Game Programming

Lecture III

Izzet Fatih Senturk

Enemy Setup

- Create a new game object: "Cube"
  - Set the position 0, 0, 0
  - Scale down to 0.75, 0.75, 0.75
  - Rename the cube to "Enemy"
- Make the cube prefab
  - Drag the "Enemy" into the "Prefabs" folder

Enemy Material

- Create a new material for the enemy
  - Create a new material named "Enemy_mat" under "Materials" folder
  - Change the color to red
- Apply the material to the "Enemy"
  - Either drag it on the object and use override
  - Or drag it on the prefab

Enemy Behavior

- Enemy moves down when it is instantiated
- Create a new C# script: “Enemy_sc”
- Apply the script to “Enemy” prefab

Define Speed

- The enemy moves 4 meters per second

```
public class Enemy_sc : MonoBehaviour
{
    [SerializeField]
    0 references
    private float _speed = 4.0f;
```

Move Down Enemy

- Enemy moves down
  - Move down the enemy according to speed defined earlier
- When the enemy is of the screen, respawn it a random x at the top of the screen
  - Check if y position is less than -5f
  - Select a random x value between -8 and 8
  - Relocate the enemy at the top (y = 7f) with the given x value

```
void Update()
{
    transform.Translate(Vector3.down * _speed * Time.deltaTime);

    if (transform.position.y < -5f)
    {
        float randomX = Random.Range(-8f, 8f);
        transform.position = new Vector3(randomX, 7, 0);
    }
}
```

Collisions

- Colliders detect collisions
- Currently “Player” and “Enemy” doesn’t collide
  - Use RigidBody to enable collisions
- Hard surface collision: when the ball hits the wall, it bounces back
- Trigger collision: collecting items (pass through the object)

- We don’t want hard surface collision for “Player”, “Enemy” and “Laser”
  - Set isTrigger true for all three game objects

RigidBody Component

- "Laser"-"Enemy" collision: Who gets the RigidBody component?
  - For performance reasons we will apply it on "Laser" only
  - Add RigidBody to "Laser" prefab
  - Uncheck "Apply Gravity"
- There are certain methods activated upon trigger
  - OnTriggerEnter
  - OnTriggerStay
  - OnTriggerExit
- These methods are called automatically by MonoBehavior just like
  - Start
  - Update
- Trigger is detected automatically
  - We can get information about the other object collided with us

RigidBody Component

- "Player"-"Enemy" collision: Who gets the RigidBody component?
  - For performance reasons we will apply it on "Enemy" only
  - Add RigidBody to "Enemy" prefab
  - Uncheck "Apply Gravity"

Detect Collision on Enemy

- Detect if “Enemy” collides with another object
  - Print the name of the collided object

```
private void OnTriggerEnter(Collider other)
{
    Debug.Log("Hit: " + other.transform.name);
}
```

Collision Behavior

- If the “Enemy” collides with “Player”
  - Damage “Player”
  - Destroy “Enemy”
- If the “Enemy” collides with “Laser”
  - Destroy “Laser”
  - Destroy “Enemy”
- How to differentiate “Enemy”, ”Laser”, and “Player” game objects?
  - Add two new tags: Enemy and Laser
  - Assign Enemy, Laser, and Player tags respectively

```
private void OnTriggerEnter(Collider other)
{
    if (other.tag == "Player")
    {
        //Damage the player
        Destroy(this.gameObject);
    }
    else if (other.tag == "Laser")
    {
        Destroy(other.gameObject);
        Destroy(this.gameObject);
    }
}
```

Player Lives and the Damage System

- Player lives is a variable
  - Define the lives variable in the "Player" script and set it to 3
  - Set it private so that other programs cannot change its value!
- Create a damage function to decrement the value of lives
  - If the lives is 0, destroy the "Player" game object

```
[SerializeField]
2 references
private int _lives = 3;
```

```
public void Damage()
{
    _lives--;

    if (_lives < 1)
    {
        Destroy(this.gameObject);
    }
}
```

Player Lives and the Damage System

- We should be able to call Damage function from the "Enemy" script upon collision
  - When "Enemy" collides with "Player" call Damage
- We will apply script communication for this purpose
  - We can use other.transform to access the transform component directly. But this is the only component we can access directly from "other" variable
  - We need to use GetComponent function

```
private void OnTriggerEnter(Collider other)
{
    if (other.tag == "Player")
    {
        //Damage the player
        Player_sc player = other.transform.GetComponent<Player_sc>();
        if (player != null)
        {
            player.Damage();
        }
        Destroy(this.gameObject);
    }
```