

# Section 4 – Quiz-I-Cool






# Important Topics



- ⚙️ Learn about UI
  - ⚙️ Canvas
  - ⚙️ Image
  - ⚙️ Text
  - ⚙️ Buttons
  - ⚙️ Sliders
- ⚙️ Storing Data 'ScriptableObject'
- ⚙️ List and Arrays

# Section Intro – Quiz-I-Cool

An isometric illustration of a game development scene. On the left, a large laptop displays the Unity logo. A person stands next to it, holding a smartphone. In the center, a large, dark, rectangular block represents a game engine or framework. To the right, a large game controller is shown, with a monitor displaying a game with three blue skulls. Several people are sitting on the floor, holding tablets and looking at the screen. In the foreground, a person is standing with their arms raised in a celebratory gesture. The background features several stacks of books or documents, and a few small, glowing cubes are scattered around.

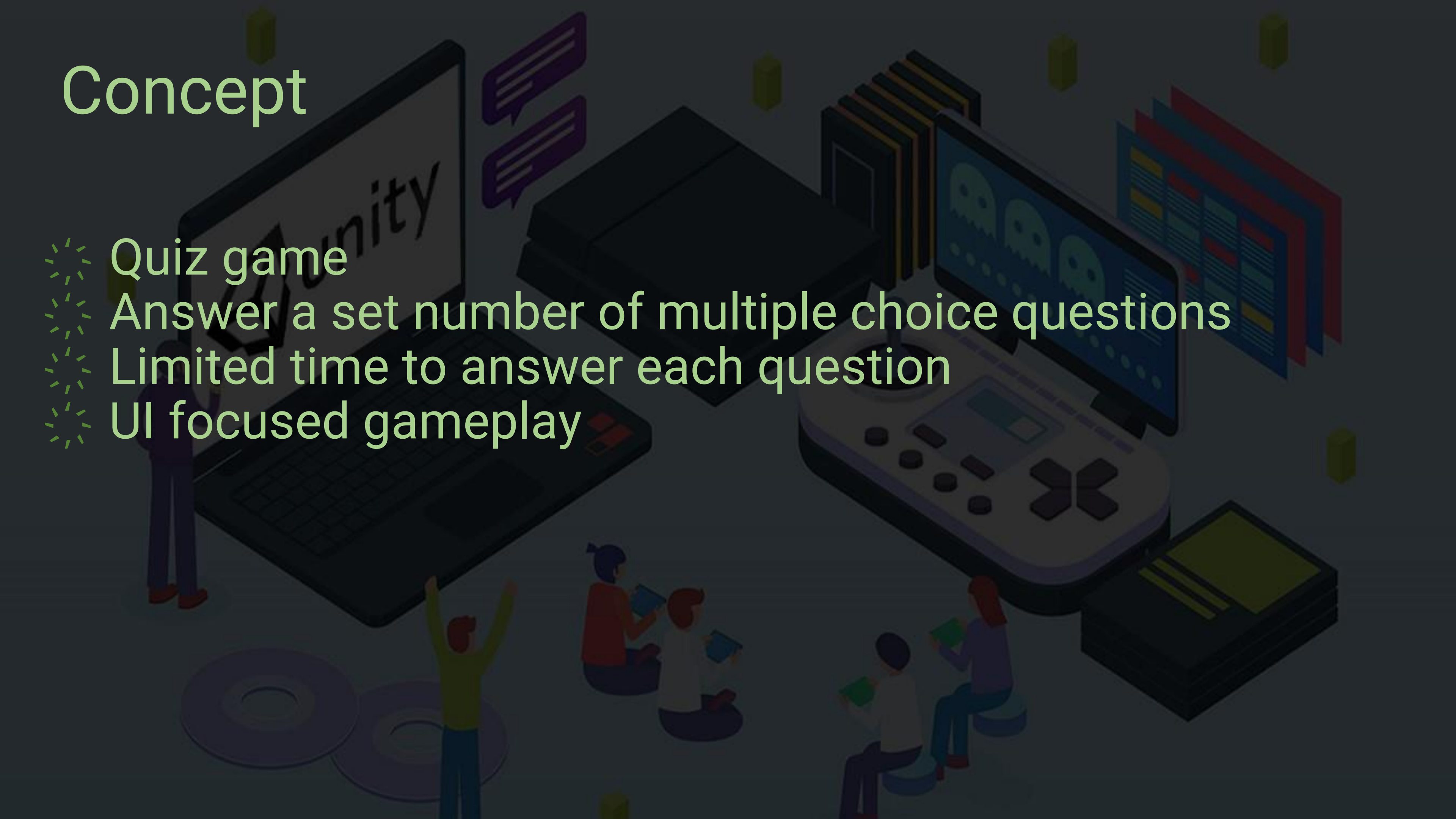
# Game Design – Quiz-I-Cool

An isometric illustration depicting various elements of game design and development. On the left, a large laptop displays the Unity logo. A person stands next to it, holding a smartphone. In the center, there are stacks of books or documents. To the right, a large game controller is shown with a screen displaying three skulls. Further right, there are more stacks of books or documents. In the foreground, several people are sitting on the floor, some holding tablets or books. The background is dark with some floating yellow cubes.



