Unit 2

https://learn.unity.com/tutorial/unit-2-introduction?uv=2021.3&pathwayId=5f7e17e1edbc2a5ec21a20af&missionId=5f71fe63edbc2a00200e9de0&projectId=5cdcc312edbc2a24a41671e6#5d1ba822edbc2a002175788d

- Introduction
- Lesson 2.1 - Player Positioning
- 1.Create a new Project for Prototype 2
- 2.Add the 3d assets + first few lines of code
- 3. Instantiation
- Random Animal Stampeed
- Collision Decisions + GAME OVER

Introduction

we will make a top down prototype where you shoot things that come after you, kindof like galiga except in 3d and not at all like galiga XD

Lesson 2.1 - Player Positioning

Summary

Overview:

You will begin this unit by creating a new project for your second Prototype and getting basic player movement working. You will first choose which character you would like, which types of animals you would like to interact with, and which food you would like to feed those animals. You will give the player basic side-to-side movement just like you did in Prototype 1, but then you will use if-then statements to keep the Player in bounds.

Project Outcome:

The player will be able to move left and right on the screen based on the user's left and right key presses, but will not be able to leave the play area on either side.

Materials

Prototype 2 - Starter Files.zip

Select your Unity version

2021.1 - 2021.3

1.Create a new Project for Prototype 2

- Open **Unity Hub** and create an empty “Prototype 2” project in your course directory on the correct Unity version. If you forget how to do this, refer to the instructions in Lesson 1.1 - Step 1

- Click to download the Prototype 2 Starter Files, **extract** the compressed folder, and then **import** the .unitypackage into your project. If you forget how to do this, refer to the instructions in Lesson 1.1 - Step 2

- From the Project window, open the **Prototype 2** scene and **delete** the SampleScene

- In the top-right of the Unity Editor, change your **Layout** from Default to your custom layout

2.Add the 3d assets + first few lines of code

1. 1. If you want, drag a different **material** from *Course Library > Materials* onto the Ground object
   2. Drag 1 **Human**, 3 **Animals**, and 1 **Food** object into the Hierarchy
   3. Rename the character “Player”, then **reposition** the animals and food so you can see them
   4. Adjust the XYZ **scale** of the food so you can easily see it from above

- **Attach** the script to the Player and open it
- In your **Assets** folder, create a “Scripts” folder, and a “PlayerController” script inside
- At the top of PlayerController.cs, declare a new
- [SerializeField]
- ***private float horizontalInput***
- In ***Update()***, set ***horizontalInput = Input.GetAxis(“Horizontal”)***, then test to make sure it works in the inspector
-

```
using UnityEngine;

public class Movement : MonoBehaviour
{
    [SerializeField]
    private float horizontalInput;
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        horizontalInput = Input.GetAxis("Horizontal");
    }
}
```

4.Move the player left-to-right

- here we can get our input with this basic code, we use [SerializeField] for sanitization and allow us to see the value in the inspection panel to make sure we didn't fck up lol
- Declare a new ***public float speed = 10.0f;***
- ***here's my code***

```
using UnityEngine;

public class Movement : MonoBehaviour
{
    [SerializeField]
    private float horizontalInput;
    [SerializeField]
    private float speed = 30.0f;
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        horizontalInput = Input.GetAxis("Horizontal");
        transform.Translate(Vector3.right * horizontalInput * Time.deltaTime * speed);
    }
}
```

5+6. Keep the player inbounds

- In ***Update()***, write an **if-statement** checking if the player’s left X position is **less than** a certain value
- In the if-statement, set the player’s position to its current position, but with a **fixed X location**

- Repeat this process for the **right side** of the screen
- Declare new ***xRange*** variable, then replace the hardcoded values with them
- Add **comments** to your code or not lol

```
using UnityEngine;

public class Movement : MonoBehaviour
{
    [SerializeField]
    private float horizontalInput;
    [SerializeField]
    private float speed = 10.0f;
    [SerializeField]
    private float xRange = 20.0f;
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        if (transform.position.x < -xRange)
        {
            transform.position = new Vector3(-xRange, transform.position.y ,
transform.position.z);
        }
        if (transform.position.x > xRange)
        {
            transform.position = new Vector3(xRange, transform.position.y,
transform.position.z);
        }
        horizontalInput = Input.GetAxis("Horizontal");
        transform.Translate(Vector3.right * horizontalInput * Time.deltaTime *
speed);
    }
}
```

New Functionality

The player can move left and right based on the user’s left and right key presses

