그래스핑 심화, 총잡기 및 총알 발사!

1. 이준

전투 시스템 만들기!

- Player 도 HP를 설정 HP = 100.0f;
- BulletSpawner 의 Bullet 데미지 50.0f; (2번 맞으면 죽도록 있도록 하자!)

PlayerController.cs* | Bullet.cs | CustomController.cs | SimpleShoot.cs | BulletSpawn

Assembly-CSharp

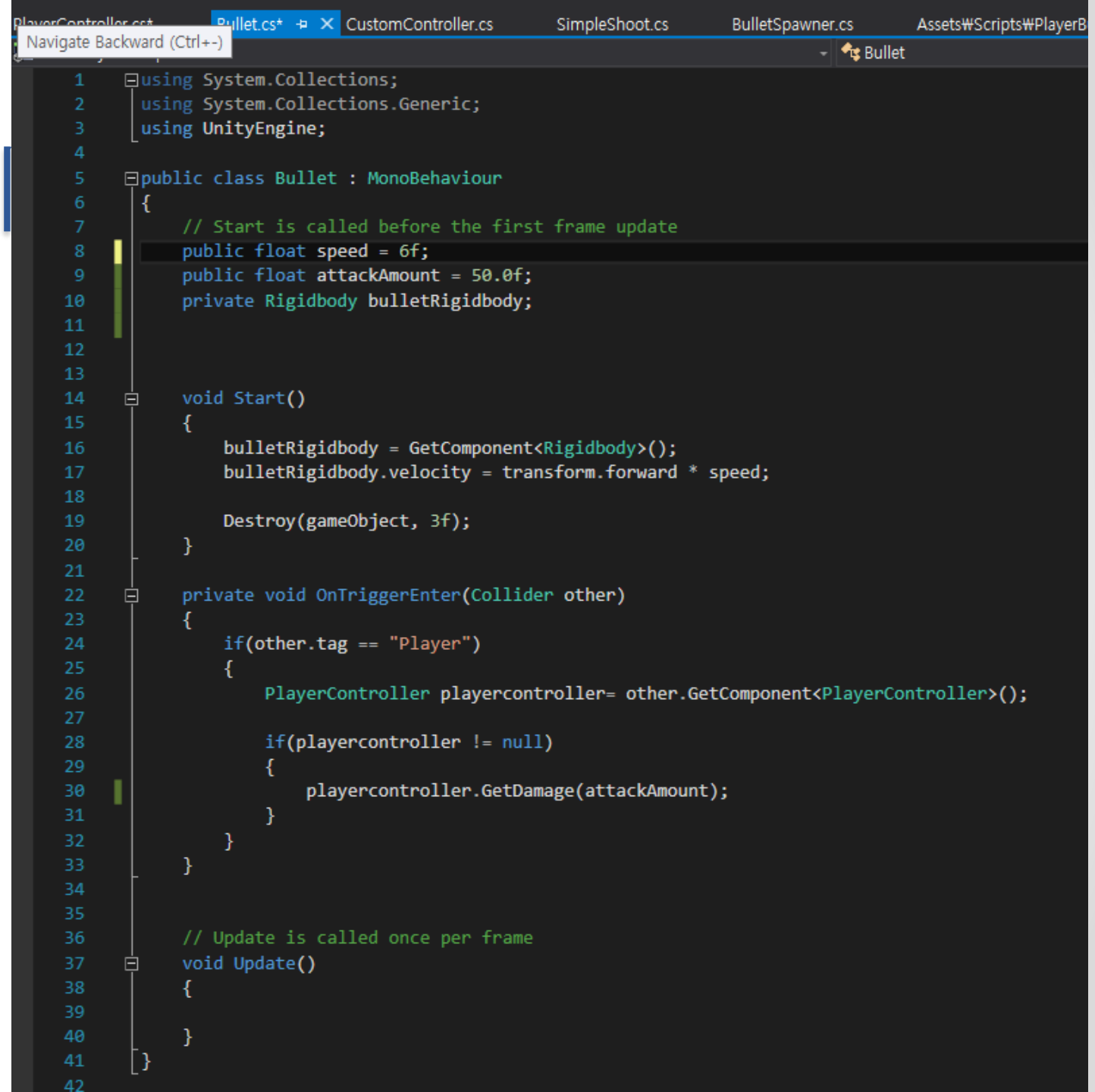
```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UIElements;
5
6  public class PlayerController : MonoBehaviour
7  {
8      private Rigidbody playerRigidbody;
9      public float speed = 8f;
10
11     public float hp = 100.0f;
12
13
14     // Start is called before the first frame update
15     void Start()
16     {
17         playerRigidbody = GetComponent<Rigidbody>();
18     }
19
20     // Update is called once per frame
21     void Update()
22     {
23
24
25
26
27
28
29
30
31
32
33
34
35
36     }
```

이부분 코드 삭제 해야 합니다~

```
38     public void GetDamage(float amount)
39     {
40         hp -= amount;
41
42         if (hp < 0)
43         {
44             //Bullet Spawner가 죽음 SetActive를 false로 설정
45             Die();
46         }
47     }
48
49
50     public void Die()
51     {
52         gameObject.SetActive(false);
53     }
54 }
```

전투 시

- Bullet.cs 도 수정
- 총알 속도고 6f 로 줄이자!