# Concept

- ⌘ Quiz game
- ⌘ Answer a set number of multiple choice questions
- ⌘ Limited time to answer each question
- ⌘ UI focused gameplay



# Gameplay Overview Screen

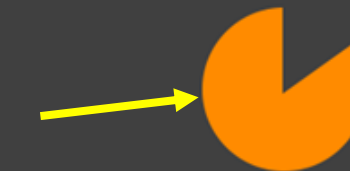
Title Text

QUIZ-I-COOL

Score: 100%

Score Text

Timer  
Image



This box will contain the question text

Question Text

Answer Buttons

Answer 1

Answer 2

Answer 3

Answer 4

Progress Slider



# Gameplay Overview Screen

## QUIZ-I-COOL

Score: 80%



Sorry, the correct answer was [Answer 3]

Confirmation Text

Answer 1

Answer 2

Answer 3

Answer 4

Highlight Correct Answer



# Gameplay Overview Screen

## QUIZ-I-COOL

Win Message →

Congratulations!  
You scored 85%

Play Again?

↗  
Replay Button



# Game Mechanics We Need

- ⦿ Mechanism to store and retrieve questions
- ⦿ Buttons to select an answers
- ⦿ Timer to put some pressure on the player
- ⦿ Progress bar to show how many questions remain
- ⦿ Scoring to show the player how well they did
- ⦿ A way to restart the game when the quiz ends

