

LAB THỰC HÀNH – CHƯƠNG 3: UNITY SCRIPTING

Lab 1 – Component Lifecycle Debugger

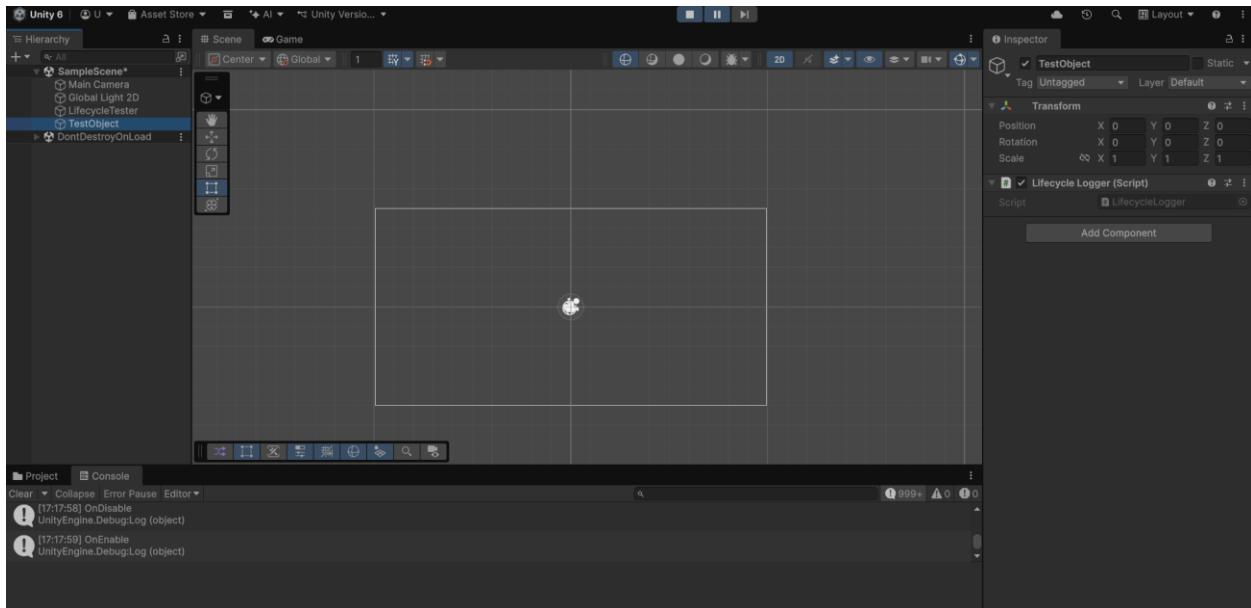
Tạo script

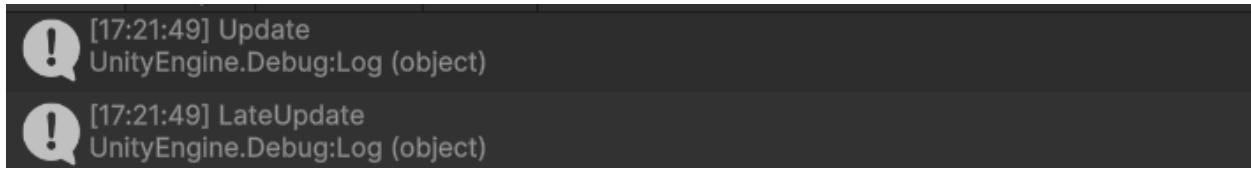
The screenshot shows the Visual Studio IDE interface. The top menu bar includes File, Edit, View, Git, Project, Debug, Test, Analyze, Tools, Extensions, Window, Help, and a Search bar. On the far right, there's a GitHub Copilot icon and a PREVIEW button. The left side features the code editor with LifecycleLogger.cs open, displaying C# code for a MonoBehaviour. The right side shows the Solution Explorer with a warning message about missing components for a full development experience, listing 'Assembly-CSharp' and 'Assembly-CSharp-Editor' as unloaded projects. The status bar at the bottom indicates the file is saved.

```
1 using UnityEngine;
2
3 public class LifecycleLogger : MonoBehaviour
4 {
5     void Awake()
6     {
7         Debug.Log("Awake");
8     }
9
10    void OnEnable()
11    {
12        Debug.Log("OnEnable");
13    }
14
15    void Start()
16    {
17        Debug.Log("Start");
18    }
19
20    void Update()
21    {
22        Debug.Log("Update");
23    }
24
25    void FixedUpdate()
26    {
27        Debug.Log("FixedUpdate");
28    }
29
30    void LateUpdate()
31    {
```

