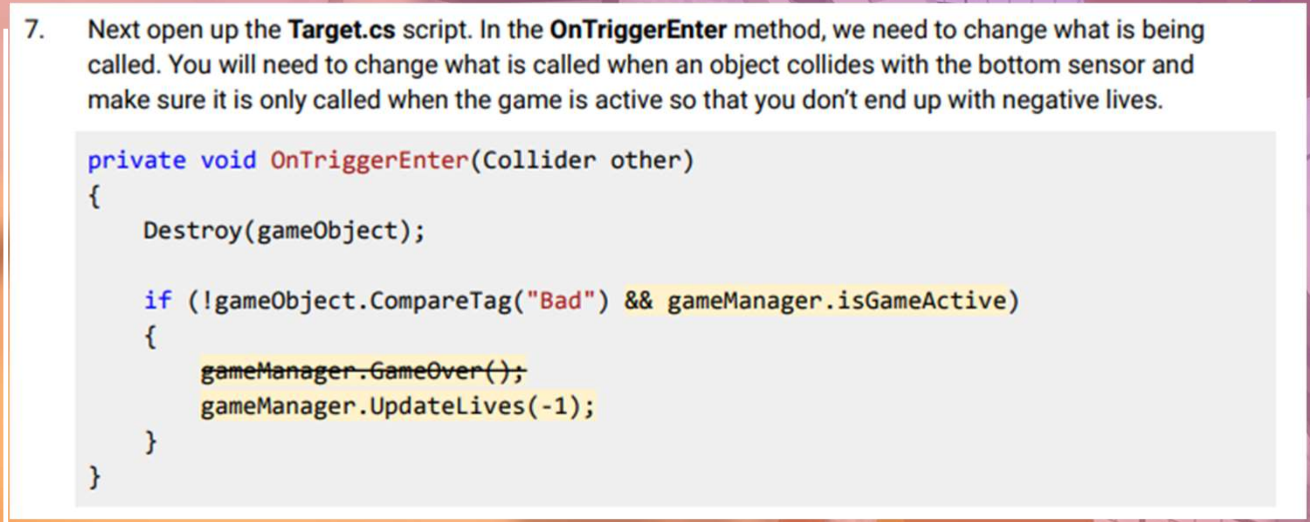
5. Just after the **UpdateScore** method, add a new method called **UpdateLives**, this will adjust the amount of lives we have and then display the result to the Text GameObject we created earlier. It will also check to see if we have run out of lives, if we have it will call the **GameOver** method.

```
public void UpdateLives(int livesToChange)
{
    lives += livesToChange;
    livesText.text = "Lives: " + lives;
    if (lives <= 0)
    {
        GameOver();
    }
}
```



6. To make sure the text appears when we start the game, update the **StartGame** method to include the new method we created. Save the script and return to Unity.