# Game Design

**Player Experience:**

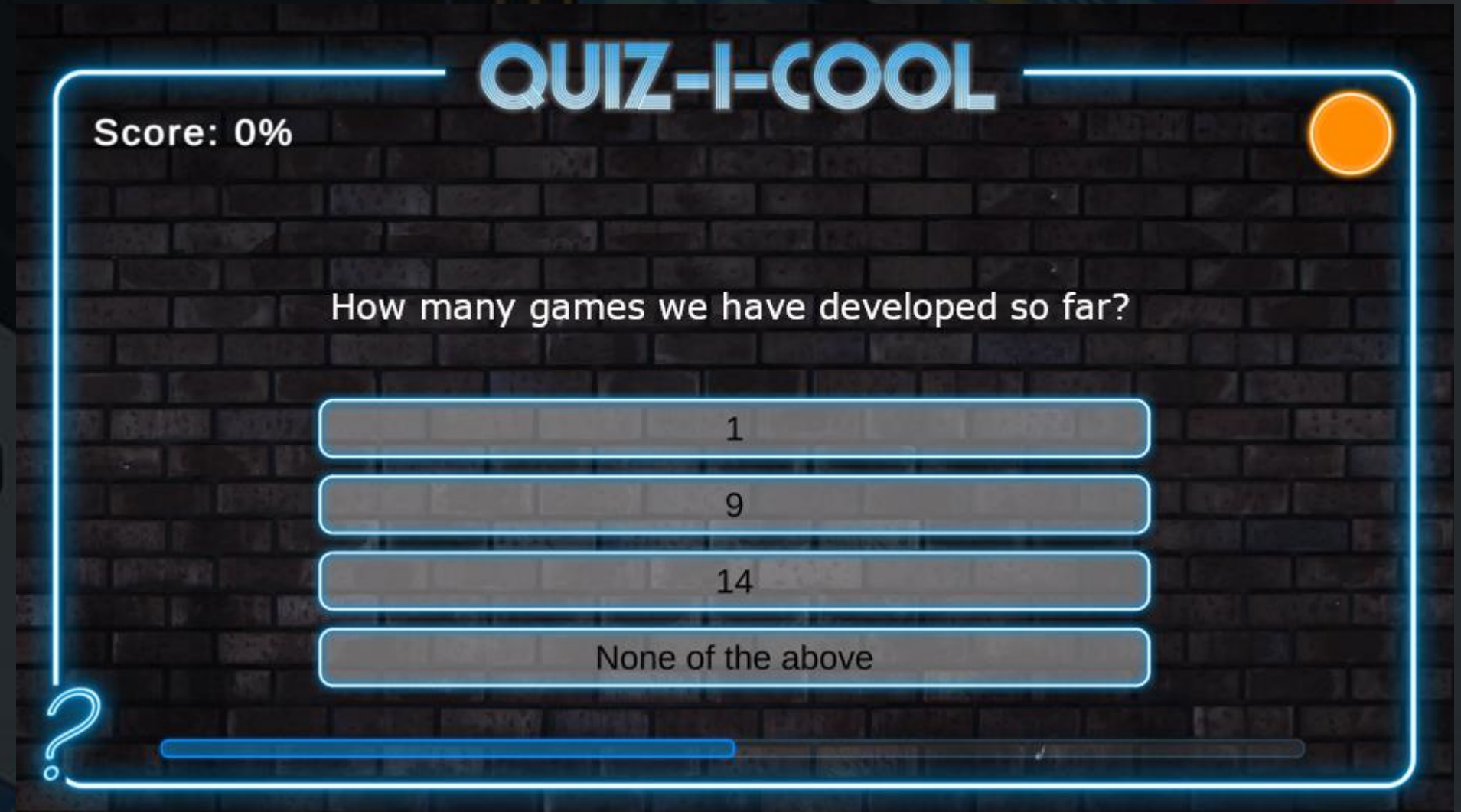
Knowledgeable /  
Intelligent

**Core Mechanic:**

Test your knowledge

**Game Loop:**

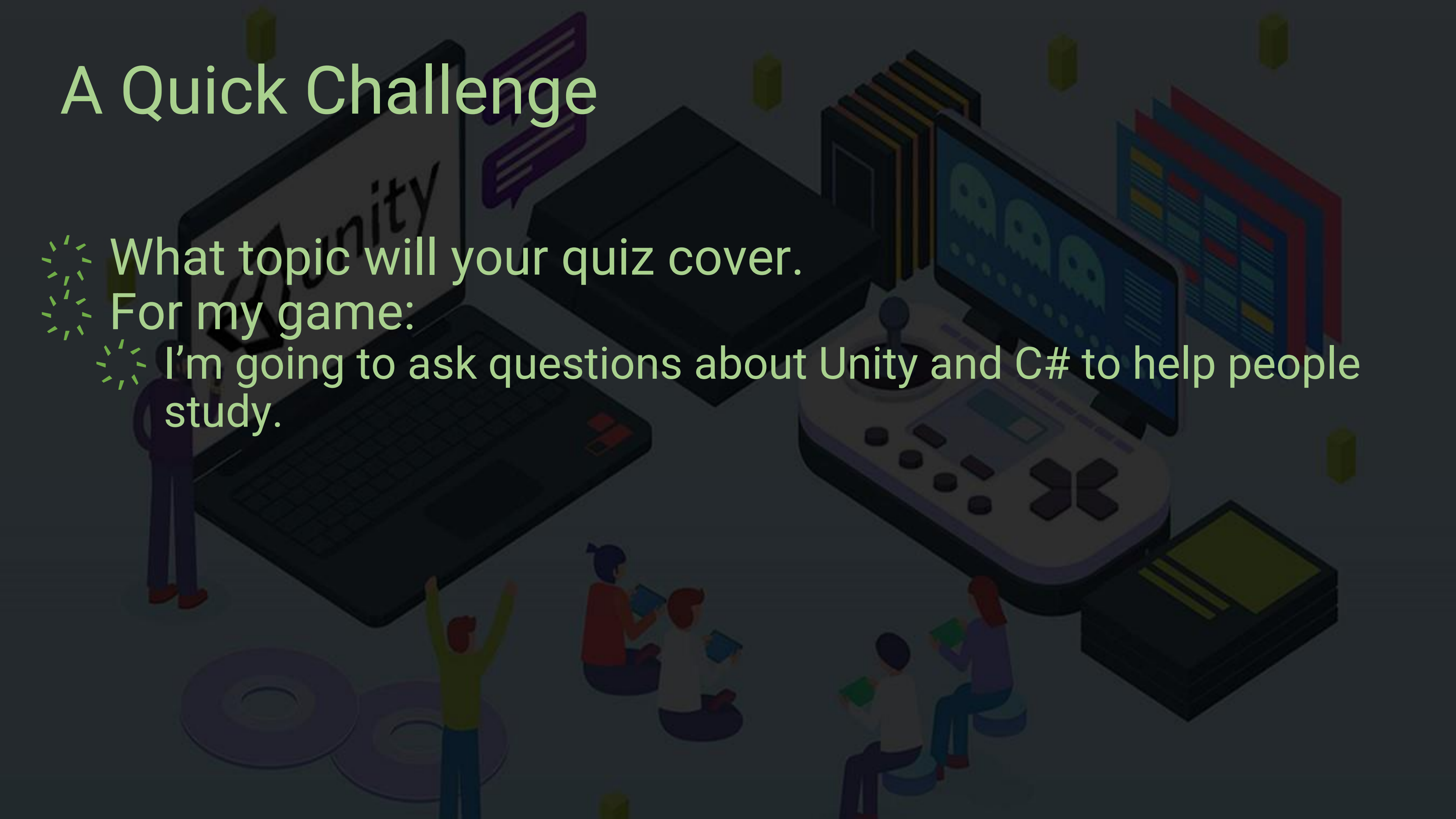
Answer a set number  
questions on a topic  
within the given time





# A Quick Challenge

- What topic will your quiz cover.
- For my game:
  - I'm going to ask questions about Unity and C# to help people study.





