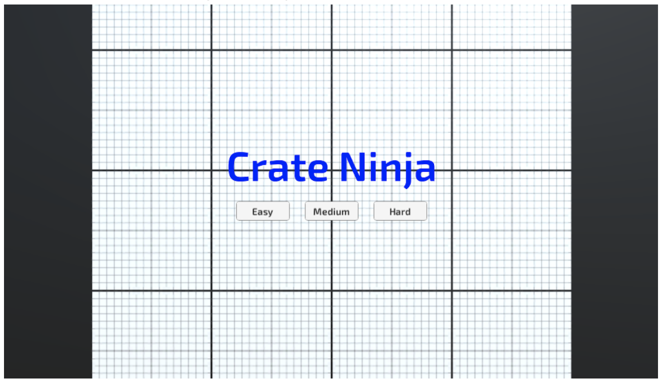
5.4 What's the Difficulty?

Steps:

Example of project by end of lesson



Length: 60 minutes

Overview: It's time for the final lesson! To finish our game, we will add a Menu and Title Screen of sorts. You will create your own title, and style the text to make it look nice. You will create three new buttons that set the difficulty of the game. The higher the difficulty, the faster the targets spawn!

Project Outcome: Starting the game will open to a beautiful menu, with the title displayed prominently and three difficulty buttons resting at the bottom of the screen. Each difficulty will affect the spawn rate of the targets, increasing the skill required to stop "good" targets from falling.

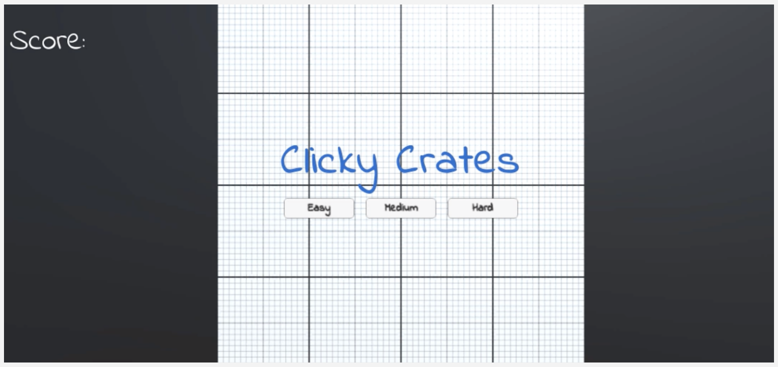
Learning Objectives: By the end of this lesson, you will be able to:
- Store UI elements in a parent object to create Menus, UI, or HUD
- Add listeners to detect when a UI Button has been clicked
- Set difficulty by passing parameters into game functions like SpawnRate

Step 1: Create Title text and 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



Step 2: Add a DifficultyButton 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>(); }
```

Step 3: Call SetDifficulty on 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

- **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

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

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

Step 4: Make your buttons 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();
}
```

Step 5: Deactivate Title Screen 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); }
```

Step 6: Use a parameter to 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; }
```

Lesson Recap

New Functionality	● Title screen that lets the user start the game ● Difficulty selection that affects spawn rate
New Concepts and Skills	● AddListener() ● Passing parameters between scripts ● Divide/Assign (/=) operator ● Grouping child objects