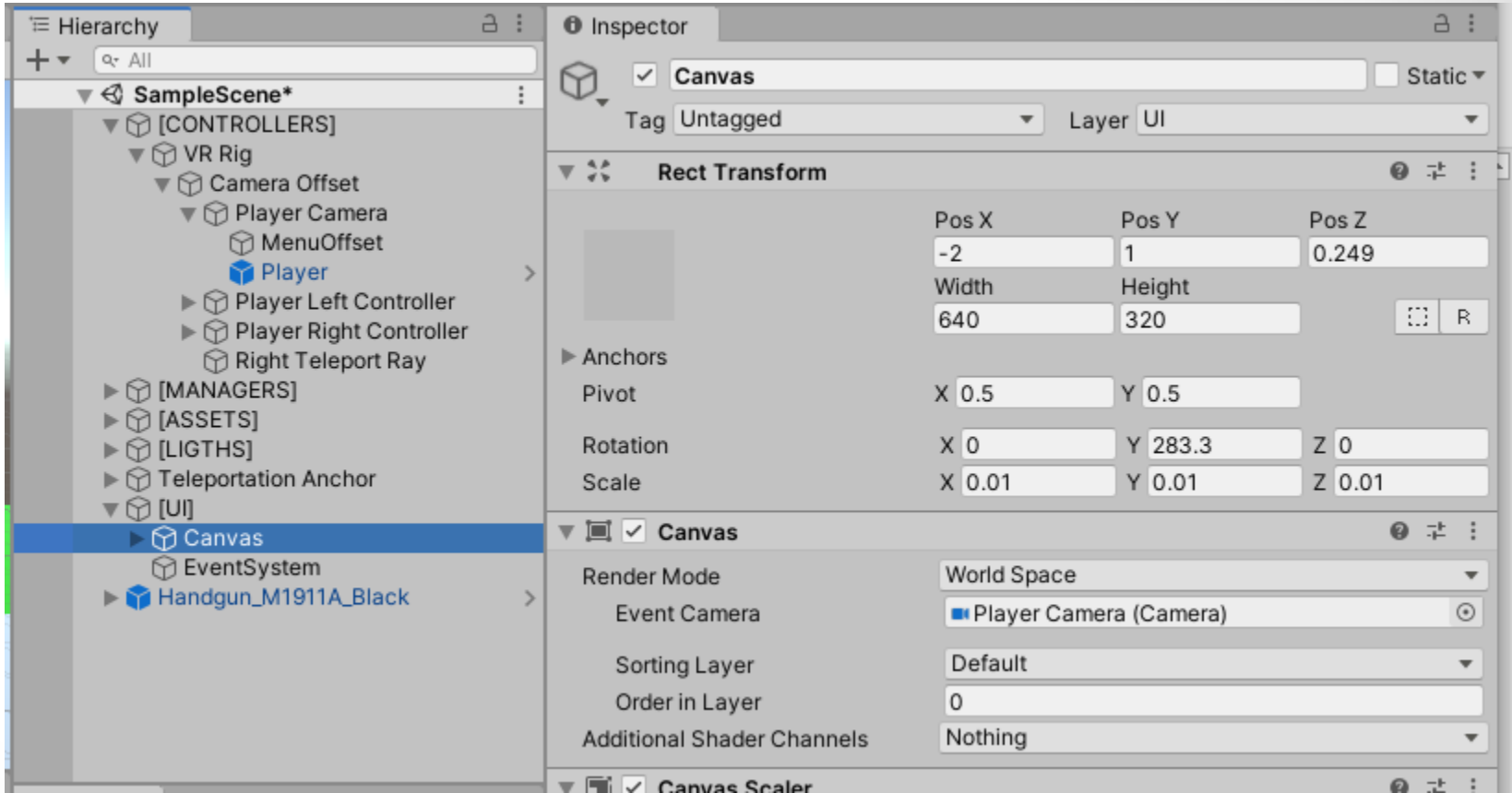


전투 시

- 이제 게임 정보를 알수 있는UI 및 재시작을 담당해주는 GameManager 가 필요함!

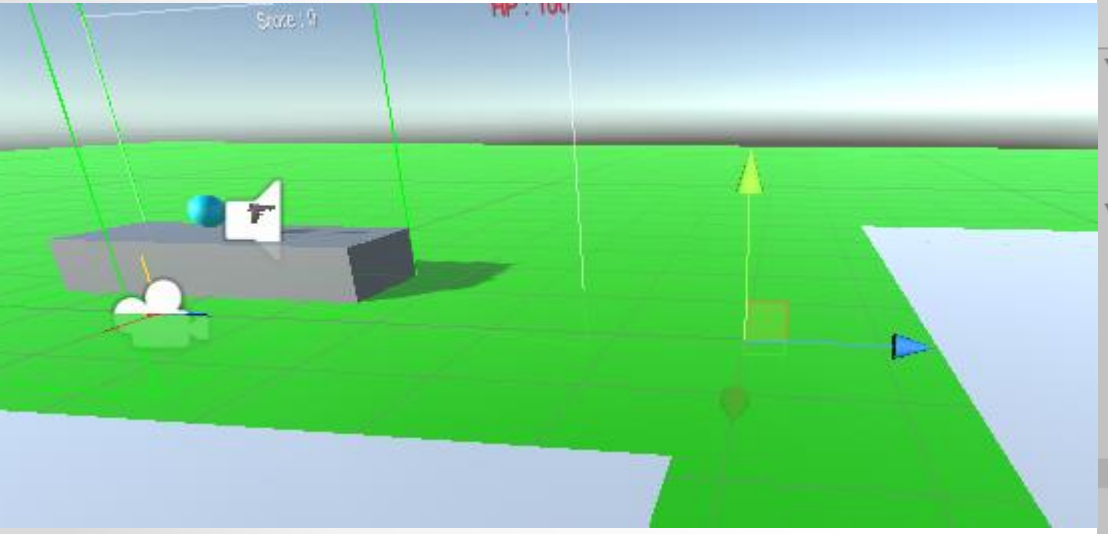
전투 시스템 만들기!

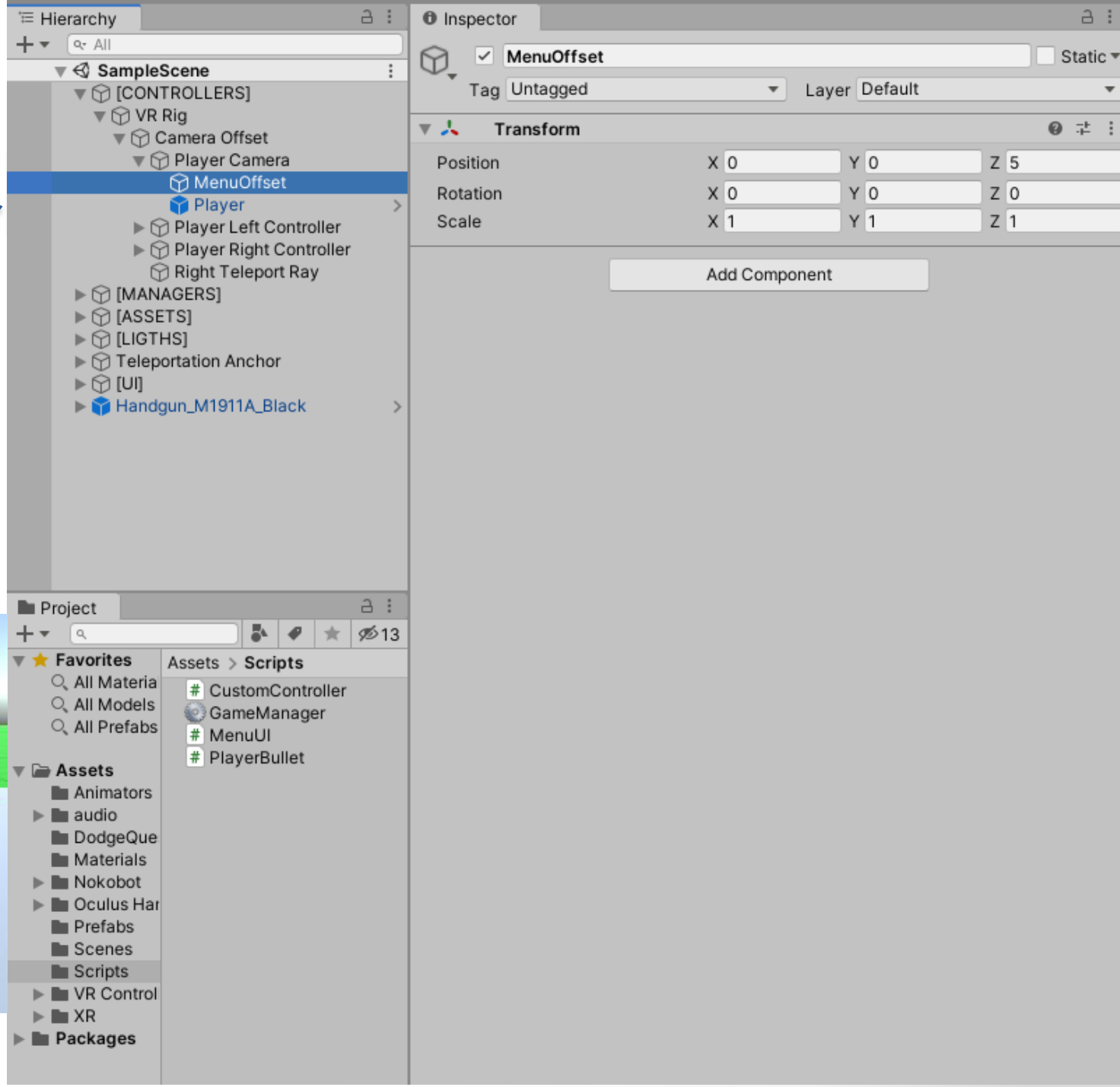
- 캔버스 설정
- 다음의 위치 정보로 변경



전투 시

- 메뉴가 플레이어에게 항상 보이게 위해서 MenuOffset을 설정
- PlayerCamera 의 자식노드
- Player 와 비슷한 방법을 적용





전투 시스템 만들기!