Toggle Active:

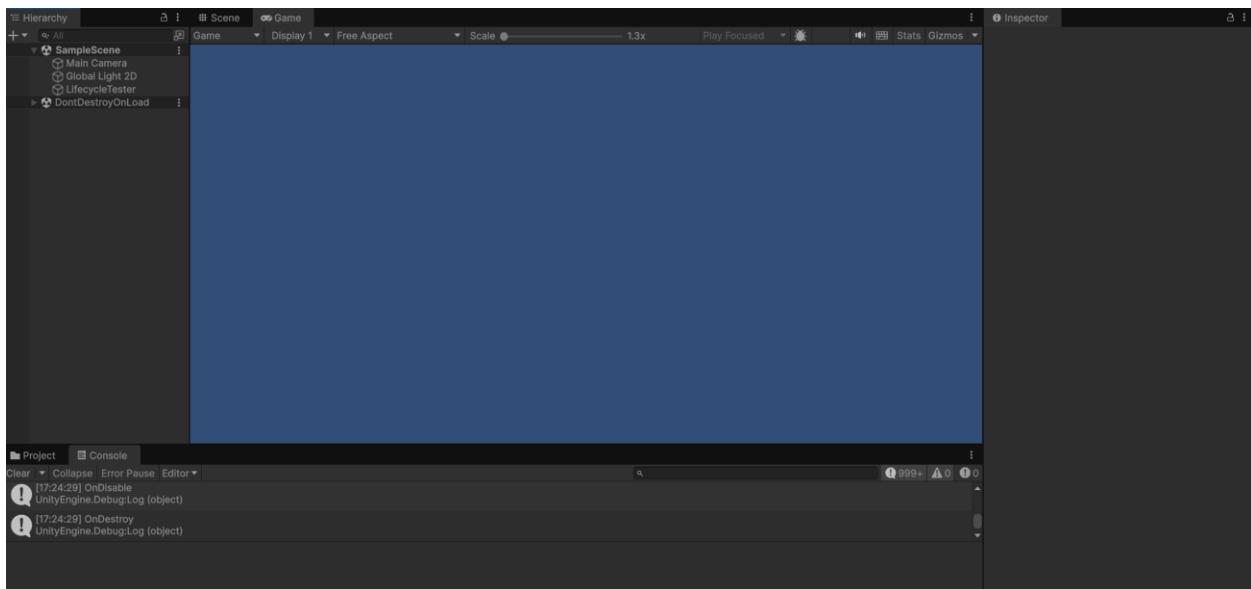
Click TestObject trong Hierarchy





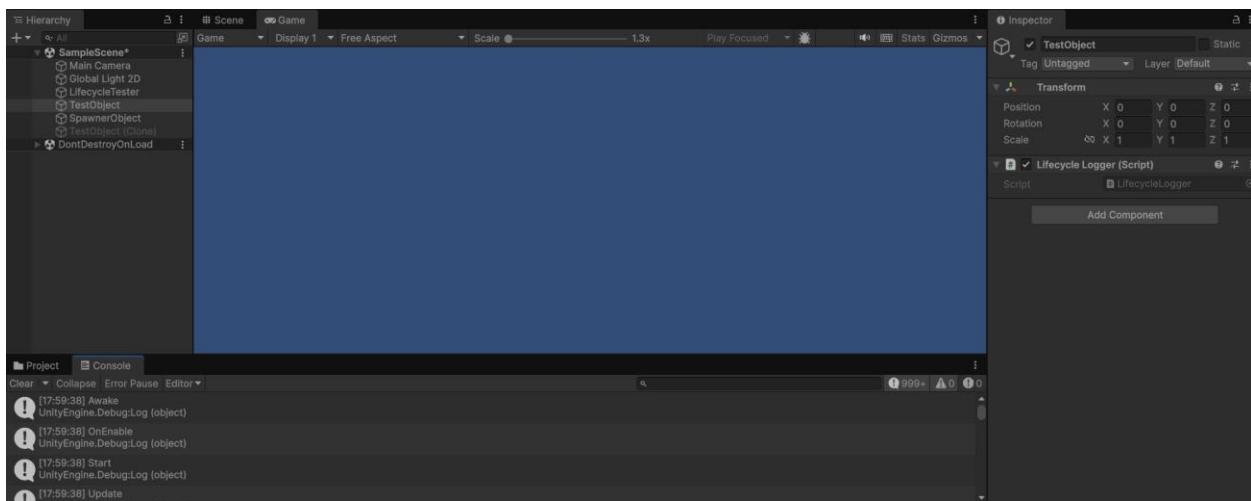
DESTROY OBJECT:

- Chọn **TestObject** trong Hierarchy chọn **Delete**



INSTANTIATE OBJECT:

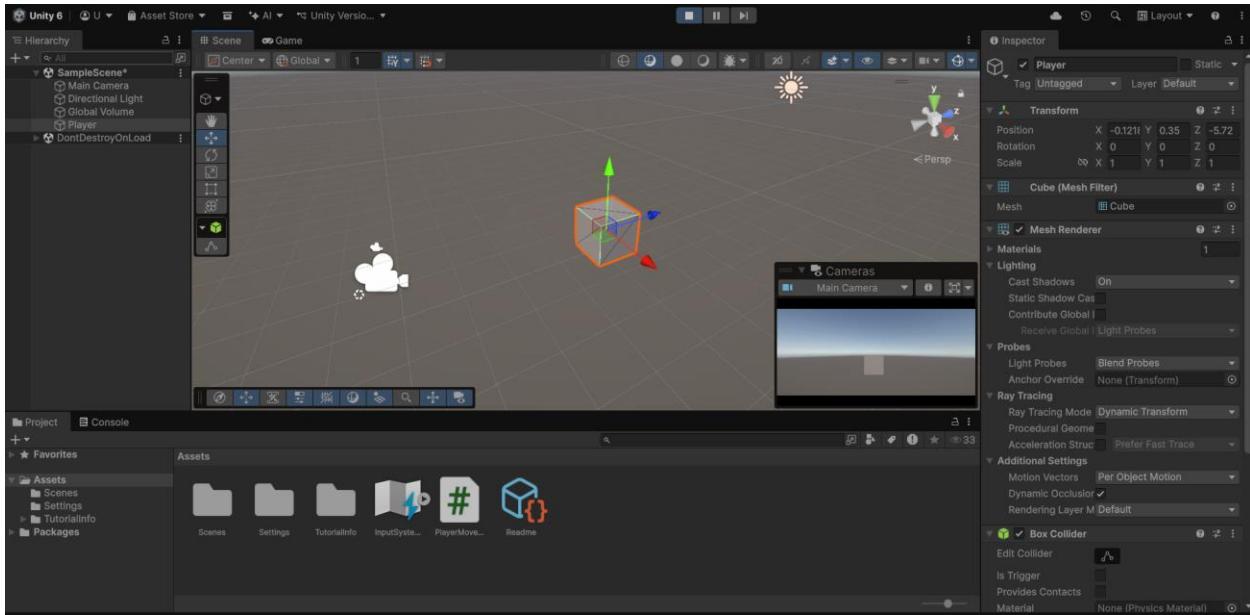
Tạo SpawnerObject và kéo **TestObject (Prefab)** vào ô Prefab



Lab 2 – Vector Movement & Gizmos

Mục tiêu :Điều khiển nhân vật bằng WASD, chuẩn hóa vector để tránh chạy chéo nhanh:

Tạo Player



Tạo PlayerMovement

```
PlayerMovement.cs* □ ×
Miscellaneous Files □ PlayerMovement □ Update()
1  using UnityEngine;
2
3  public class PlayerMovement : MonoBehaviour
4  {
5      public float speed = 5f;           // Tốc độ di chuyển
6      private Vector3 moveDirection;    // Lưu hướng di chuyển
7
8      void Update()
9      {
10         // Nhận input WASD
11         float h = Input.GetAxisRaw("Horizontal"); // A, D
12         float v = Input.GetAxisRaw("Vertical");   // W, S
13
14         // Tạo vector di chuyển
15         Vector3 move = new Vector3(h, 0, v);
16
17         // Chuẩn hóa vector để tránh chạy chéo nhanh
18         if (move.magnitude > 1)
19         {
20             move = move.normalized;
21         }
22
23         // Lưu hướng để vẽ Gizmos
24         moveDirection = move;
25
26         // Di chuyển nhân vật
27         transform.position += move * speed * Time.deltaTime;
28     }
29
30     // Vẽ Gizmos hiển thị hướng di chuyển

```

Đã điều khiển nhân vật bằng WASD , chuẩn hóa vector để tránh chạy chéo nhanh và Gizmos hiển thị hướng di chuyển.

