unity

5.2 Keeping Score

Steps:

Step 1: Add Score text position it on screen

Step 2: Edit the Score Text's properties

Step 3: Initialize score text and variable

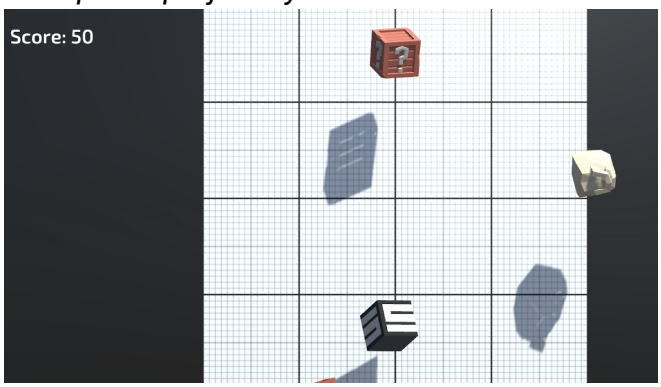
Step 4: Create a new UpdateScore method

Step 5: Add score when targets are destroyed

Step 6: Assign a point value to each target

Step 7: Add a Particle explosion

Example of project by end of lesson



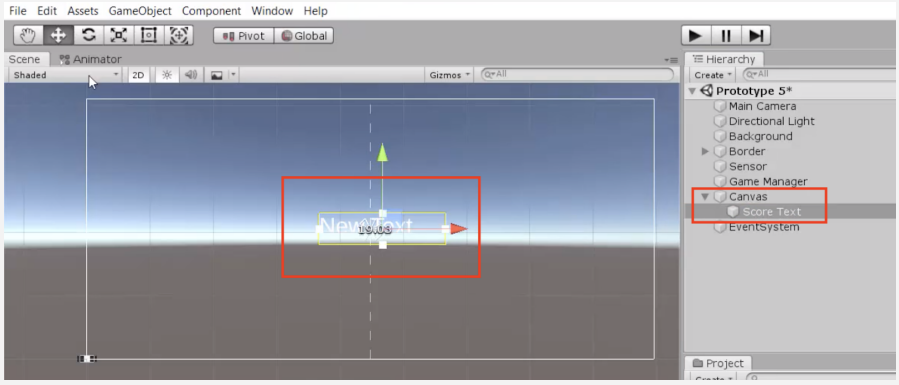
Length:	60 minutes
Overview:	Objects fly into the scene and the player can click to destroy them, but nothing happens. In this lesson, we will display a score in the user interface that tracks and displays the player's points. We will give each target object a different point value, adding or subtracting points on click. Lastly, we will add cool explosions when each target is destroyed.
Project Outcome:	A "Score: " section will display in the UI, starting at zero. When the player clicks a target, the score will update and particles will explode as the target is destroyed. Each "Good" target adds a different point value to the score, while the "Bad" target subtracts from the score.
Learning Objectives:	By the end of this lesson, you will be able to: - Create UI Elements in the Canvas - Lock elements and objects into place with Anchors - Use variables and script communication to update elements in the UI

Step 1: Add Score text position it on screen

In order to display the score on-screen, we need to add our very first UI element.

1. In the Hierarchy, *Create > UI > **TextMeshPro text***, then if prompted click the button to **Import TMP Essentials**
2. Rename the new object "Score Text", then **zoom out** to see the **canvas** in Scene view
3. Change the **Anchor Point** so that it is anchored from the **top-left corner**
4. In the inspector, change its **Pos X** and **Pos Y** so that it is in the top-left corner

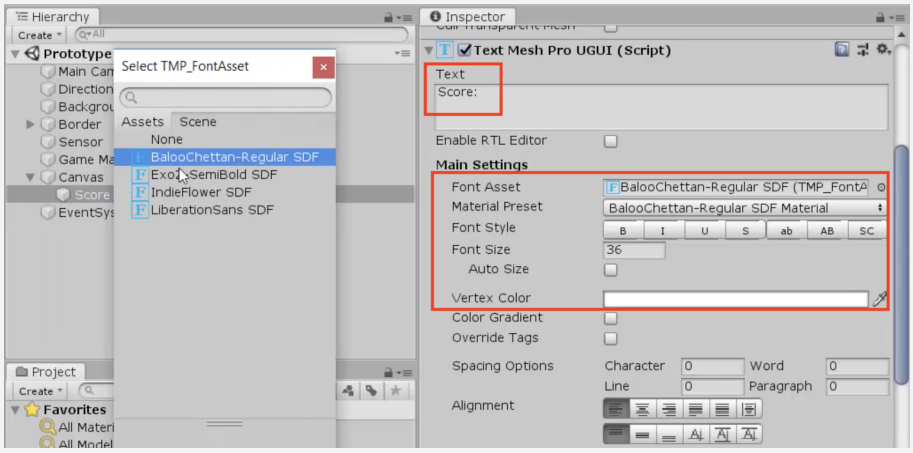
- **New Concept:** Text Mesh Pro / TMPro
- **New Concept:** Canvas
- **New Concept:** Anchor Points
- **Tip:** Look at how it displays in scene vs game view. It may be hard to see white text depending on the background



Step 2: Edit the Score Text's properties

Now that the basic text is in the scene and positioned properly, we should edit its properties so that it looks nice and has the correct text.