- MenuUI.cs 파일 생성후 다음과 같이 코딩
- 실시간으로 Canvas의 위치를 가상의 MenuOffset으로 설정해줌!

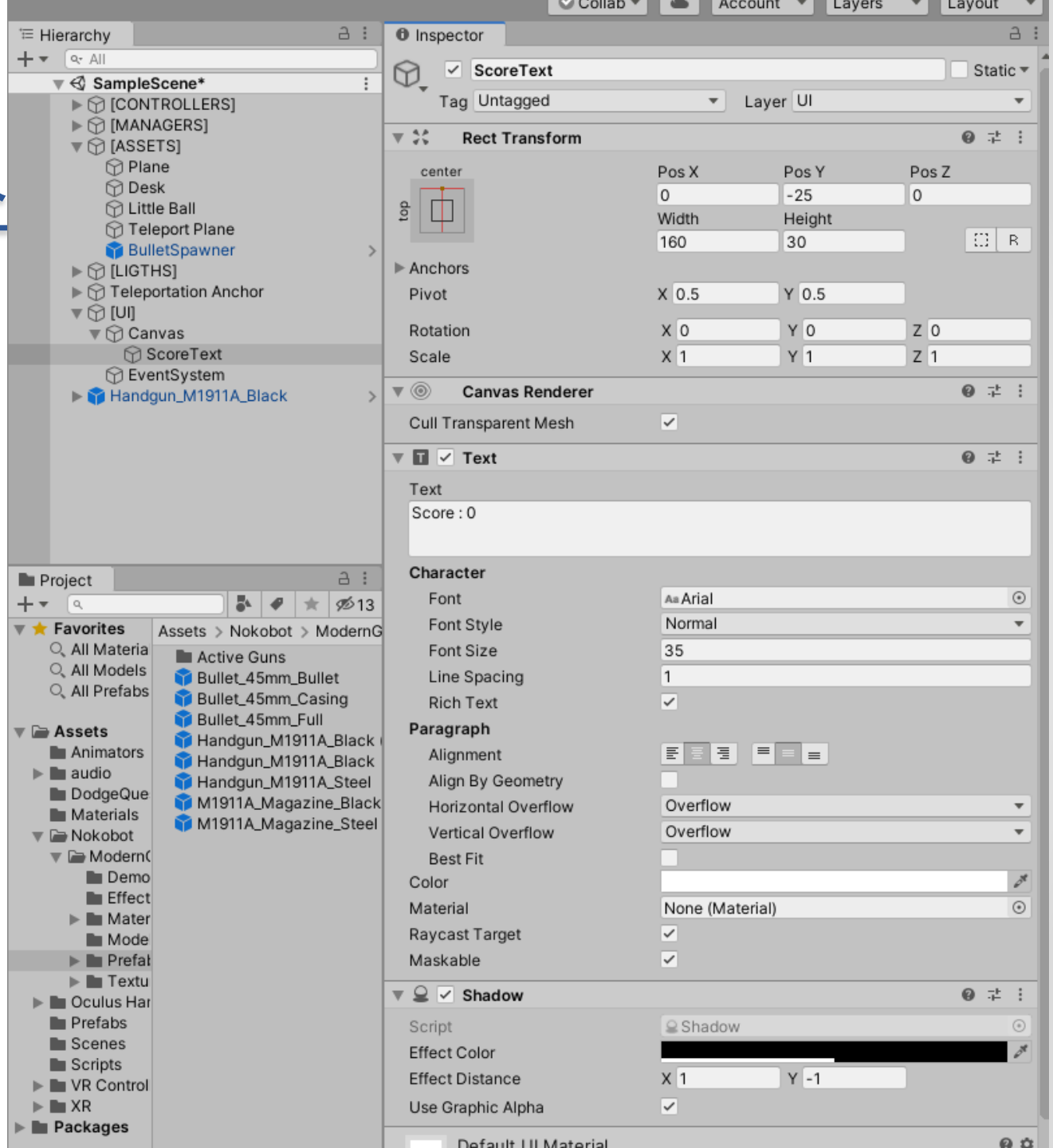
Assembly-CSharp — MenuUI

C:¥Users¥User¥Documents¥Quest2_UI¥Assets¥Scripts¥MenuUI.cs

```
2   using System.Collections.Generic;
3   using UnityEngine;
4   using UnityEngine.UI;
5
6   public class MenuUI : MonoBehaviour
7   {
8
9
10      public GameObject menuOffset;
11
12      // Start is called before the first frame update
13      void Start()
14      {
15
16      }
17
18      void Update()
19      {
20          if (menuOffset != null)
21          {
22              this.transform.position = menuOffset.transform.position;
23              this.transform.rotation = menuOffset.transform.rotation;
24
25          }
26      }
27  }
```