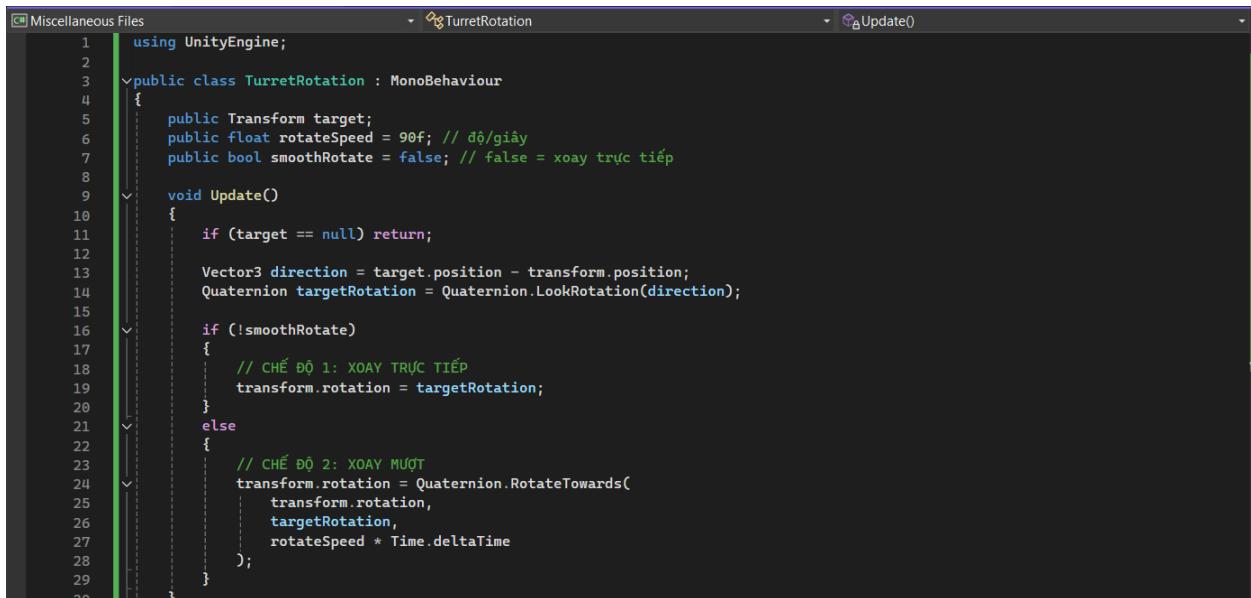
Normalize: là đưa vector về độ dài bằng **1** nhưng **giữ nguyên hướng**, giúp nhân vật di chuyển **cùng tốc độ** dù đi thẳng hay đi chéo, tránh trường hợp nhân vật nhiều phím làm chạy nhanh hơn.

