Multi-game2

발표자: 김원렬

목차

01
RPC

RPC 개념 재정리

02
UI

UI 만들기

03
Background

04
Prefab

Player&Bullet의
Inspector, Script

01

RPC

RPC 개념 재정리

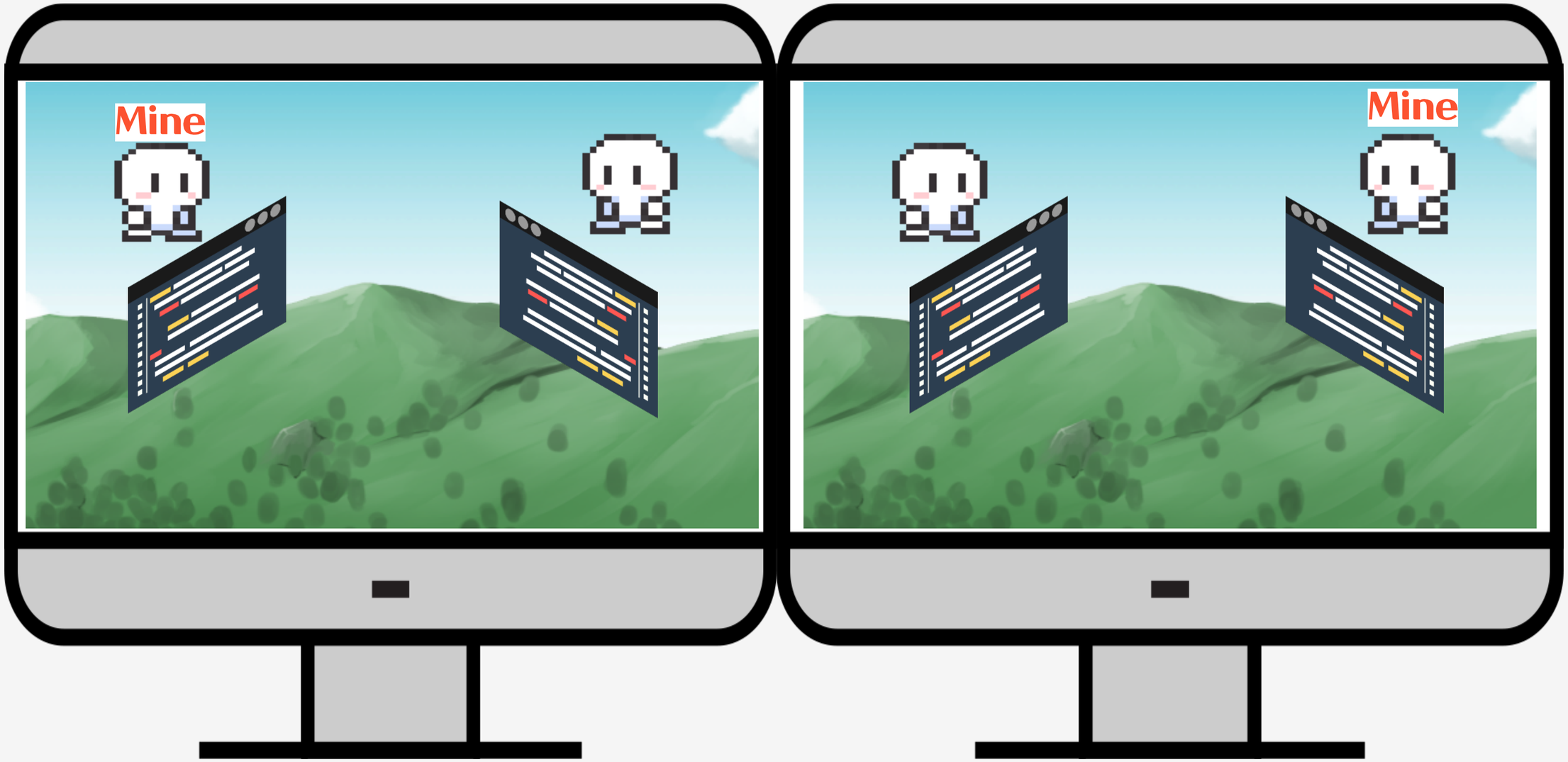
RPC

```
if (PV.IsMine)
{
    float axis = Input.GetAxisRaw("Horizontal");
    transform.Translate(new Vector3(axis * Time.deltaTime * 7, 0, 0));

    //RPC 함수는 모든 플레이어에게 정해진 함수를 실행하도록 하는 함수
    //AllBuffered를 사용하여 버퍼에 마지막 종료 모습?을 저장해놓고 다시 불러올 때 사용
    if (axis != 0) PV.RPC("FlipXRPC", RpcTarget.AllBuffered, axis);
}
```