# UI Canvas



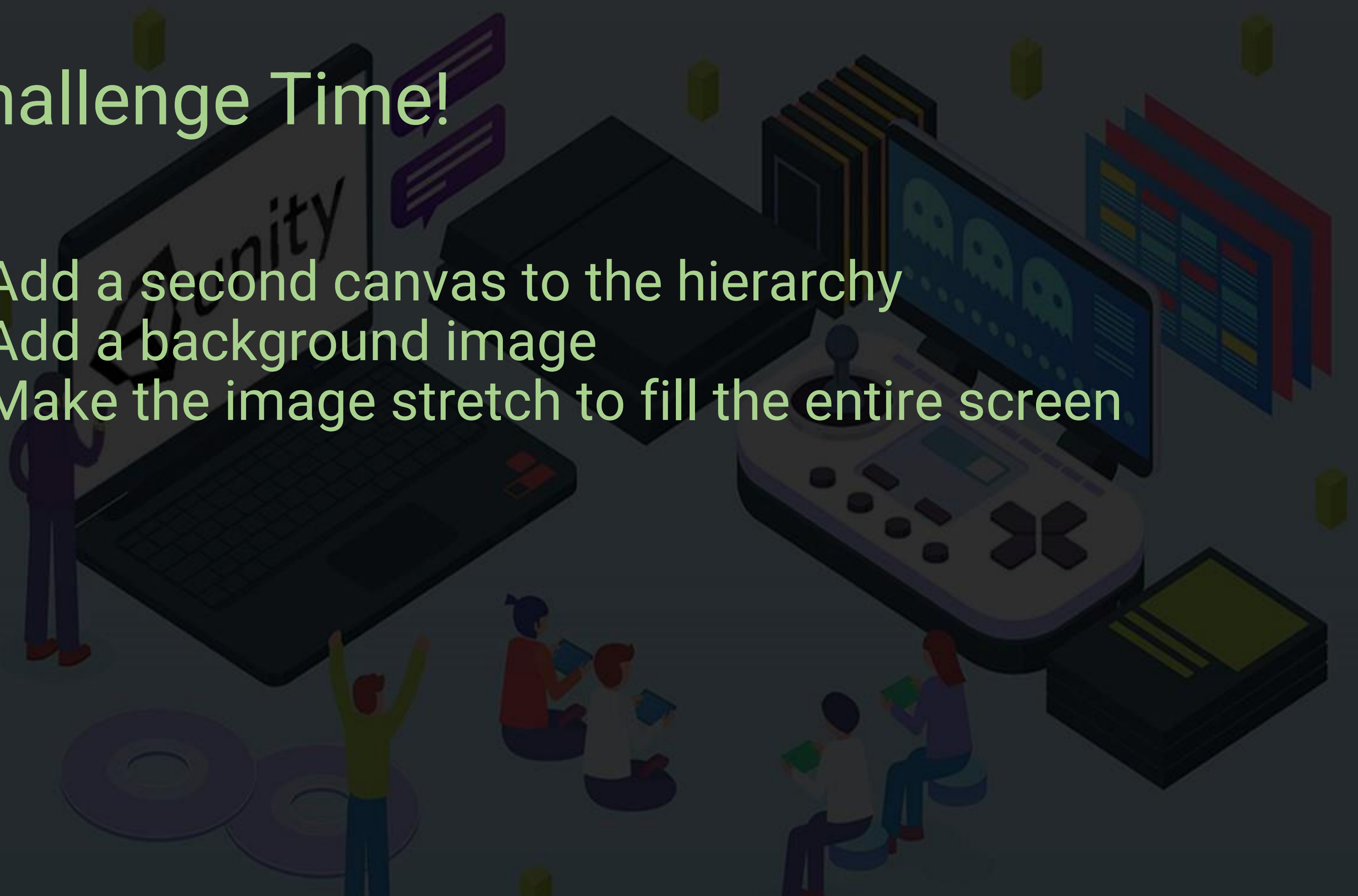
# UI Canvas

- ⌘ UI = User Interface
- ⌘ Text, buttons, sliders, menus, etc.
- ⌘ UI elements live on the “Canvas”
- ⌘ The canvas generally exists in “Screen Space” and is mostly separate from the game world
- ⌘ You can have multiple canvases



# Challenge Time!

- ✦ Add a second canvas to the hierarchy
- ✦ Add a background image
- ✦ Make the image stretch to fill the entire screen





# TextMesh Pro



# Find a Font!

- ☀ Do some font shopping
  - ☀ [Browse Fonts - Google Fonts](#)
  - ☀ [DaFont - Download fonts](#)
- ☀ Find a free font and add it to your asset folder
- ☀ Double check the usage rights!



# More Text!

- ✱ Fine tune your title text
- ✱ Play around with TMPro to see what you can make
- ✱ Add a new “QuizCanvas” to the hierarchy
- ✱ Add a TextMesh Pro element for the question text



# Button Layout



# Set Up Your Buttons

- ✱ Add an image to your buttons
- ✱ Remember to slice your sprites!
- ✱ Organize your buttons using a layout group
- ✱ Change the spacing, padding, and alignment

# Scriptable Objects






# What is a Scriptable Object?

- ✧ It's just a data container!
- ✧ Keeps the data out of our scripts
- ✧ Help us save memory by storing data in one place
- ✧ They don't need to be attached to game objects
- ✧ They're lightweight and convenient
- ✧ Act as a template for consistency

# Examples

- Weapon stats in an RPG
- Card data in a CCG
- We'll be using them to store
  - Question text
  - Possible answers
  - Correct answer

 Question 2 (Question SO) Open

Script QuestionSO

Question

If all other conditions are false, which conditional statement would you use as a catch all?