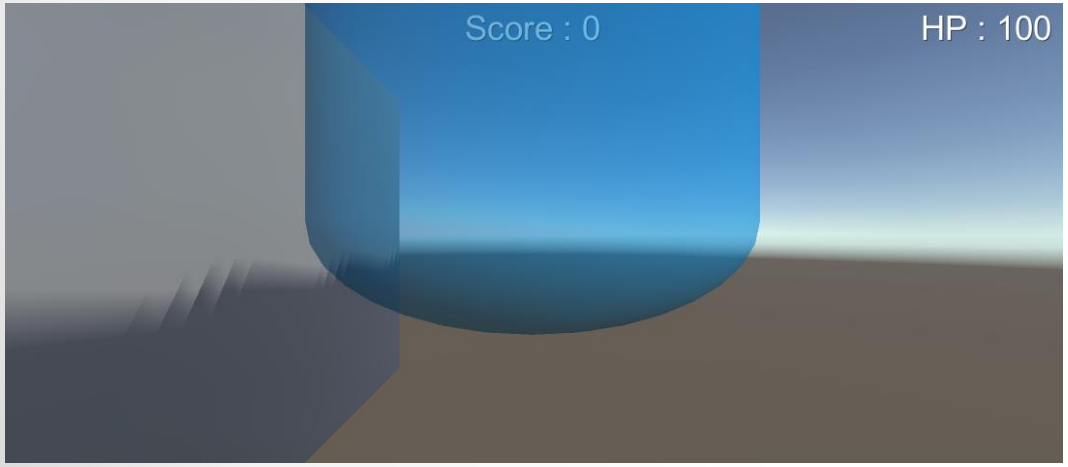
전투 시스

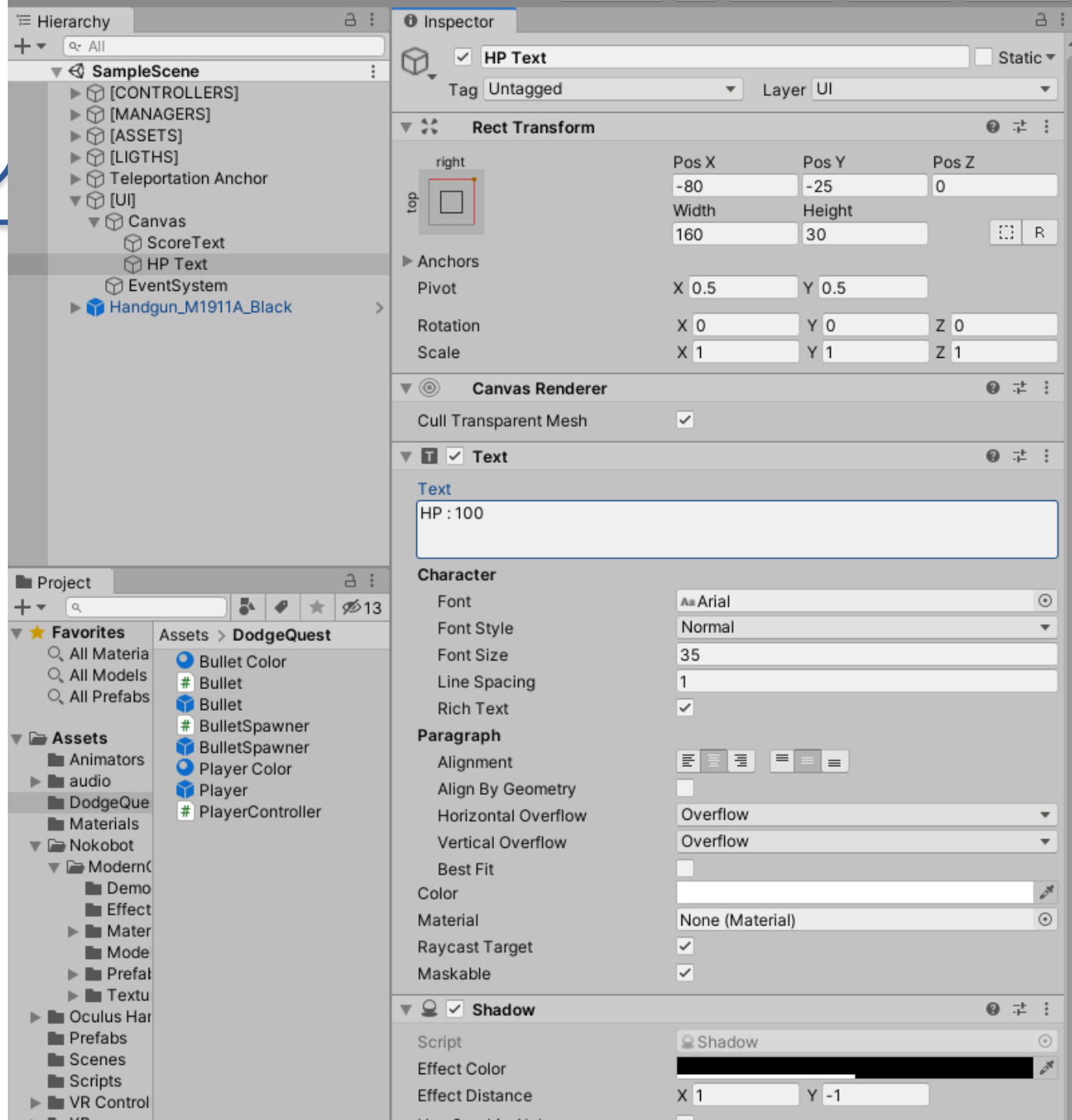
- **Score Text!**
  - 앵커 프리셋 Top-Center
  - PosY = -25
  - Text 는 Score : 0
  - 가운데 정렬
  - Overflow 설정
  - Shadow 넣기



전투 시스

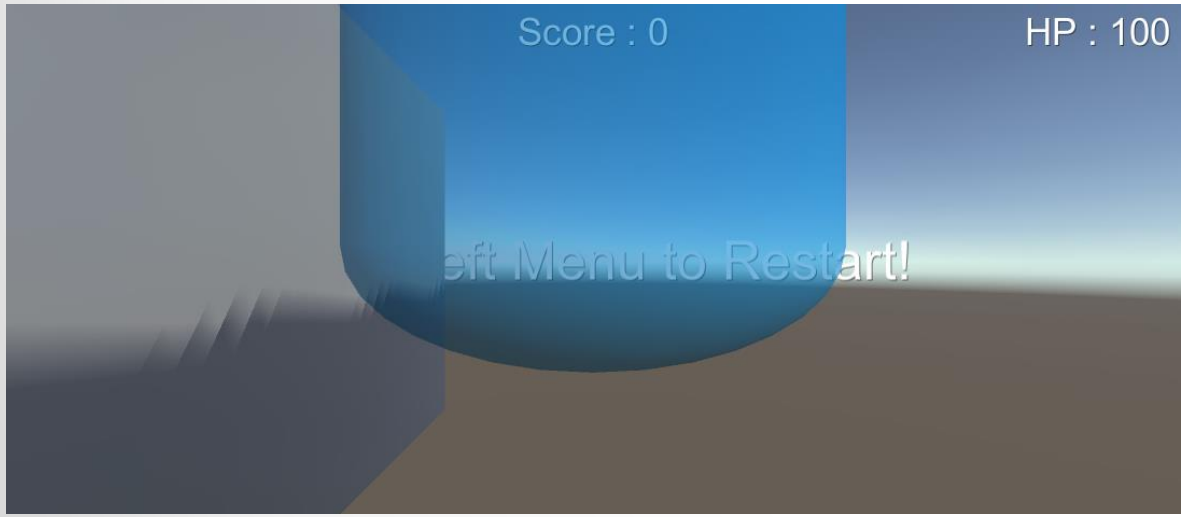
- **HP Text!**
  - 앵커 프리셋 Top-Center
  - PosX = -80 PosY = -25
  - Text 는 HP : 100
  - 가운데 정렬
  - Overflow 설정
  - Shadow 넣기