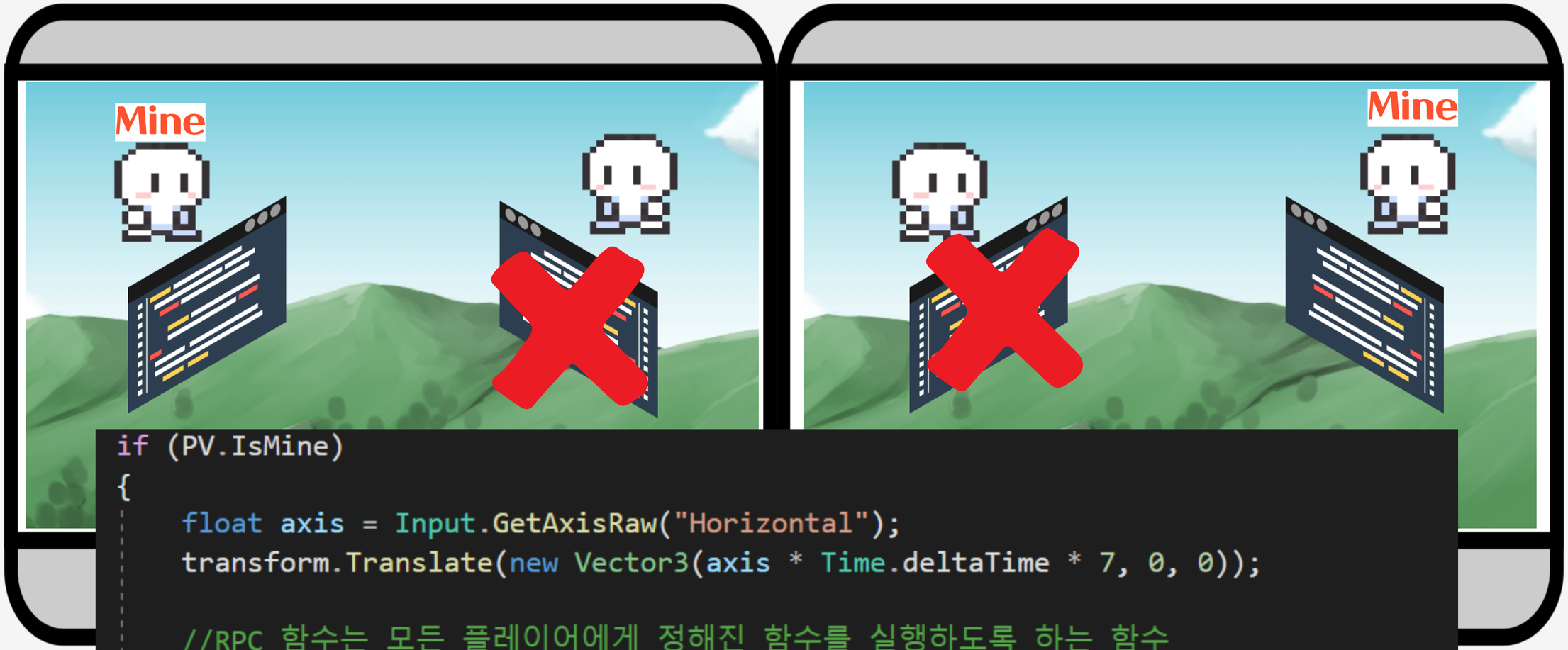
RPC 함수로 플레이어 좌우회전을 구현했다.

```
[PunRPC]
참조 0개
void FlipXRPC(float axis)
{
    SR.flipX = axis == -1;
}
```