▼ Answers 4

Element 0	if
Element 1	else if
Element 2	else
Element 3	when

+ -

Correct Answer Index 2

# Structure

## Our Code

Quiz.cs

GetQuestionData()

Question 3

DisplayQuestion()

CheckAnswer()

## Scriptable Objects

Question 1

Question 2

Question 3

Question 4





# Getter Methods



# Getter Methods

- ⌘ Gives a script read-only access to a private variable
- ⌘ Protects the contents of a private variable

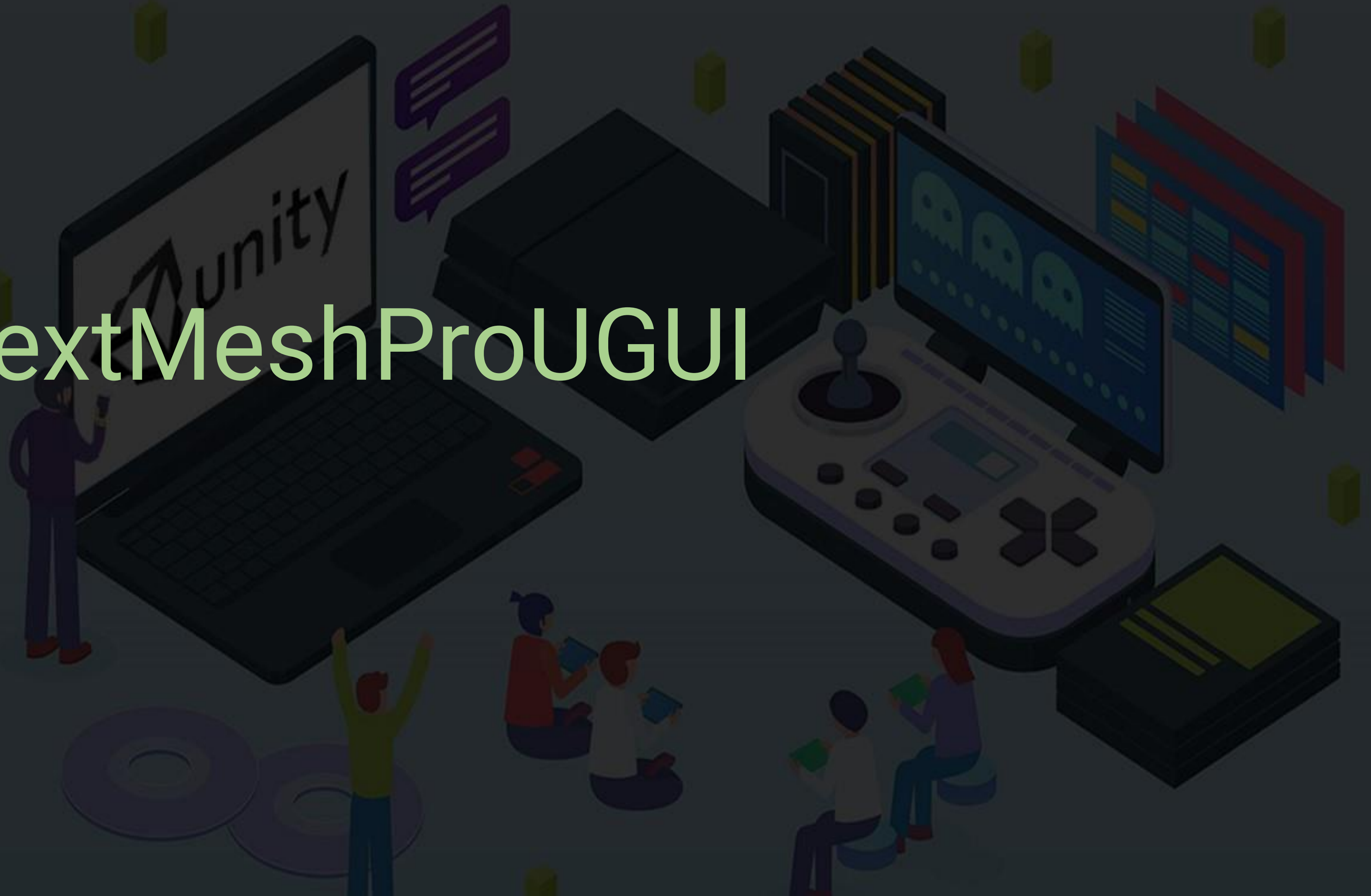




# Challenge

- ✱ Create two getter methods called;
  - ✱ GetCorrectAnswerIndex()
  - ✱ GetAnswer(int index)

# TextMeshProUGUI





# For Loops



# Challenge

- ⚡ Change the text displayed on the button to the first answer stored in our QuestionSO.

Remember:

Our scriptable objects contain the getter method:

```
GetAnswer(int index)
```



# Swapping Sprites



# Challenge

- ⦿ Change the question text to display the correct answer.
- ⦿ Change the image on the button that contains the correct answer.

Remember:

The correct answer index is stored in the scriptable object.