Lab 3 – Quaternion Rotation

Tạo **Cube** → rename: Turret

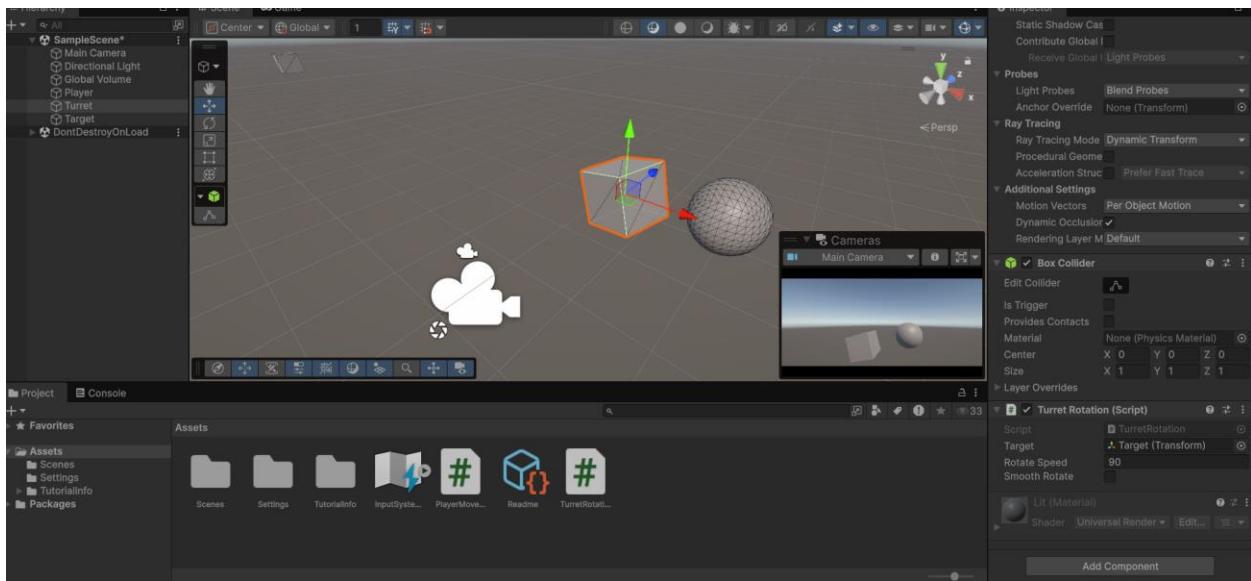
Tạo **Sphere** → rename: Target

Tạo script: **TurretRotation.cs**



```
1  using UnityEngine;
2
3  public class TurretRotation : MonoBehaviour
4  {
5      public Transform target;
6      public float rotateSpeed = 90f; // độ/giây
7      public bool smoothRotate = false; // false = xoay trực tiếp
8
9      void Update()
10     {
11         if (target == null) return;
12
13         Vector3 direction = target.position - transform.position;
14         Quaternion targetRotation = Quaternion.LookRotation(direction);
15
16         if (!smoothRotate)
17         {
18             // CHẾ ĐỘ 1: XOAY TRỰC TIẾP
19             transform.rotation = targetRotation;
20         }
21         else
22         {
23             // CHẾ ĐỘ 2: XOAY MƯỢT
24             transform.rotation = Quaternion.RotateTowards(
25                 transform.rotation,
26                 targetRotation,
27                 rotateSpeed * Time.deltaTime
28             );
29     }
30 }
```

Turret xoay nhìn target bằng LookAt và RotateTowards/Slerp.



20260120-1532-16.12
39605.mp4

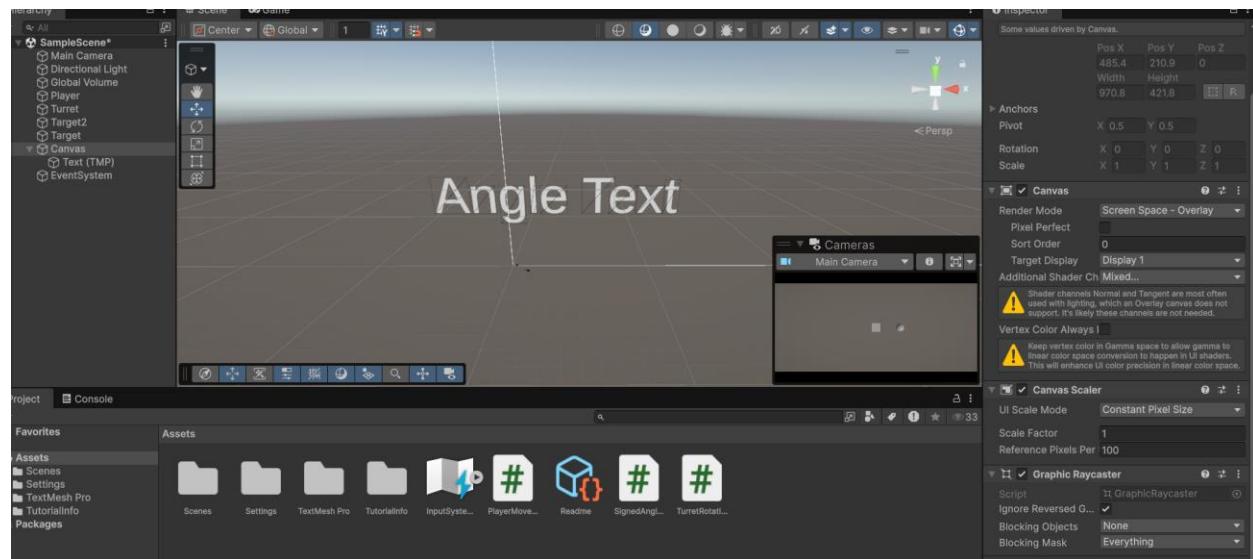
Lab 4 – Signed Angle (2D/Topdown)