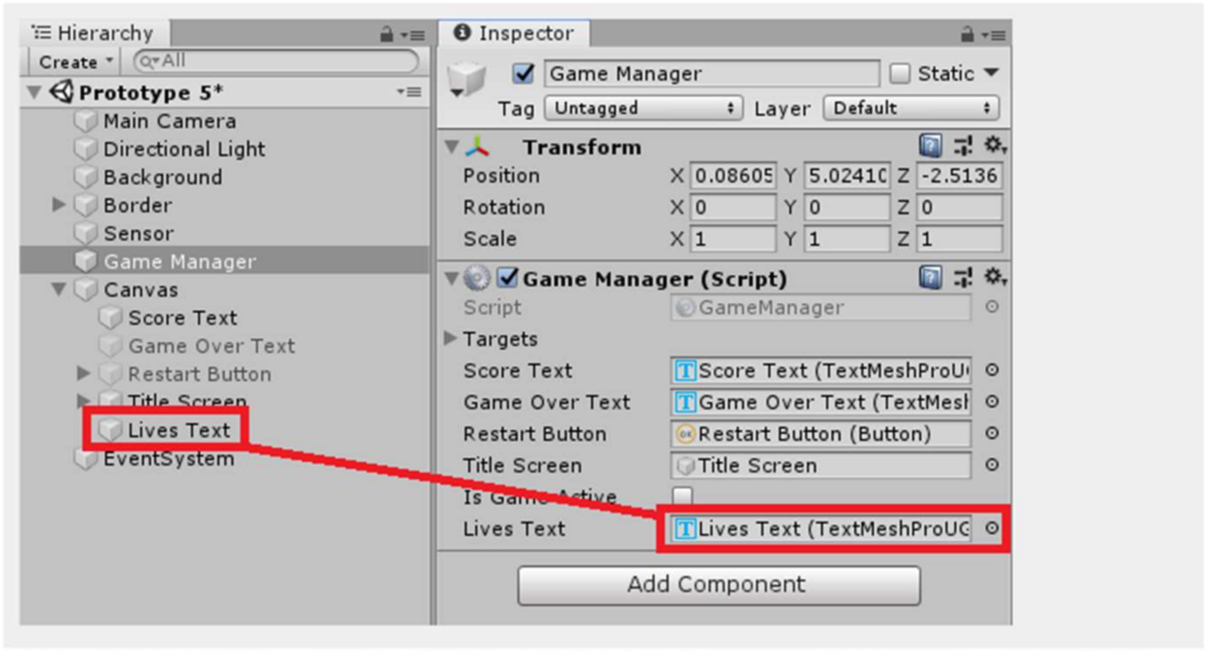
```
public void StartGame(int difficulty)
{
    spawnRate /= difficulty;

    isGameActive = true;

    StartCoroutine(SpawnTarget());
    score = 0;
    UpdateScore(0);
    UpdateLives(3);


    titleScreen.gameObject.SetActive(false);
}
```



7. Next open up the **Target.cs** script. In the **OnTriggerEnter** method, we need to change what is being called. You will need to change what is called when an object collides with the bottom sensor and make sure it is only called when the game is active so that you don't end up with negative lives.

```
private void OnTriggerEnter(Collider other)
{
    Destroy(gameObject);

    if (!gameObject.CompareTag("Bad") && gameManager.isGameActive)
    {
        ~~gameManager.GameOver();~~
        gameManager.UpdateLives(-1);
    }
}
```

8. Save the script and return to Unity. We now need to set up our GameManager in the Inspector. Select the GameManger in the Hierarchy to display all the components in the Inspector. Drag the Lives Text that was created earlier into the Lives Text field on the **GameManager (Script)** component.



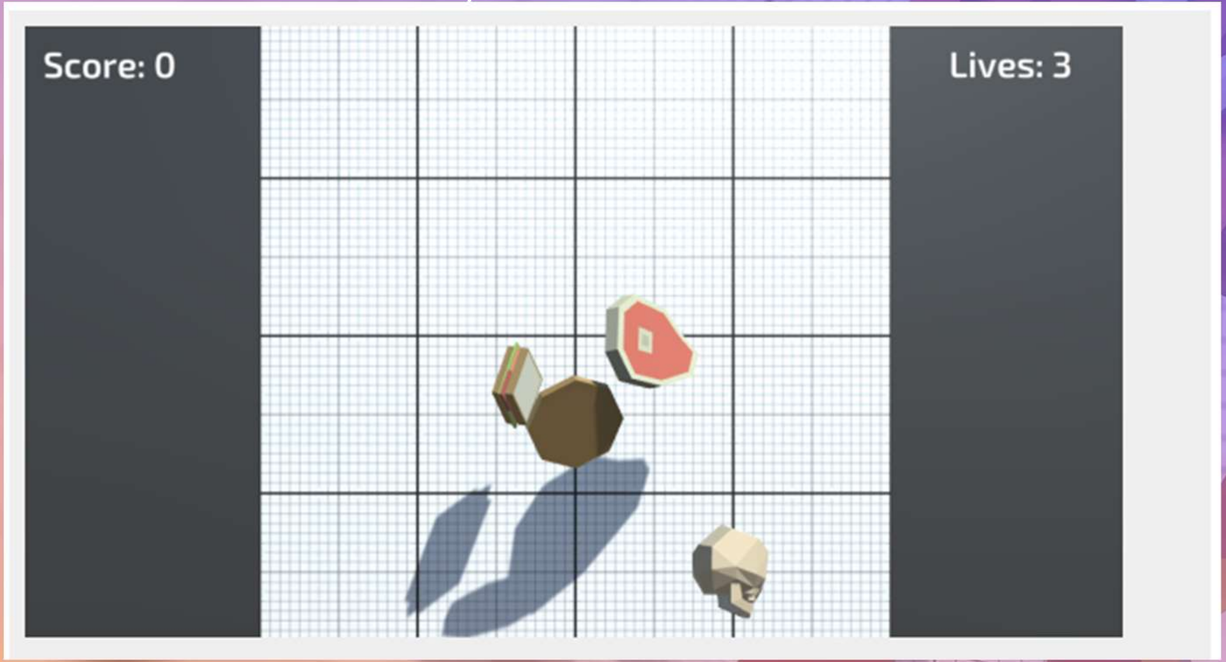
Hierarchy: Prototype 5*, Main Camera, Directional Light, Background, Border, Sensor, Game Manager, Canvas, Score Text, Game Over Text, Restart Button, Title Screen, Lives Text, EventSystem