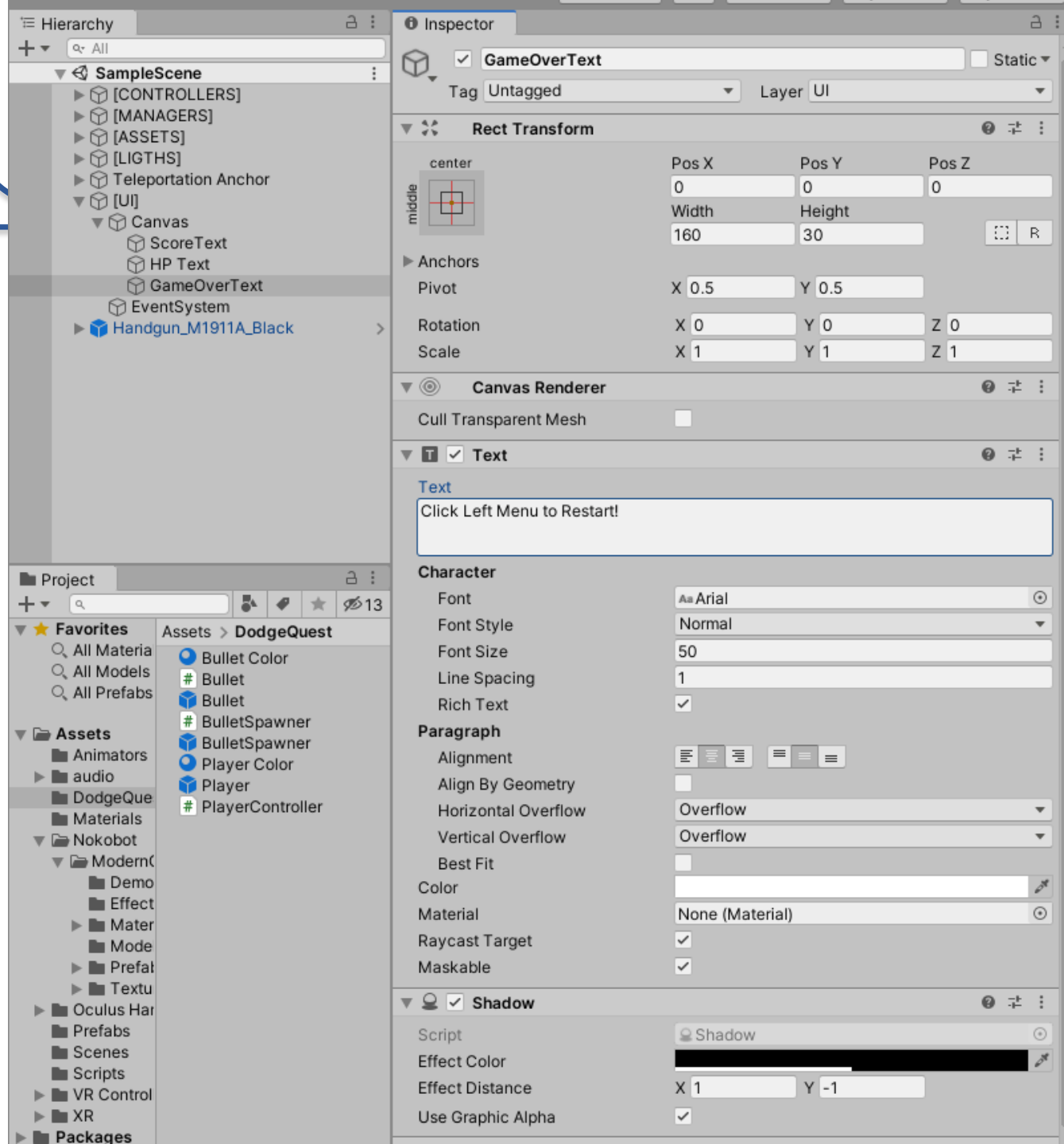




전투 시스

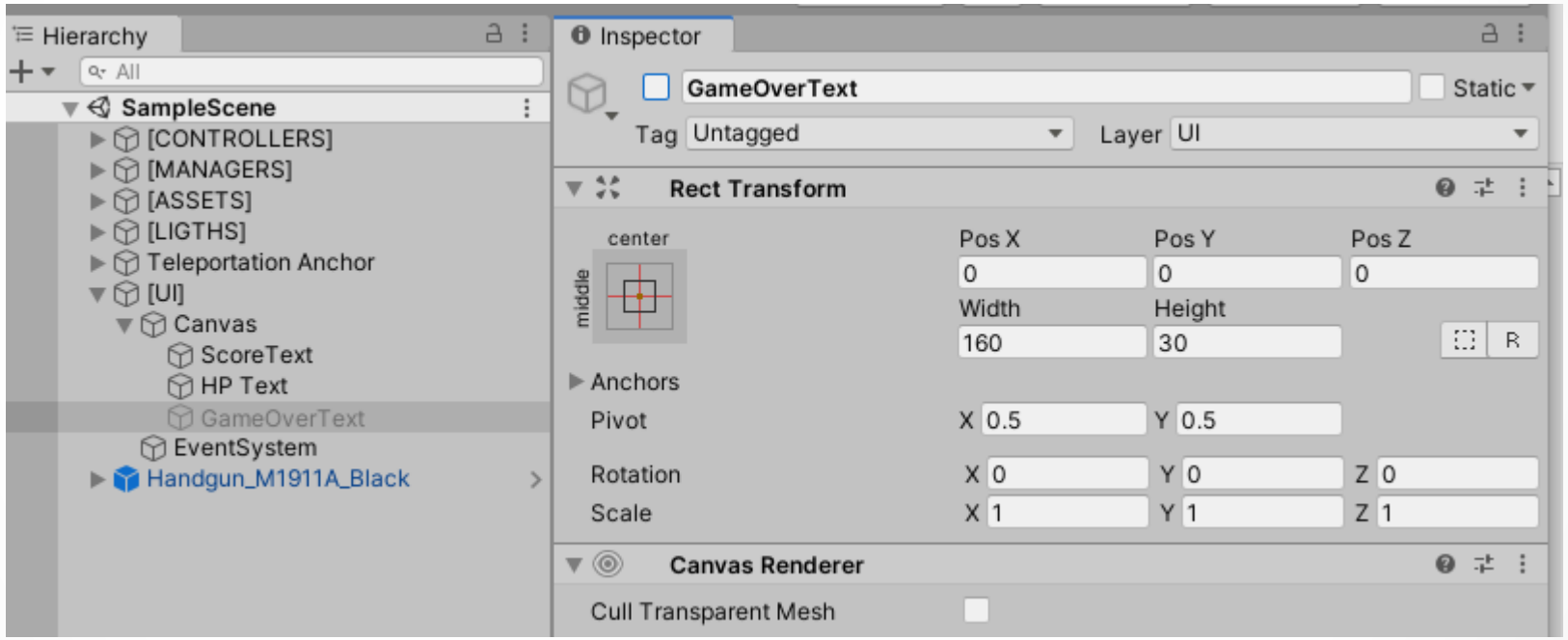
- **GameOverText!**
  - 앵커 프리셋 Center
  - Text Click Left Menu to Restart!
  - 가운데 정렬
  - Overflow 설정
  - Shadow 넣기





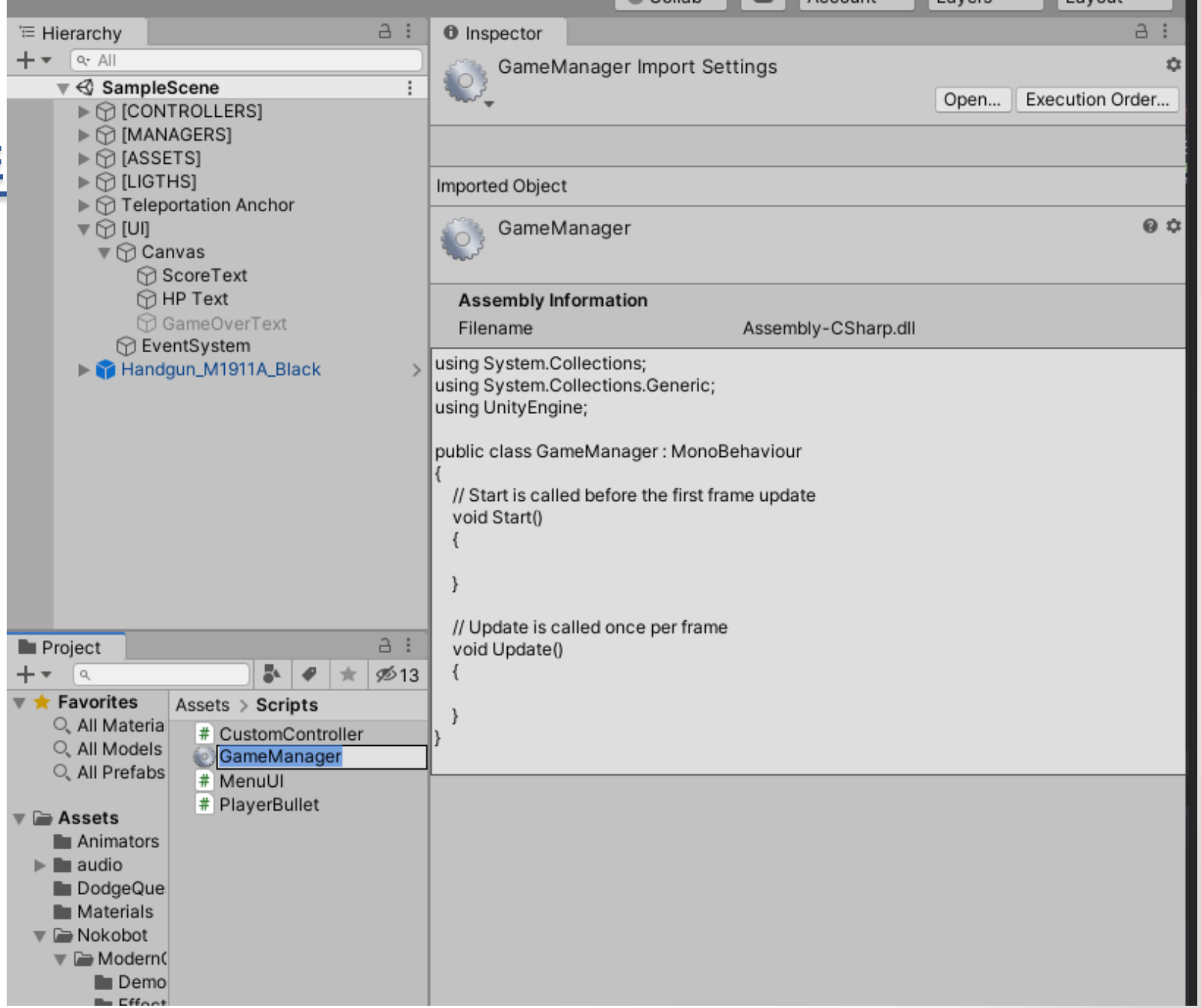
전투 시스템 만들기!