The player will not be able to leave the play area on either side
New Concepts & Skills
Adjust object scale
If-statements
Greater/Less than operators
Next Lesson, We’ll learn how to create and throw endless amounts of food to feed our animals!

now use chat gpt to teach me thing and fix a bug that can occur when you move fast

float clampedX = Mathf.Clamp(desiredpoz.x, -Xrange, Xrange) become the min/max

```
using UnityEngine;
//needed a bit of editing coz cgpt3.5 is stooooooooopid
public class Movement : MonoBehaviour
{
    [SerializeField]
    private float speed = 10.0f;
    [SerializeField]
    private float xRange = 10.0f;
    [SerializeField]
    private float horizontalInput;
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        horizontalInput = Input.GetAxis("Horizontal");

        // Calculate the desired position based on the input
        Vector3 desiredPosition = transform.position + Vector3.right * -horizontalInput *
Time.deltaTime * speed;

        // Clamp the desired position within the x range
        float clampedX = Mathf.Clamp(desiredPosition.x, -xRange, xRange);
```

```
        desiredPosition = new Vector3(clampedX, desiredPosition.y, desiredPosition.z);
        // Move the object to the clamped position
        transform.position = desiredPosition;
    }
}
```

3. Instantiation

The first thing we must do is give the projectile some forward movement so it can zip across the scene when it's launched by the player.

1.Make the projectile fly forwards

- Create a new "MoveForward" script, **attach** it to the food object, then open it
- Declare a new ***public float speed*** variable***;***
- In ***Update()***, add ***transform.Translate(Vector3.forward * Time.deltaTime * speed);*** , then **save**
- In the **Inspector**, set the projectile's **speed** variable, then test

```
using UnityEngine;

public class Pizza_gun : MonoBehaviour
{
    public GameObject Ammo;

    public float Weapon_Bullet_Projectile_Speed = 60.0f;
    // Start is called before the first frame update
    void Start()
    {
        Ammo = GetComponent<GameObject>();
    }

    // Update is called once per frame
    void Update()
    {
        transform.Translate(Vector3.forward * Time.deltaTime *
Weapon_Bullet_Projectile_Speed);
    }
}
```

- Create a new "Prefabs" folder, drag your food into it, and choose **Original Prefab**
- In PlayerController.cs, declare a new ***public GameObject projectilePrefab;*** variable

- **Select** the Player in the hierarchy, then **drag** the object from your Prefabs folder onto the new **Projectile Prefab box** in the inspector
- Try **dragging** the projectile into the scene at runtime to make sure they fly
- I added the ammo = get component thing for maybe powerups
- In PlayerController.cs, in ***Update()***, add an **if-statement** checking for a spacebar press: ***if (Input.GetKeyDown(KeyCode.Space)) {}***
- Inside the if-statement, add a comment saying that you should ***// Launch a projectile from the player***
- Inside the if-statement, use the **Instantiate** method to spawn a projectile at the player's location with the prefab's rotation
- **Select** all three animals in the hierarchy and *Add Component* > ***Move Forward*****Rotate** all animals on the Y axis by **180 degrees** to face down
- Edit their **speed values** and **test** to see how it looks
- Drag all three animals into the **Prefabs folder**, choosing "Original Prefab"
- **Test** by dragging prefabs into scene view during gameplay
-

```
using UnityEngine;


public class Movement : MonoBehaviour
{
    [SerializeField]
    private float speed = 10.0f;
    [SerializeField]
    private float xRange = 10.0f;
    [SerializeField]
    private float horizontalInput;
    [SerializeField]
    public GameObject ProjectilePrefab;
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {



        horizontalInput = Input.GetAxis("Horizontal");
        // Calculate the desired position based on the input
        Vector3 desiredPosition = transform.position + Vector3.right * -horizontalInput *
```

```
Time.deltaTime * speed;
        // Clamp the desired position within the x range
        float clampedX = Mathf.Clamp(desiredPosition.x, -xRange, xRange);
        desiredPosition = new Vector3(clampedX, desiredPosition.y, desiredPosition.z);
        // Move the object to the clamped position
        transform.position = desiredPosition;
        if (Input.GetKeyDown(KeyCode.Space))
        {
            // shoot pizza
            Instantiate(ProjectilePrefab, transform.position,
ProjectilePrefab.transform.rotation);
        }
    }
}
```

5.Make animals into prefabs

- **Rotate** all animals on the Y axis by **180 degrees** to face down/ towards the player
- **Select** all three animals in the hierarchy and *Add Component* > ***Move Forward***
- Edit their **speed values** and **test** to see how it looks
- Drag all three animals into the **Prefabs folder**, choosing “Original Prefab”
- **Test** by dragging prefabs into scene view during gameplay