Inspector: Game Manager, Tag Untagged, Layer Default

Transform
Position X 0.08605 Y 5.0241C Z -2.5136
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

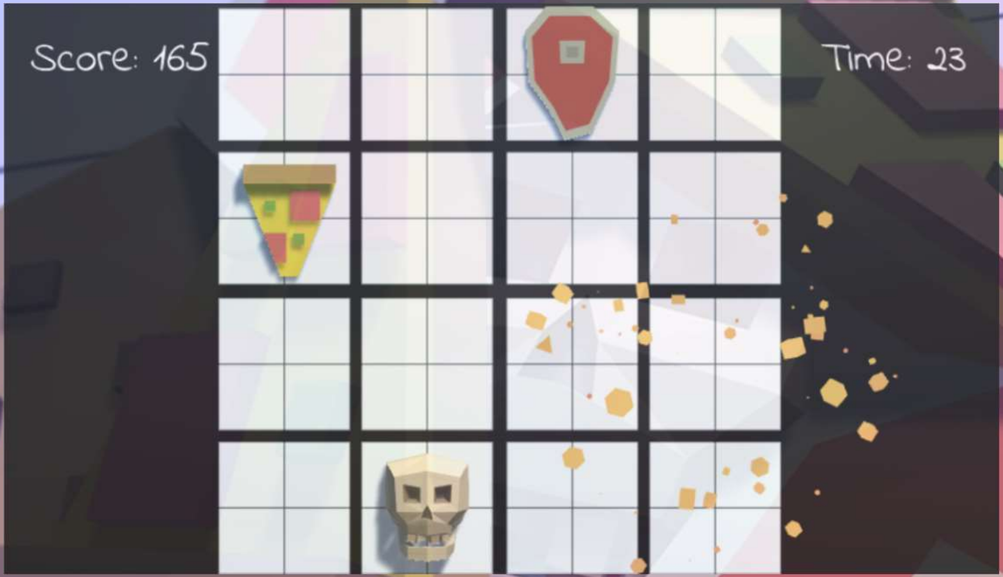
Game Manager (Script)
Script: GameManager
Targets
Score Text: Score Text (TextMeshProU
Game Over Text: Game Over Text (TextMesh
Restart Button: Restart Button (Button)
Title Screen: Title Screen
Is Game Active: ☐
Lives Text: Lives Text (TextMeshProUG