Tạo “Target”

Tạo SignedAngleRotate.cs:

```
1  using UnityEngine;
2
3  public class SignedAngleRotate : MonoBehaviour
4  {
5      public Transform target;
6      public float currentAngle;
7
8      void Update()
9      {
10         if (target == null) return;
11
12         // Vector hướng hiện tại của Player (trục Z)
13         Vector3 forward = transform.forward;
14
15         // Vector từ Player tới Target
16         Vector3 toTarget = target.position - transform.position;
17
18         // Chỉ xét mặt phẳng XZ (topdown)
19         forward.y = 0;
20         toTarget.y = 0;
21
22         // Tính góc SignedAngle
23         currentAngle = Vector3.SignedAngle(forward, toTarget, Vector3.up);
24
25         // Xoay Player theo góc đó
26         transform.Rotate(0, currentAngle * Time.deltaTime, 0);
27     }
28 }
```

Chọn Text (TMP) → Inspector: Angle: 0



Di chuyển Target

Angle: -23.4

Angle: 45.1

LAB 5 – OBSERVER PATTERN (C# EVENT)

TAO PLAYER HEALTH (SUBJECT)

Tạo Script PlayerHealth

The screenshot shows the Unity Editor interface. At the top, the code editor displays the `PlayerHealth` script:

```
1  using UnityEngine;
2  using System;
3
4  public class PlayerHealth : MonoBehaviour
5  {
6      public int maxHealth = 100;
7      public int currentHealth;
8
9      // EVENT: phát khi máu thay đổi
10     public event Action<int> OnHealthChanged;
11
12    void Start()
13    {
14        currentHealth = maxHealth;
15        OnHealthChanged?.Invoke(currentHealth);
16    }
17
18    void Update()
19    {
20        // Nhấn H để trừ máu
21        if (Input.GetKeyDown(KeyCode.H))
22        {
23            TakeDamage(10);
24        }
25    }
26
27    void TakeDamage(int damage)
28    {
29        currentHealth -= damage;
30        currentHealth = Mathf.Clamp(currentHealth, 0, maxHealth);
31    }
}
```

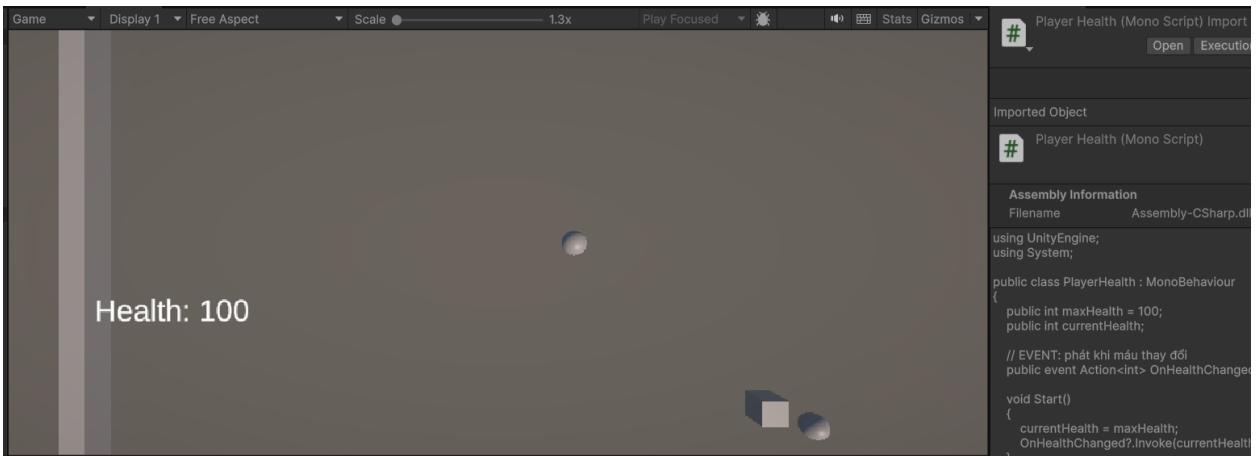
In the middle, the Unity Hierarchy window shows the `SampleScene*` containing objects like Main Camera, Directional Light, Global Volume, Player, Turret, Target2, Target, and Canvas. The Player object has a Box Collider component attached.