- GameOverText 는 일단 비활성화하기!



전투

- 게임 플레이를 담당하는
- GameManger.cs 파일을 생성!



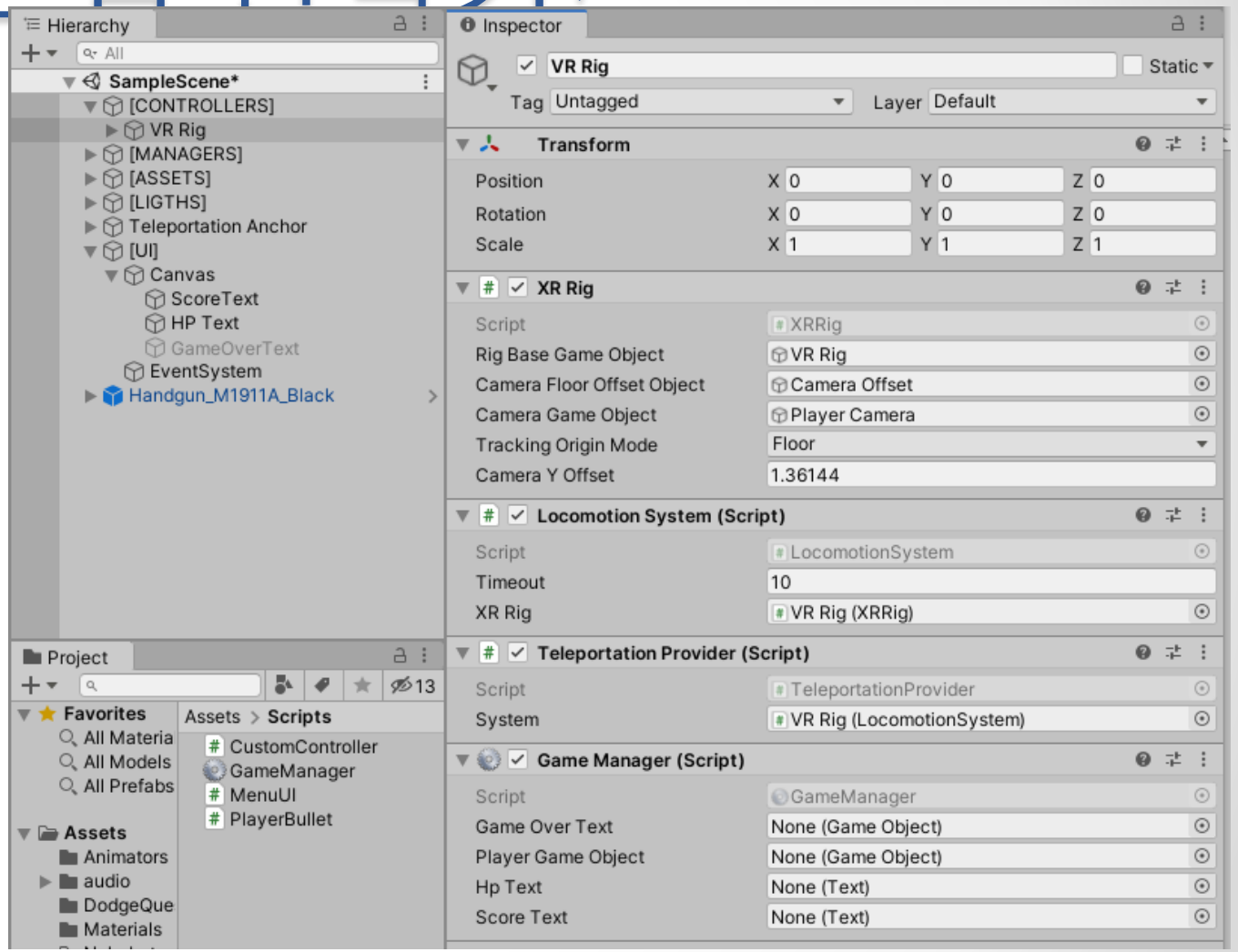
전투 시

- GameManger.cs 파일
- 닷지 플레이어와 유사함
- UI 텍스트 변수들을 가지고,
- Score 와 현재 플레이어의 HP 표현!

```
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class GameManager : MonoBehaviour
{
    public GameObject gameOverText;
    public GameObject playerGameObject;
    public Text hpText;
    public Text scoreText;
    int score;
    public bool isGameOver;

    void Start()
    {
        score = 0;
        isGameOver = false;
    }
    void Update()
    {
        if(!isGameOver)
        {
            hpText.text = "HP :" + (int)playerGameObject.GetComponent<PlayerController>().hp;
            scoreText.text = "Score :" + (int)score;
        }
    }
    public void GetScored(int value)
    {
        score += value;
    }
    public void EndGame()
    {
        isGameOver = true;
        gameOverText.SetActive(true);
    }
    public void RestartGame()
    {
        SceneManager.LoadScene("SampleScene");
    }

}
```

