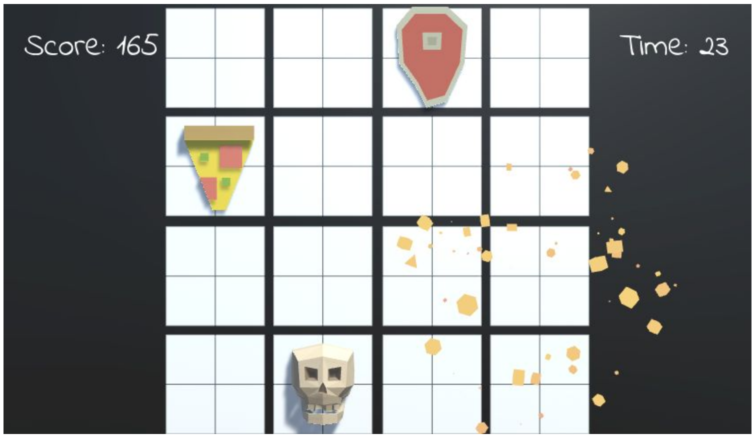
Challenge 5

Whack-a-Food



Challenge Overview: Put your User Interface skills to the test with this whack-a-mole-like challenge in which you have to get all the food that pops up on a grid while avoiding the skulls. You will have to debug buttons, mouse clicks, score tracking, restart sequences, and difficulty setting to get to the bottom of this one.

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 Instructions:

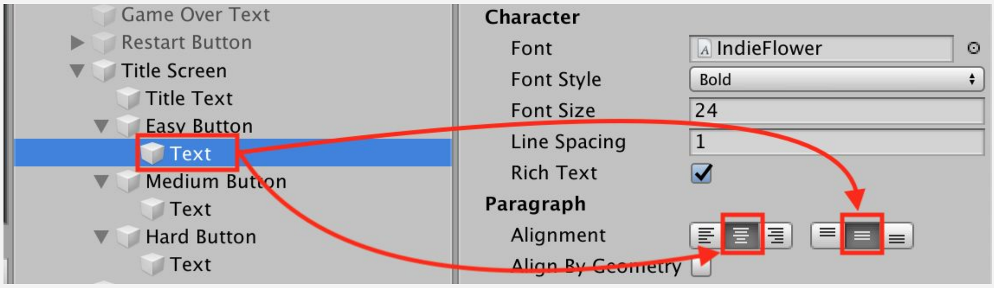
- Open your **Prototype 5** project
- **Download** the "Challenge 5 Starter Files" from the Tutorial Materials section, then double-click on it to **Import**
- In the *Project Window > Assets > Challenge 5 > **Instructions*** folder, use the "Challenge 5 - Outcome" video as a guide to complete the challenge

	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 spawn rate 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

	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.

Challenge Solution

1 Expand each of the “Easy”, “Medium”, and “Hard” buttons to access their “Text” object properties, then select the horizontal and vertical alignment buttons in the “Paragraph” properties



Game Over Text
Restart Button
Title Screen
Title Text
Easy Button
Text
Medium Button
Text
Hard Button
Text

Character
Font: IndieFlower
Font Style: Bold
Font Size: 24
Line Spacing: 1
Rich Text: ☑
Paragraph
Alignment
Align By Geometry: □

2 In TargetX.cs, change OnMouseEnter() to OnMouseDown()

```
private void OnMouse~~Enter~~ Down() {
```

3 In GameManagerX.cs, in UpdateScore(), concatenate the word “Score: “ with the score value:

```
public void UpdateScore(int scoreToAdd) {
   score += scoreToAdd;
   scoreText.text = ~~"score"~~ "Score: " + score;
}
```

4 In GameManagerX.cs, in GameOver(), change SetActive(false) to “true”

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

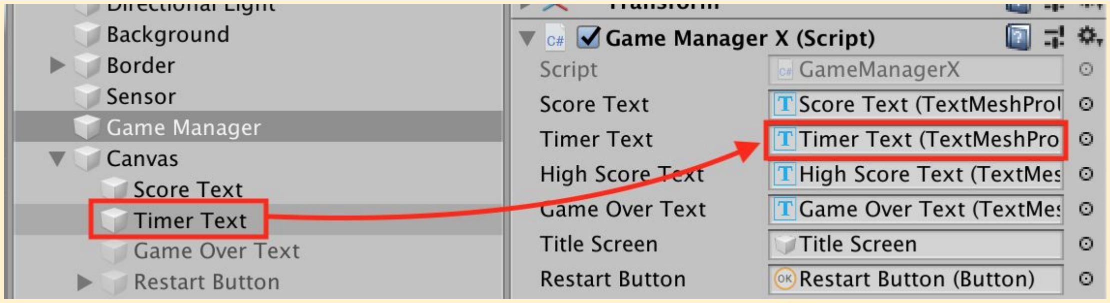
5 In GameManagerX.cs, in StartGame(), add an “int difficulty” parameter and divide the spawnRate by it. Then in DifficultyButtonX.cs, in SetDifficulty(), pass in the “difficulty” value from the buttons.

```
GameManagerX.cs
public void StartGame(int difficulty){
   spawnRate /= ~~5~~ difficulty;
   ...
}
```

```
DifficultyButtonX.cs
void SetDifficulty() {
   ...
   gameManagerX.StartGame(difficulty);
}
```

Bonus Challenge Solution

X1 Duplicate the "Score Text" object in the hierarchy to create a new "Timer text" object, then in GameManagerX.cs declare a new ***TextMeshProUGUI timerText*** variable and assign it in the inspector



Game Manager X (Script)

Script	GameManagerX
Score Text	Score Text (TextMeshProl
Timer Text	Timer Text (TextMeshPro
High Score Text	High Score Text (TextMes
Game Over Text	Game Over Text (TextMe:
Title Screen	Title Screen
Restart Button	Restart Button (Button)

X2 In GameManagerX.cs, in StartGame(), set your new timerText variable to your starting time

```
public void StartGame(int difficulty) {
  ...
  timeLeft = 60;
}
```

X3 In GameManagerX.cs, add an Update() function that, if the game is active, subtracts from the timeLeft and sets the timerText to a rounded version of that timeLeft. Then, if timeLeft is less than zero, calls the game over method.

```
private void Update() {
  if (isGameActive) {
    timeLeft -= Time.deltaTime;
    timerText.SetText("Time: " + Mathf.Round(timeLeft));
    if (timeLeft < 0) {
      GameOver();
    }
  }
}
```