On the right, the Inspector window shows the `PlayerHealth` component settings:

- Box Collider: Center (X: 0, Y: 0, Z: 0), Size (X: 1, Y: 1, Z: 1)
- Player Movement (Script): Speed: 5
- Player Health (Script): Max Health: 100, Current Health: 70

The Project window at the bottom lists Assets such as Scenes, Settings, TextMesh Pro, TutorialInfo, InputSystem, PlayerHealth, PlayerMove..., Readme, SignedAngle, and TurretRotate... .

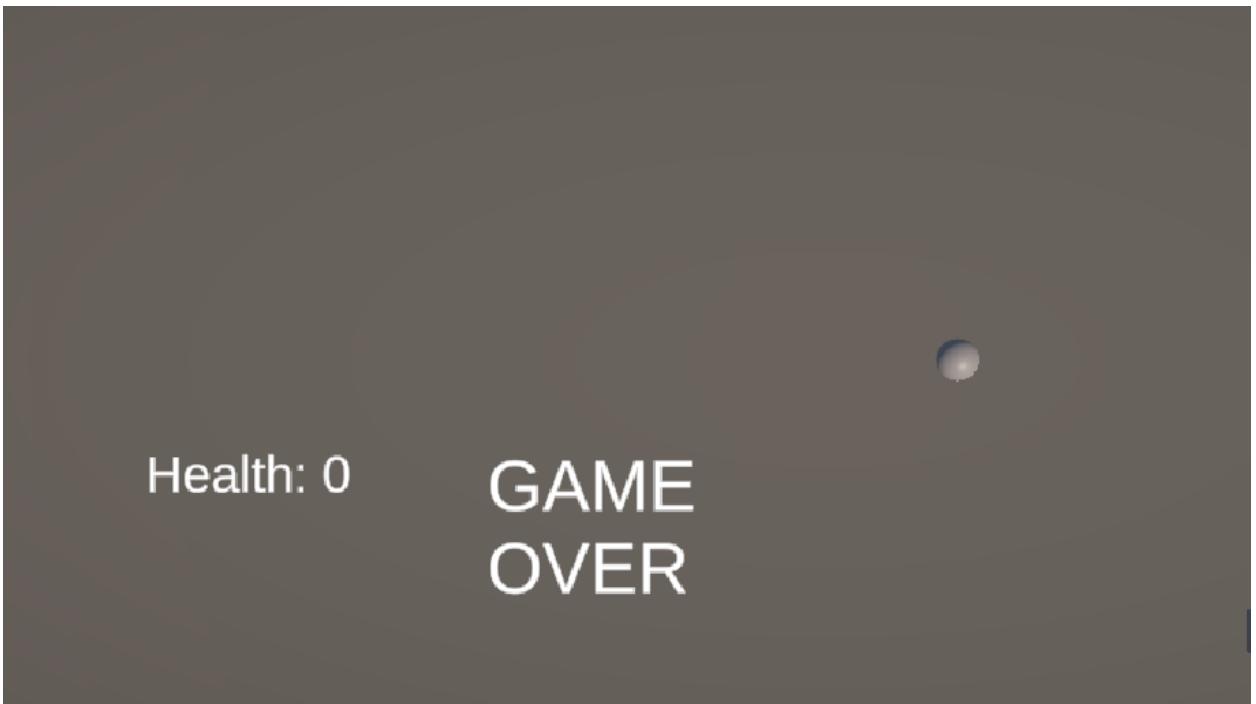
TEXT HEALTH



Trong Inspector:

- Text: Health
- Font Size: 28
- Auto Size: **OFF**

Tạo GameOver Text và Khi HP = 0:



LAB 6 – OBSERVER PATTERN (UNITYEVENT)

làm giống Lab 5, nhưng KHÔNG dùng event C#, chỉ dùng UnityEvent.

```
using UnityEngine;
using UnityEngine.Events;

public class PlayerHealthUE : MonoBehaviour
{
    public int health = 100;

    public UnityEvent<int> OnHealthChanged;

    void Update()
    {
        if (Input.GetKeyDown(KeyCode.H))
        {
            health -= 10;
            health = Mathf.Clamp(health, 0, 100);

            OnHealthChanged.Invoke(health);
        }
    }
}
```