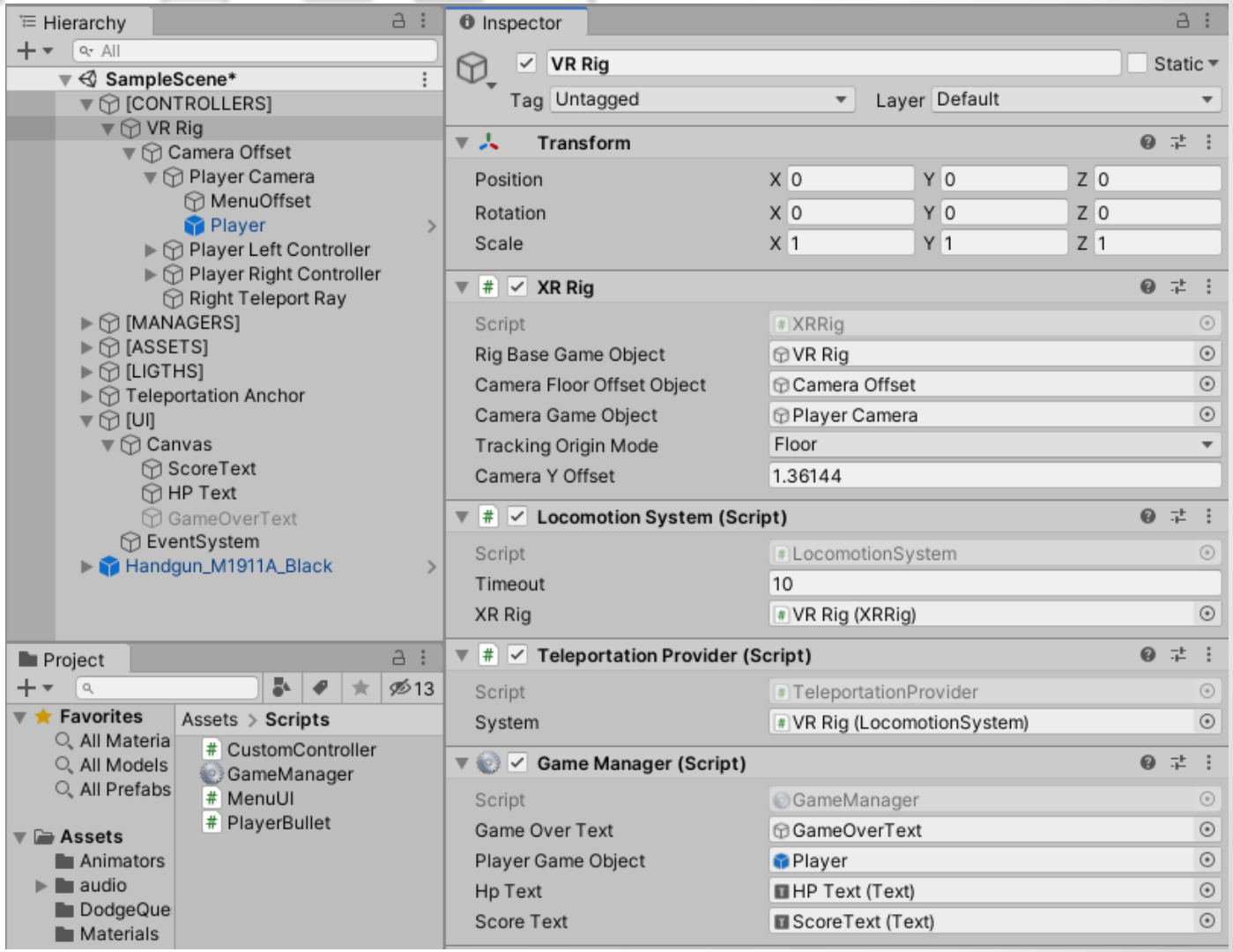
전투 시스템 만들기!

- VRRig에 GameManager 추가!



전투 시스템 만들기!

- Game Manager에 다음의
- 내용들을 드래그 & 드롭



전투 시스템 만들기!

- BulletSpawner.cs 위 GetDamager 함수 수정!
- GameManager를 가져와서 점수를 1점 더한다

```
50
51      public void GetDamage(float amount)
52      {
53          hp -= amount;
54
55          if(hp < 0)
56          {
57              //Bullet Spawner가 죽음 SetActive를 false로 설정
58              gameObject.SetActive(false);
59
60              FindObjectOfType<GameManager>().GetScored(1);
61          }
62      }
```

전투 시스템 만들기!

- PlayerController.cs의 Die 함수 수정!
- 게임 매니저를 찾아서 게임을 종료한다!

```
49
50      public void Die()
51      {
52          gameObject.SetActive(false);
53
54          FindObjectOfType<GameManager>().EndGame();
55      }
```