9. Save the scene and play it. Notice that when the objects fall off the screen, the lives counter goes down.



CHALLENGE 4

Whack-a-Food



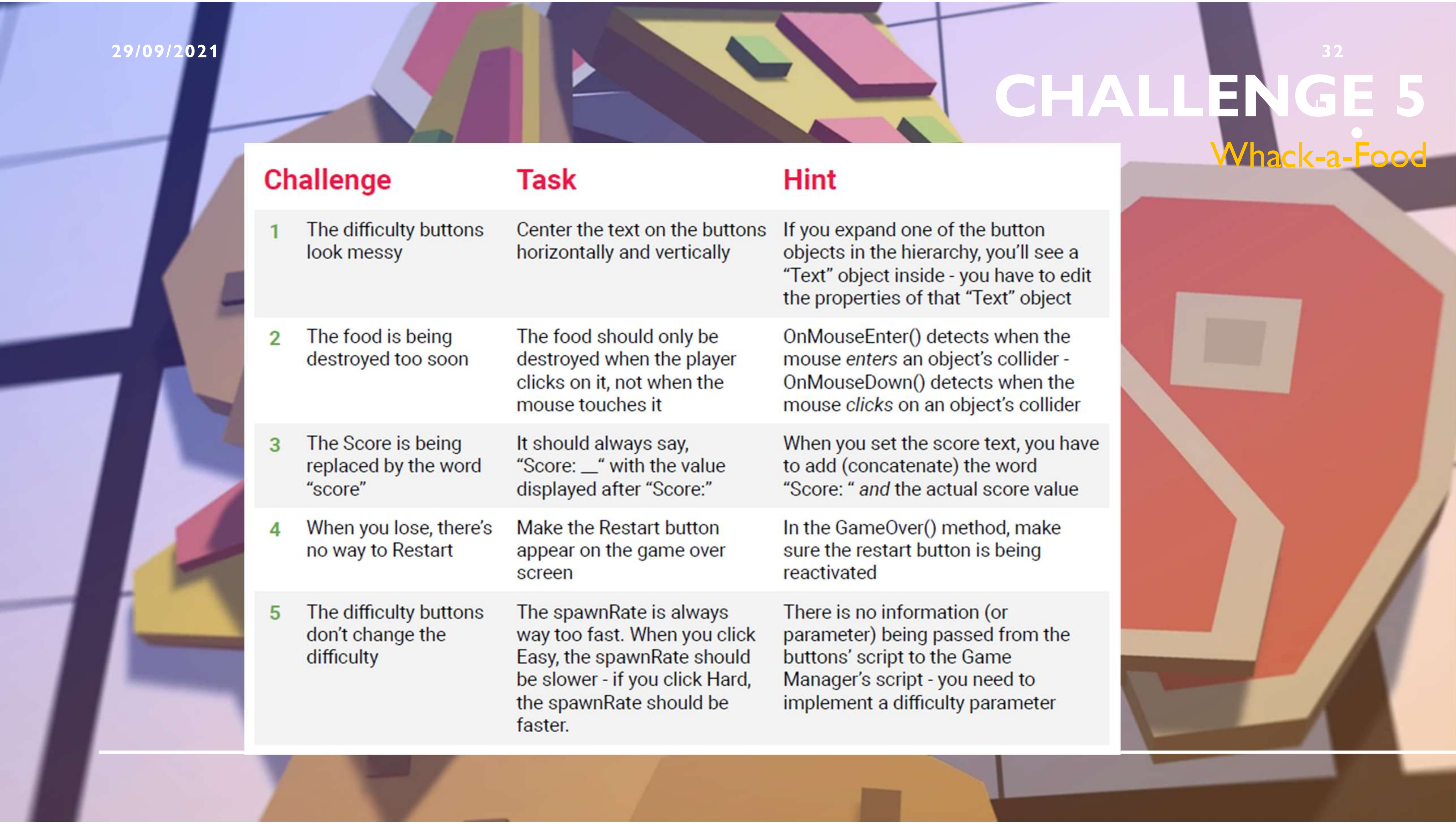
CHALLENGE 5

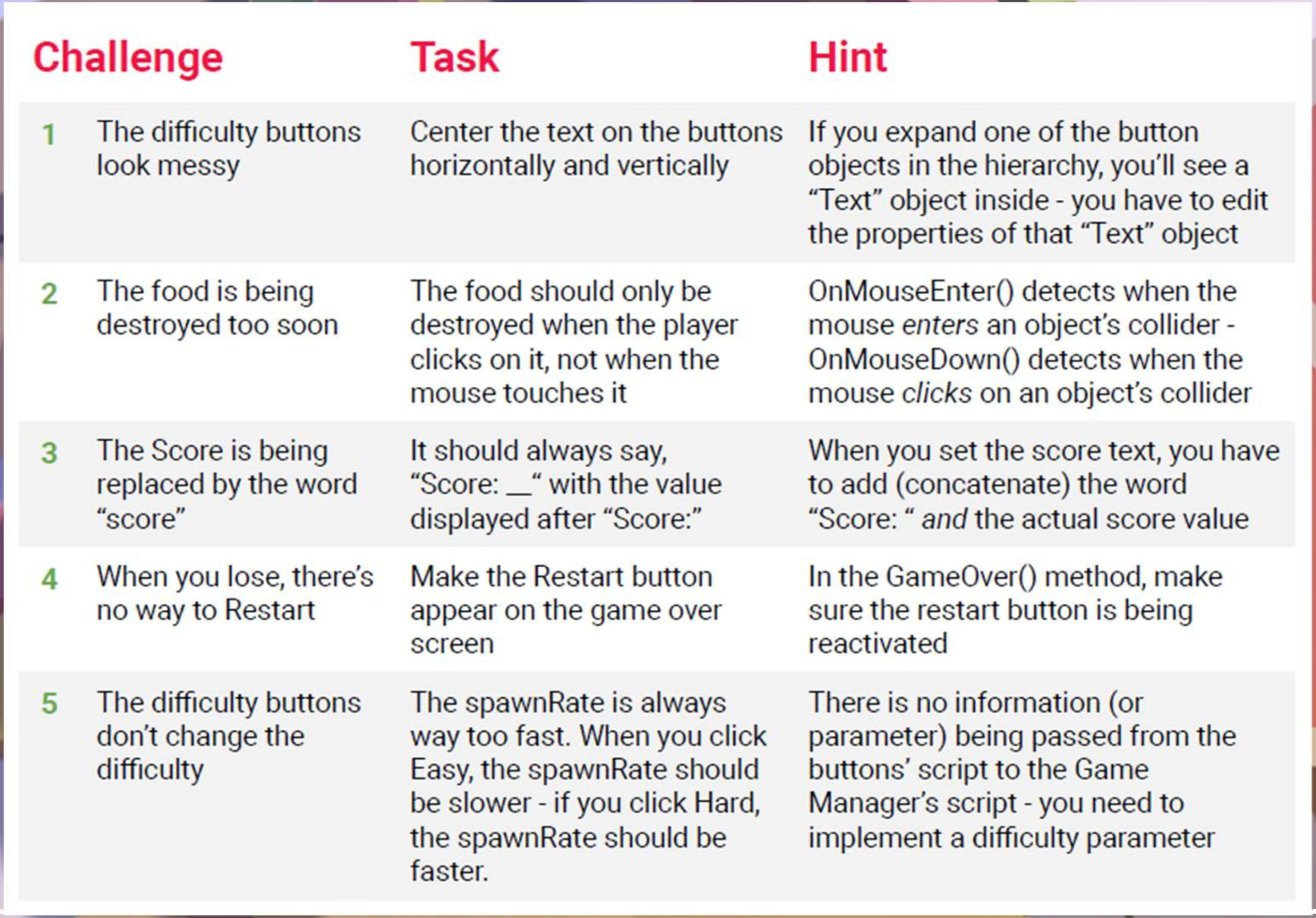
Whack-a-Food

Challenge Outcome:	- All of the buttons look nice with their text properly aligned - When you select a difficulty, the spawn rate changes accordingly - When you click a food, it is destroyed and the score is updated in the top-left - When you lose the game, a restart button appears that lets you play again
Challenge Objectives:	In this challenge, you will reinforce the following skills/concepts: - Working with text and button objects to get them looking the way you want - Using Unity's various mouse-related methods appropriately - Displaying variables on text objects properly using concatenation - Activating and deactivating objects based on game states - Passing information between scripts using custom methods and parameters