# Button States



# Game Flow

Display New Question

Turn buttons on

Answer Question

Turn buttons off





# Challenge

✱ Write SetDefaultButtonSprites()

Logic:

- ✱ Loop through all the answer buttons
- ✱ Get the Image component on each button
- ✱ Change the sprite back to the default sprite

# Simple Timers





# Timer

- ⌚ What state is the game in?
  - ⌚ answering question or showing answer
- ⌚ Has the timer run down?
- ⌚ Change the fill amount of the timer image
- ⌚ When time runs out, change the state of the game

# Challenge

- ⚡ Change the state of `isAnsweringQuestion` when the `timerValue` reaches zero
- ⚡ Set the `timerValue` to match the state that we are in.



# Connecting the Timer



# Challenge

⚡ Change the fill amount of the timer image every frame

Hint:

We've already worked out the fill fraction in Timer.cs





# Lists



# What is a List?

- ⌘ They're kind of like arrays!
- ⌘ A grouping of multiple variables of the same type
- ⌘ Each item stored in an List is called an 'element'
- ⌘ Each element can be accessed by its index number
- ⌘ Counting starts at zero!
- ⌘ They're mutable - meaning we can change their size!



# Syntax

## Array

```
Int[] oddNumbers = new int[5]
```

## List

```
List<int> oddNumbers = new List<int>()
```

# Useful Methods & Properties

Check item count:

`List.Count`

Check if item exists:

`List.Contains(3)`

Add an item:

`List.Add(3)`

Remove an item:

`List.Remove(3)`

Remove item at index:

`List.RemoveAt(0)`

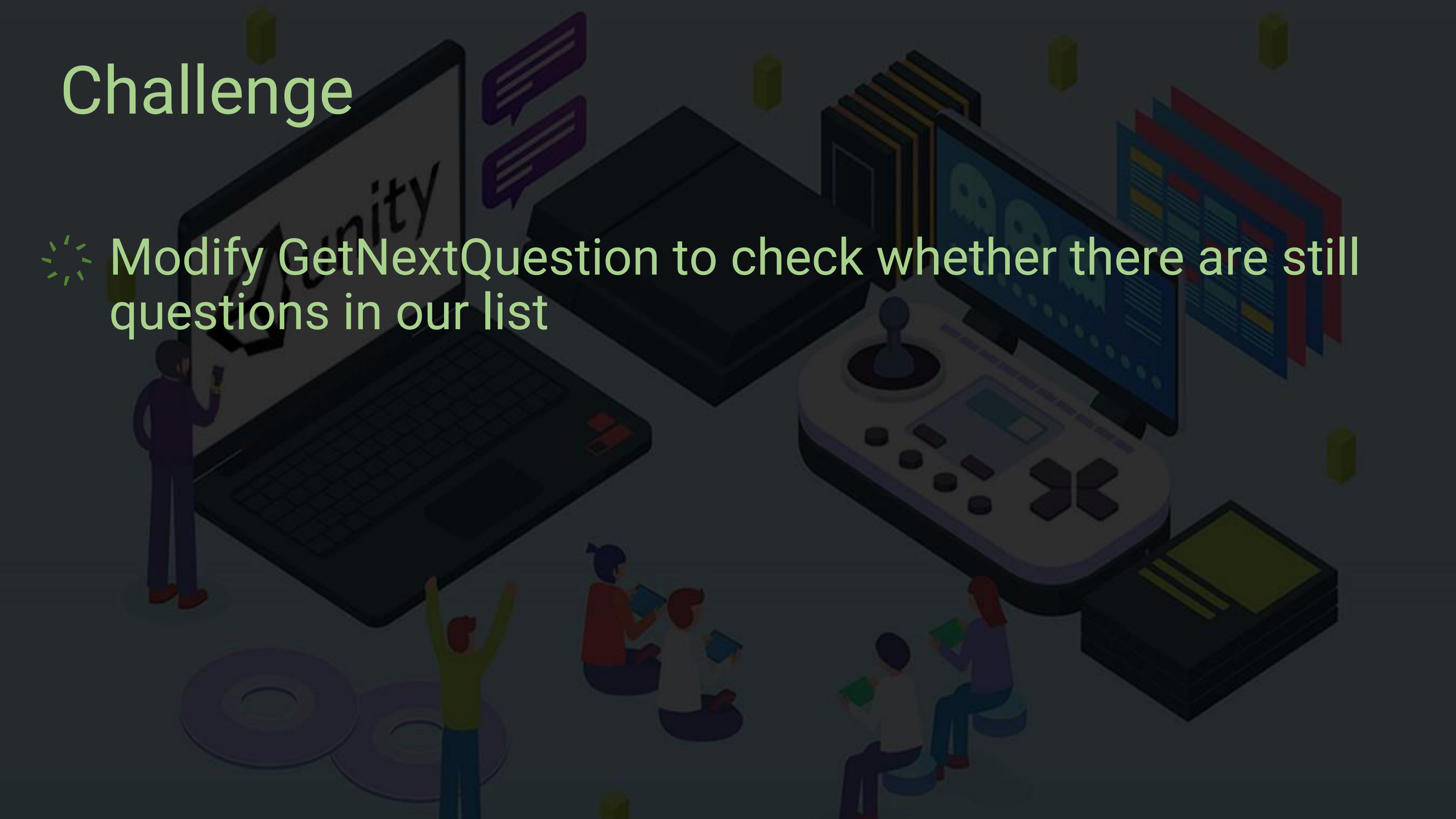
Clear the list:

`List.Clear()`



# Challenge

- ⚙️ Modify GetNextQuestion to check whether there are still questions in our list



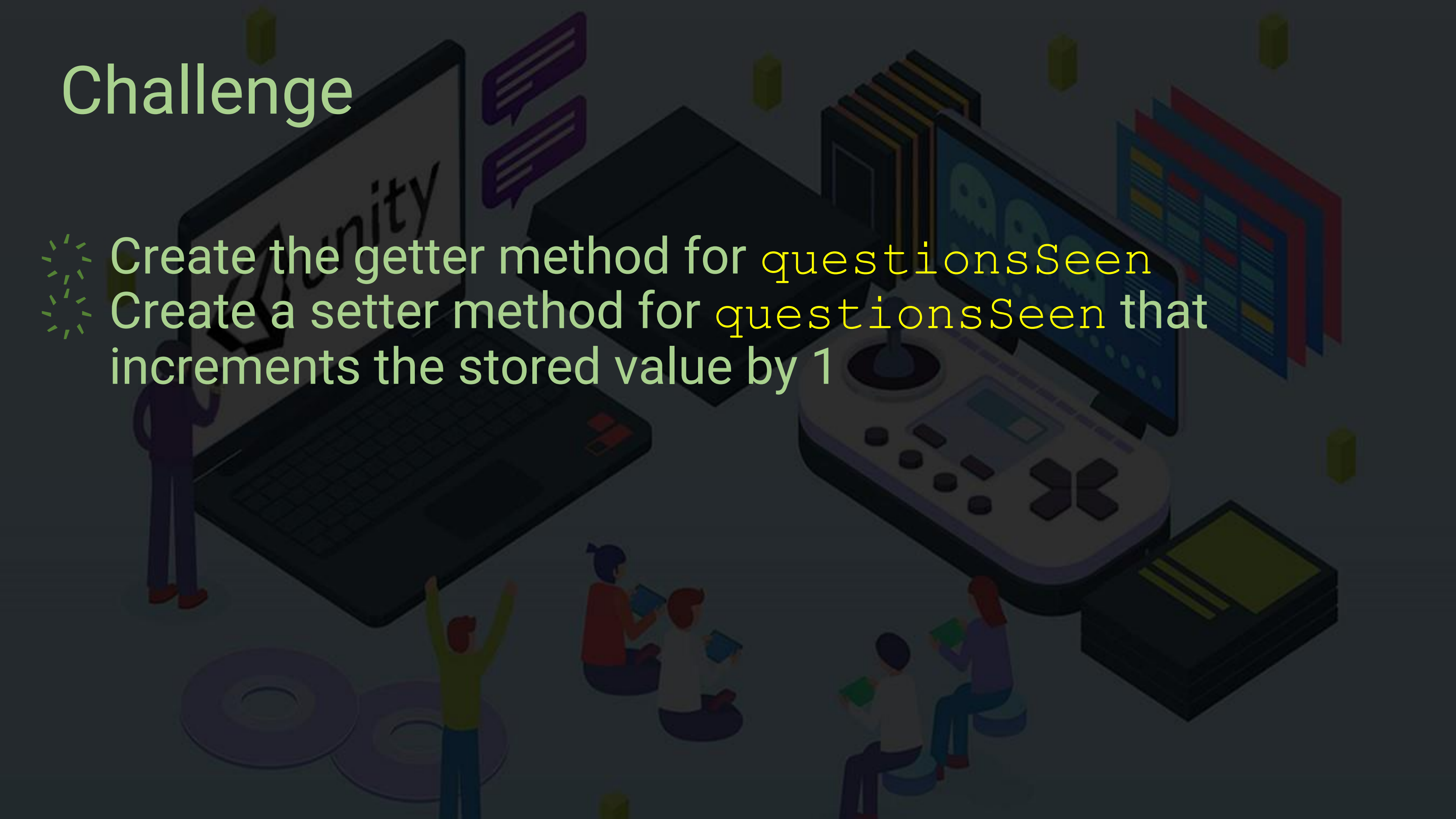
# Keeping Score





# Challenge

- ✦ Create the getter method for `questionsSeen`
- ✦ Create a setter method for `questionsSeen` that increments the stored value by 1



# Sliders





# Challenge

- ☼ Set up your progress bar!
- ☼ Resize and position your slider on the canvas
- ☼ Change the background and fill colors / sprites
- ☼ Set up (or disable) the handle