6.Destroy projectiles offscreen

- Create “DestroyOutOfBounds” script and apply it to the **projectile**
- Add a new ***private float topBound*** variable and initialize it ***= 30;***
- Write code to destroy if out of top bounds ***if (transform.position.z > topBound) { Destroy(gameObject); }***
- In the Inspector **Overrides** drop-down, click **Apply all** to apply it to prefab

7. Destroy animals offscreen

- Create **else-if statement** to check if objects are beneath ***lowerBound***: ***else if (transform.position.z < lowerBound)***
- **Apply** the script to all of the animals, then **Override** the prefabs
-

8.Lesson Recap

New Functionality

- The player can press the Spacebar to launch a projectile prefab,

Projectile and Animals are removed from the scene if they leave the screen

New Concepts & Skills

- Create Prefabs
- Override Prefabs
- Test for Key presses
- Instantiate objects
- Destroy objects
- Else-if statements

Next Lesson

- Instead of dropping all these animal prefabs onto the scene, we'll create a herd of animals roaming the plain!

(

Lesson 2.2 - Food Flight
Skills practiced:
Absolute Beginner Code Comprehension
Interpret simple code
Improve simple code using the features of an IDE
Absolute Beginner Application Scripting
Use common logic structures to control the execution of code.
Write code that utilizes the various Unity APIs
Implement appropriate data types
Write code that integrates into an existing system
Implement a code style that is efficient and easy to read
Prototype new concepts
)

Random Animal Stampeed

1. Create a spawn manager

- In the Hierarchy, create an **Empty object** called “SpawnManager”
- Create a new script called “SpawnManager”, attach it to the **Spawn Manager**, and open it
- Declare new ***public GameObject[ ] animalPrefabs;***
- In the Inspector, change the **Array size** to match your animal count, then **assign** your animals by **dragging** them from the Project window into the empty slots **Note:** Make sure you drag them from the **Project** window; not the Hierarchy! If you're going to spawn objects, you need to make sure you're using Prefabs, which are stored in the Project window.

2. Spawn an animal if S is pressed

- In ***Update()***, write an if-then statement to **instantiate** a new animal prefab at the top of the screen if **S** is pressed
- Declare a new ***public int animalIndex*** and incorporate it in the **Instantiate** call, then test editing the value in the Inspector

3. Spawn random animals from an array

- In the if-statement checking if S is pressed, generate a random ***int animalIndex*** between 0 and the length of the array
- Remove the global ***animalIndex*** variable, since it is only needed locally in the **if-statement**

4. Randomize the spawn location

- **Replace** the X value for the Vector3 with ***Random.Range(-20, 20)***, then test
- Within the **if-statement**, make a new local ***Vector3 spawnPos*** variable
- At the top of the class, create ***private float*** variables for ***spawnRangeX*** and ***spawnPosZ***

5. Change the perspective of the camera

- Toggle between **Perspective** and **Isometric** view in the Scene view to appreciate the difference
- Select the **camera** and change the **Projection** from “Perspective” to “Orthographic”

6. Lesson Recap

New Functionality

- ○ The player can press the S to spawn an animal
  ○ Animal selection and spawn location are randomized
  ○ Camera projection (perspective/orthographic) selected

New Concepts & Skills

  ○ Spawn Manager
  ○ Arrays
  ○ Keycodes
  ○ Random generation
  ○ Local vs Global variables
  ○ Perspective vs Isometric projections

Next Lesson

- Using collisions to feed our animals!

Collision Decisions + GAME OVER

1. Make a new method to spawn animals

- In **SpawnManager.cs**, create a new ***void SpawnRandomAnimal() {}*** function beneath ***Update()***
- Cut and paste the code from the **if-then statement** to the **new function**
- Call ***SpawnRandomAnimal();*** if **S** is pressed

2. Spawn the animals at timed intervals

- In ***Start()***, use ***InvokeRepeating*** to spawn the animals based on an interval, then **test**.
- Remove the **if-then statement** that tests for **S** being pressed
- Declare new ***private startDelay*** and ***spawnInterval*** variables then playtest and tweak variable values

3. Add collider and trigger components

- Double-click on one of the **animal prefabs**, then *Add Component > Box Collider*
- Click **Edit Collider**, then **drag** the collider handles to encompass the object
- Check the "**Is Trigger**" checkbox
- Repeat this process for each of the **animals** and the **projectile**
- Add a **RigidBody component** to the projectile and uncheck "use gravity"