CHALLENGE 5

Whack-a-Food

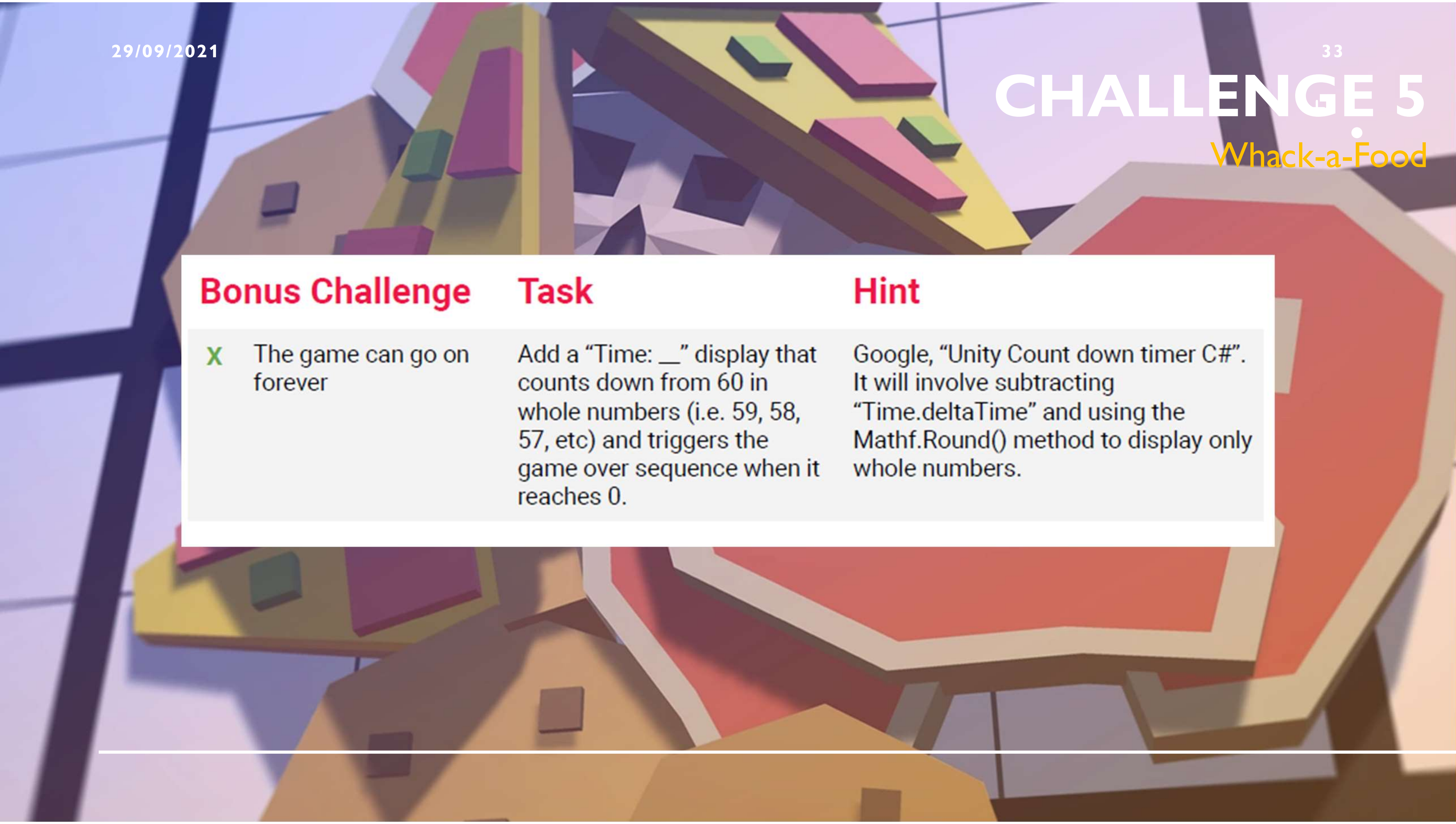


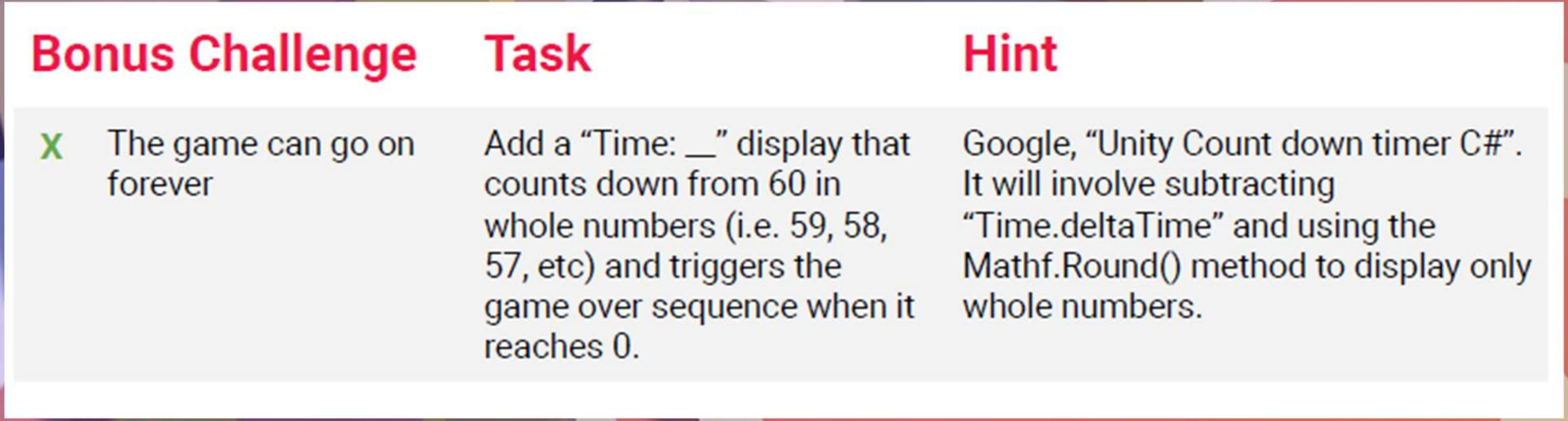


	Challenge	Task	Hint
1	The difficulty buttons look messy	Center the text on the buttons horizontally and vertically	If you expand one of the button objects in the hierarchy, you'll see a "Text" object inside - you have to edit the properties of that "Text" object
2	The food is being destroyed too soon	The food should only be destroyed when the player clicks on it, not when the mouse touches it	OnMouseEnter() detects when the mouse *enters* an object's collider - OnMouseDown() detects when the mouse *clicks* on an object's collider
3	The Score is being replaced by the word "score"	It should always say, "Score: __" with the value displayed after "Score:"	When you set the score text, you have to add (concatenate) the word "Score: " *and* the actual score value
4	When you lose, there's no way to Restart	Make the Restart button appear on the game over screen	In the GameOver() method, make sure the restart button is being reactivated
5	The difficulty buttons don't change the difficulty	The spawnRate is always way too fast. When you click Easy, the spawnRate should be slower - if you click Hard, the spawnRate should be faster.	There is no information (or parameter) being passed from the buttons' script to the Game Manager's script - you need to implement a difficulty parameter

CHALLENGE 5

Whack-a-Food





Bonus Challenge	Task	Hint
X The game can go on forever	Add a “Time: __” display that counts down from 60 in whole numbers (i.e. 59, 58, 57, etc) and triggers the game over sequence when it reaches 0.	Google, “Unity Count down timer C#”. It will involve subtracting “Time.deltaTime” and using the Mathf.Round() method to display only whole numbers.