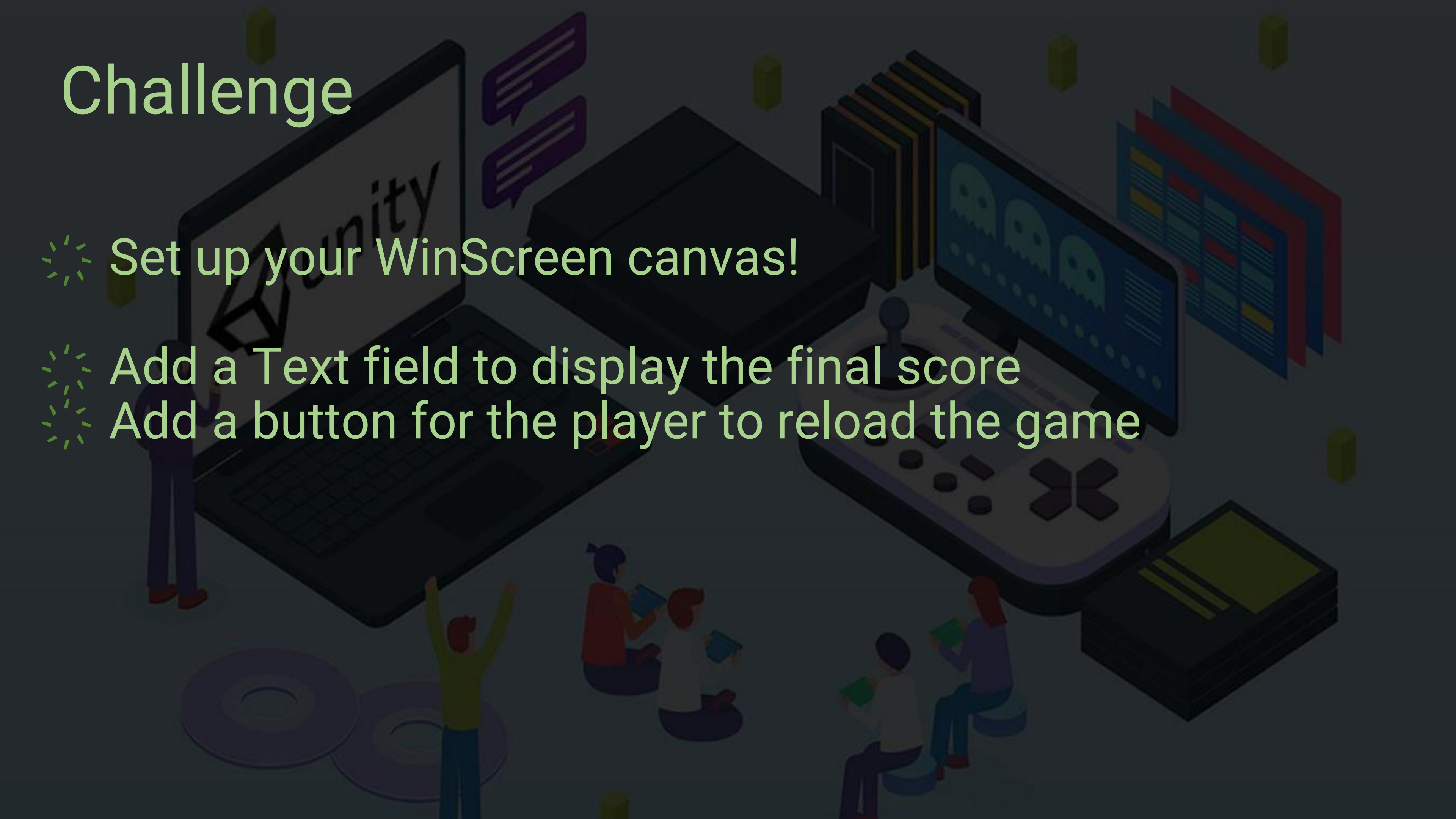
# End Screen





# Challenge

- ⚙️ Set up your WinScreen canvas!
- ⚙️ Add a Text field to display the final score
- ⚙️ Add a button for the player to reload the game



# Game Manager





# Challenge

- ⦿ Check if the game has been completed
- ⦿ Disable the QuizCanvas
- ⦿ Enable the WinCanvas



# References

- ⚙ [Basic Layout | Unity UI | 1.0.0 \(unity3d.com\)](#)
- ⚙ [DaFont - Download fonts](#)
- ⚙ [Button | Unity UI | 1.0.0 \(unity3d.com\)](#)
- ⚙ [Unity - Manual: ScriptableObject \(unity3d.com\)](#)
- ⚙ [Arrays - C# Programming Guide | Microsoft Learn](#)
- ⚙ [Iteration statements - C# reference | Microsoft Learn](#)
- ⚙ [Unity - Scripting API: UI.Image.sprite \(unity3d.com\)](#)
- ⚙ [Unity - Scripting API: Button \(unity3d.com\)](#)
- ⚙ [Unity - Scripting API: UI.Image.fillAmount \(unity3d.com\)](#)



# References

- ⚙️ [List<T> Class \(System.Collections.Generic\) | Microsoft Learn](#)
- ⚙️ [Casting and type conversions - C# Programming Guide | Microsoft Learn](#)
- ⚙️ [Unity - Scripting API: Mathf.RoundToInt \(unity3d.com\)](#)
- ⚙️ [Unity - Scripting API: Slider \(unity3d.com\)](#)
- ⚙️ [Unity - Scripting API: SceneManager.SceneManager.LoadScene \(unity3d.com\)](#)
- ⚙️ [Unity - Manual: Script Execution Order settings \(unity3d.com\)](#)