4. Destroy objects on collision

- Create a new **DetectCollisions.cs** script, add it to each animal prefab, then **open** it
- Before the final **}** add an ***OnTriggerEnter*** function using **autocomplete**

- In ***OnTriggerEnter***, put ***Destroy(gameObject);***, then test
- In ***OnTriggerEnter***, put ***Destroy(other.gameObject);***

5. Trigger a “Game Over” message

- In DestroyOutOfBounds.cs, in the **else-if condition** that checks if the animals reach the bottom of the screen, add a Game Over messsage: ***Debug.Log(“Game Over!”)***
- Clean up your code with **comments**
- If using Visual Studio, Click *Edit* > *Advanced* > ***Format document*** to fix any indentation issues (On a **Mac**, click *Edit* > *Format* > *Format Document)*

6. Lesson Recap

New Functionality

- Animals spawn on a timed interval and walk down the screen
- When animals get past the player, it triggers a “Game Over” message
- If a projectile collides with an animal, both objects are removed

New Concepts & Skills

- Create custom methods/functions
- InvokeRepeating() to repeat code
- Colliders and Triggers
- Override functions
- Log Debug messages to console

Pizza_gun

```
using UnityEngine;

public class Pizza_gun : MonoBehaviour
{
    [SerializeField]
    public float Weapon_Bullet_Projectile_Speed = 60.0f;
    [SerializeField]
    private float topBound = -60;

    // Update is called once per frame
    void Update()
    {
        transform.Translate(Vector3.forward * Time.deltaTime *
Weapon_Bullet_Projectile_Speed);
```

```
       if (transform.position.z < topBound)
        {
            Destroy(gameObject);
        }

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

Spawn_manager

- ```
  using System.Collections;
  using System.Collections.Generic;
  using UnityEngine;

  public class SpawnManager : MonoBehaviour
  {
      public GameObject[] animalPrefabs = new GameObject[10];
      [SerializeField]
      private float spawnRangeX = 20.0f;
      [SerializeField]
      private float spawnPozY = -20.0f;
      [SerializeField]
      private float startDelay = 5f;
      [SerializeField]
     private float spawnInterval = 2f;
      // Start is called before the first frame update
      void Start()
      {
          // invoke repeating, will make the method or void def function with name
  "nameInQuotes"
          // it will call that function after , delay, then chozen interval
          InvokeRepeating("SpawnRandomAnimal", startDelay, spawnInterval);
      }

      // Update is called once per frame
  ```
  ```

```
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.S))
        {
           SpawnRandomAnimal();
        }
    }
    void SpawnRandomAnimal()
    {
        // Select a random index within the array range
        int random_animal_Index = Random.Range(0, animalPrefabs.Length);
        Vector3 spawnPos = new Vector3(Random.Range(-spawnRangeX, spawnRangeX),
spawnPozY, -30);
        // Instantiate the selected animalPrefab
        Instantiate(animalPrefabs[random_animal_Index], spawnPos,
            animalPrefabs[random_animal_Index].transform.rotation);
    }
}
```

- movement

```
using UnityEngine;

public class Movement : MonoBehaviour
{
    [SerializeField]
    private float speed = 10.0f;
    [SerializeField]
    private float xRange = 10.0f;
    [SerializeField]
    private float horizontalInput;
    [SerializeField]
    public GameObject ProjectilePrefab;
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
```

```
    void Update()
    {


        horizontalInput = Input.GetAxis("Horizontal");
        // Calculate the desired position based on the input
        Vector3 desiredPosition = transform.position + Vector3.right * -
horizontalInput * Time.deltaTime * speed;
        // Clamp the desired position within the x range
        float clampedX = Mathf.Clamp(desiredPosition.x, -xRange, xRange);
        desiredPosition = new Vector3(clampedX, desiredPosition.y,
desiredPosition.z);
        // Move the object to the clamped position
        transform.position = desiredPosition;
        if (Input.GetKeyDown(KeyCode.Space))
        {
            // shoot pizza
            Instantiate(ProjectilePrefab, transform.position,
ProjectilePrefab.transform.rotation);
        }
    }
}
```

Animal Movement

```
using UnityEngine;

public class AnimalMovement : MonoBehaviour
{



    public float Animal_Movement_Speed = 50.0f;
    public float lowerBound = 50;
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
```

```
    {
        transform.Translate(Vector3.forward * Time.deltaTime *
Animal_Movement_Speed);
        if (transform.position.z > lowerBound)
        {
            Destroy(gameObject);
            Debug.Log("In the vernacular of your people... \n Game Over!");
        }
    }
}
```

-