1. Change its text to "Score:"
2. Choose a **Font Asset**, **Style**, **Size**, and **Vertex color** to look good with your background



Step 3: Initialize score text and variable

We have a great place to display score in the UI, but nothing is displaying there! We need the UI to display a score variable, so the player can keep track of their points.

1. At the top of **GameManager.cs**, add "using TMPro;"
2. Declare a new ***public TextMeshProUGUI scoreText***, then assign that variable in the inspector
3. Create a new ***private int score*** variable and initialize it in ***Start()*** as ***score = 0;***
4. Also in **Start()**, set ***scoreText.text = "Score: " + score;***

- **New Concept:** Importing Libraries

```
using TMPro;
```

```
private int score;
public TextMeshProUGUI scoreText;

void Start() {
  StartCoroutine(SpawnTarget());
  score = 0;
  scoreText.text = "Score: " + score; }
```

Step 4: Create a new UpdateScore method

The score text displays the score variable perfectly, but it never gets updated. We need to write a new function that racks up points to display in the UI.

1. Create a new ***private void UpdateScore*** method that requires one ***int scoreToAdd*** parameter
2. Cut and paste ***scoreText.text = "Score: " + score;*** into the new method, then call ***UpdateScore(0)*** in ***Start()***
3. In ***UpdateScore()***, increment the score by adding ***score += scoreToAdd;***
4. Call ***UpdateScore(5)*** in the **spawnTarget()** function

- **New Concept:** Custom functions requiring parameters
- **Don't worry:** It doesn't make sense to add to score when spawned, this is just temporary

```
void Start() {
  ...scoreText.text = "Score: " + score;
  UpdateScore(0); }

IEnumerator SpawnTarget() {
  while (true) {  ... UpdateScore(5);  }

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

Step 5: Add score when targets are destroyed

Now that we have a method to update the score, we should call it in the target script whenever a target is destroyed.

1. In GameManager.cs, make the ***UpdateScore*** method ***public***
2. In Target.cs, create a reference to ***private GameManager gameManager;***
3. Initialize GameManager in ***Start()*** using the ***Find()*** method
4. When a target is **destroyed**, call ***UpdateScore(5);***, then **delete** the method call from SpawnTarget()

- **Tip:** Feel free to reference old code: We used script communication in Unit 3 to stop the game on GameOver
- **Warning:** If you try to call UpdateScore while it's private, it won't work

GameManager.cs

```
IEnumerator SpawnTarget() {
  while (true) {  ... UpdateScore(5);  }

private public void UpdateScore(int scoreToAdd) { ... }
```

Target.cs

```
private GameManager gameManager;

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

private void OnMouseDown() {
  ... gameManager.UpdateScore(5); }
```

Step 6: Assign a point value to each target

The score gets updated when targets are clicked, but we want to give each of the targets a different value. The good objects should vary in point value, and the bad object should subtract points.

1. In Target.cs, create a new ***public int pointValue*** variable
2. In each of the **Target prefab's** inspectors, set the **Point Value** to whatever they're worth, including the bad target's **negative value**
3. Add the new variable to ***UpdateScore(pointValue);***

- **Tip:** Here's the beauty of variables at work. Each target $ can have their own unique pointValue!

```
public int pointValue;

private void OnMouseDown() {
  Destroy(gameObject);
  gameManager.UpdateScore(5 pointValue); }
```

Step 7: Add a Particle explosion

The score is totally functional, but clicking targets is sort of… unsatisfying. To spice things up, let's add some explosive particles whenever a target gets clicked!

1. In Target.cs, add a new ***public ParticleSystem explosionParticle*** variable
2. For each of your target prefabs, assign a **particle prefab** from *Course Library > Particles* to the **Explosion Particle** variable
3. In the ***OnMouseDown()*** function, **instantiate** a new explosion prefab

```
public ParticleSystem explosionParticle;

private void OnMouseDown() {
  Destroy(gameObject);
  Instantiate(explosionParticle, transform.position,
  explosionParticle.transform.rotation);
  gameManager.UpdateScore(pointValue); }
```

Lesson Recap

New Functionality	• There is a UI element for score on the screen • The player's score is tracked and displayed by the score text when hit a target • There are particle explosions when the player gets an object
New Concepts and Skills	• TextMeshPro • Canvas • Anchor Points • Import Libraries • Custom methods with parameters • Calling methods from other scripts
Next Lesson	• We'll use some UI elements again - this time to tell the player the game is over and reset our game!