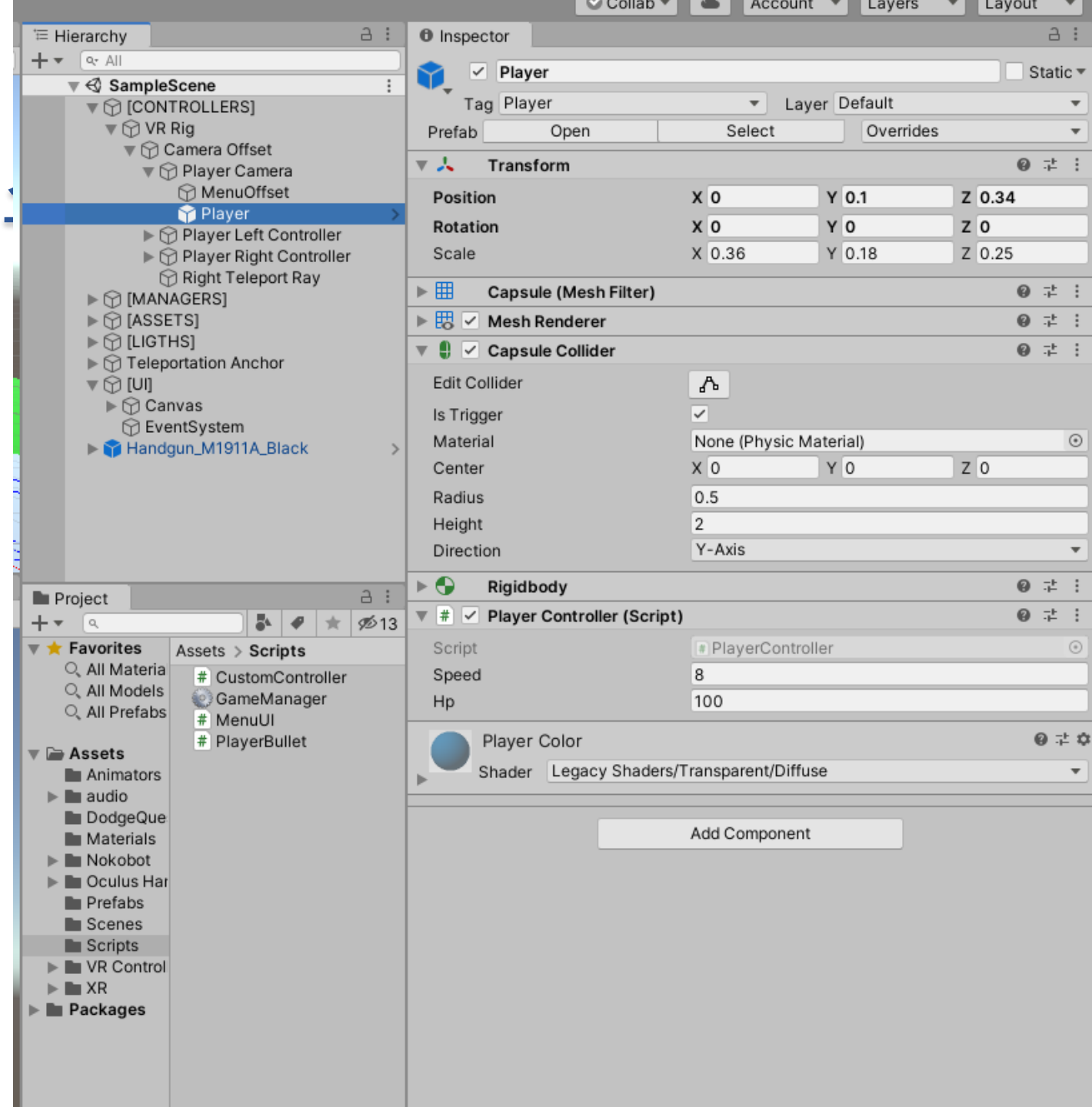
전투 시스

- 왼쪽 메뉴 버튼 클릭 인식!
- CustomController.cs 수정!
- menuButton 부분만 코드를 추가하면 됨!

```
71      void Update()
72      {
73          if (!availableDevice.isValid)
74          {
75              TryInitialize();
76          }
77          if (renderController)
78          {
79              handInstance.SetActive(false);
80              controllerInstance.SetActive(true);
81          }
82          else
83          {
84              handInstance.SetActive(true);
85              controllerInstance.SetActive(false);
86              UpdateHandAnimation(); // 핸드 애니메이션은 여기서만 수행
87          }
88          if (HandGun != null)
89          {
90              bool menuButtonValue;
91              if (availableDevice.TryGetFeatureValue(CommonUsages.triggerButton, out menuButtonValue) && menuButtonValue)
92              {
93                  if(triggerButton == false)
94                  {
95                      HandGun.GetComponent<SimpleShoot>().Shoot();
96                      triggerButton = true;
97                  }
98              }
99              else
00              {
01                  triggerButton = false;
02              }
03          }
04          if (FindObjectOfType<GameManager>().isGameOver)
05          {
06              bool menuButtonValue;
07              if (availableDevice.TryGetFeatureValue(CommonUsages.menuButton, out menuButtonValue) && menuButtonValue)
08              {
09                  FindObjectOfType<GameManager>().RestartGame();
10              }
11          }
12      }
```

전투 시

- 플레이어 최종 정리
- Capsule Collider 의 Is Trigger 활성화!
- Mesh Render 비활성화 (반투명한 파란색이 더 이상 안나오도록!)



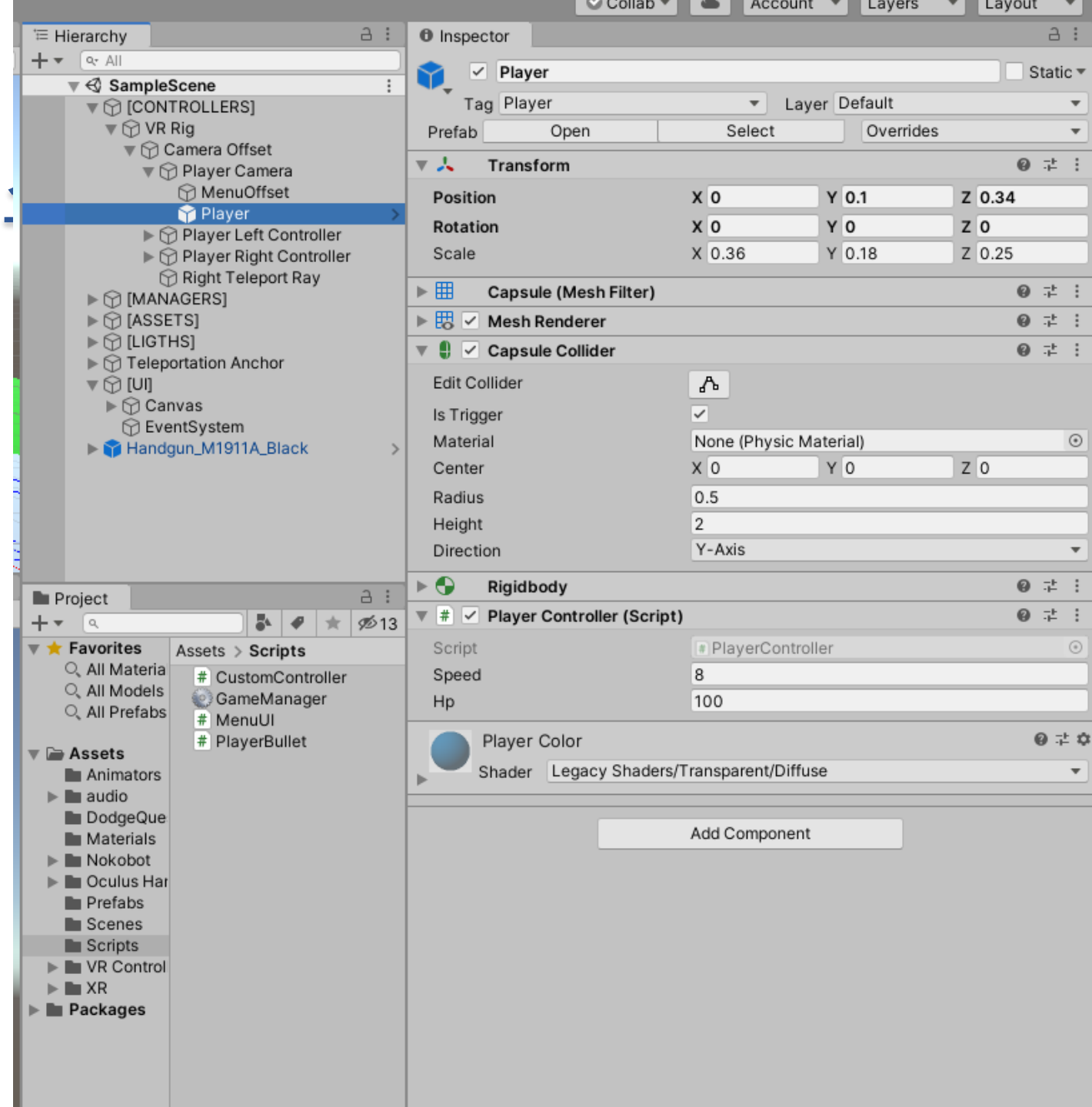
전투 시스템

- PlayerController 스크립트도 Update 에서 키보드 입력을 통한 이동 부분 비활성화 시키기!
- 그냥 조작시 조이스틱 입력이 적용되어서 이상한 결과가 나옴!

```
public class PlayerController : MonoBehaviour
{
    private Rigidbody playerRigidbody;
    public float speed = 8f;
    public float hp = 100.0f;
    // Start is called before the first frame update
    void Start()
    {
        playerRigidbody = GetComponent<Rigidbody>();
    }

    void Update()
    {
        //float xInput = Input.GetAxis("Horizontal");
        //float zInput = Input.GetAxis("Vertical");

        //float xSpeed = xInput * speed;
        //float zSpeed = zInput * speed;

        //Vector3 newVelocity = new Vector3(xSpeed, 0f, zSpeed);
        //playerRigidbody.velocity = newVelocity;
    }
    public void GetDamage(float amount)
    {
        hp -= amount;
        if (hp < 0)
        {
            Die();
        }
    }

    public void Die()
    {
        gameObject.SetActive(false);

        FindObjectOfType<GameManager>().EndGame();
    }
}
```

전투 시

- 플레이어 최종 정리
- Capsule Collider 의 Is Trigger 활성화!
- Mesh Render 비활성화 (반투명한 파란색이 더 이상 안나오도록!)



전투 시스템 만들기!

- 플레이어가 죽었을때,
  BulletSpawner가 더 이상 공격 하지
  않도록 하기!
- BulletSpawner.cs 파일 수정

```
31      // Update is called once per frame
32      void Update()
33      {
34          timeAffterSpawn += Time.deltaTime;
35
36          if(timeAffterSpawn >= spwanRate)
37          {
38              timeAffterSpawn = 0f;
39
40              if (!FindObjectOfType<GameManager>().isGameOver)
41              {
42                  GameObject bullet = Instantiate(bulletPrefab, transform.position, transform.rotation);
43                  bullet.transform.LookAt(target);
44                  fireAudio.PlayOneShot(fireClip);
45              }
46
47              spwanRate = Random.Range(spwanRateMin, spwanRateMax);
48          }
49
50      }
51
```

전투 시

- 결과 점검!