RPC



RPC

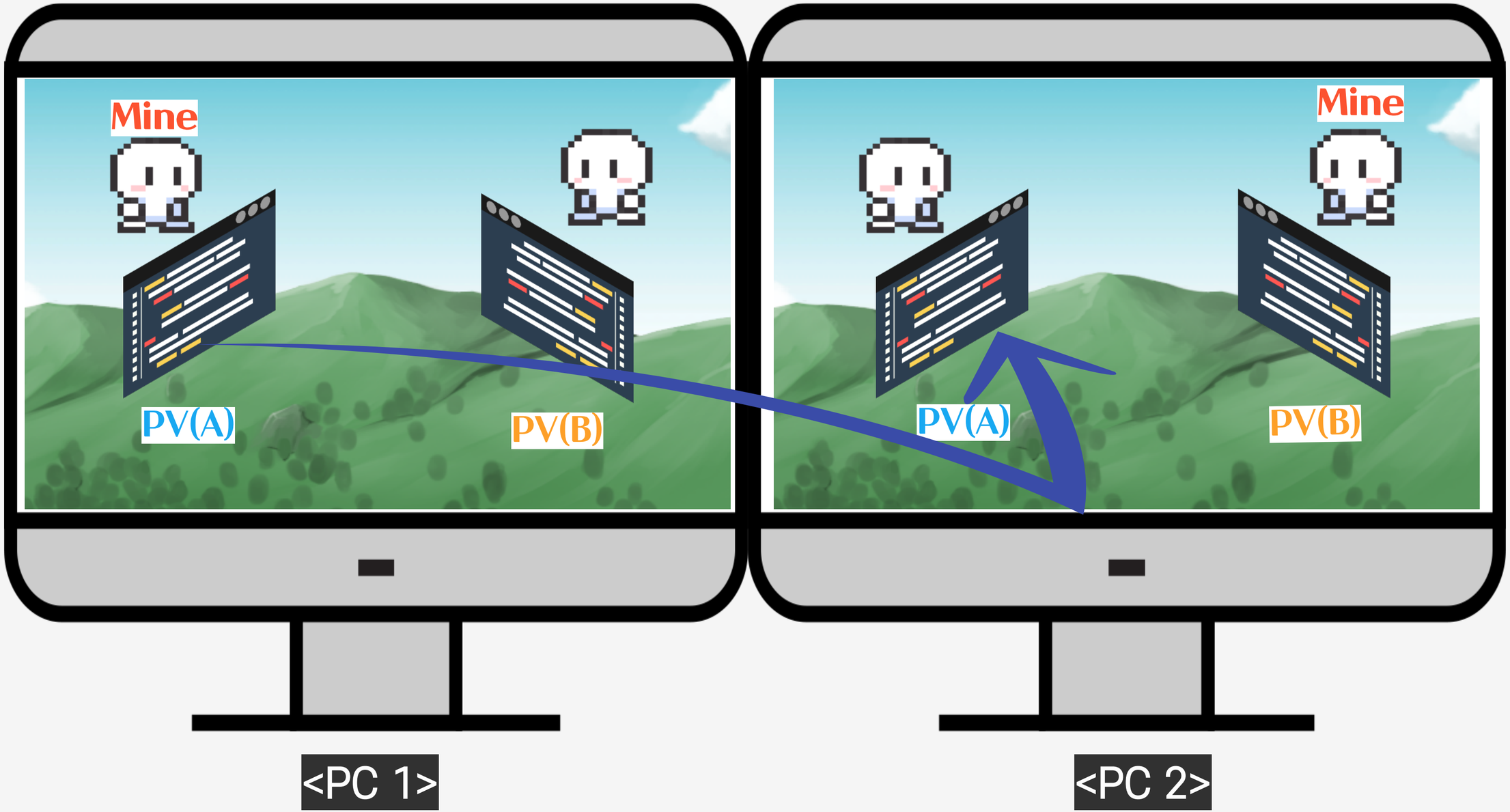



```
if (PV.IsMine)
{
    float axis = Input.GetAxisRaw("Horizontal");
    transform.Translate(new Vector3(axis * Time.deltaTime * 7, 0, 0));

    //RPC 함수는 모든 플레이어에게 정해진 함수를 실행하도록 하는 함수
    //AllBuffered를 사용하여 버퍼에 마지막 종료 모습?을 저장해놓고 다시 불러올 때 사용
    if (axis != 0) PV.RPC("FlipXRPC", RpcTarget.AllBuffered, axis);
}
```

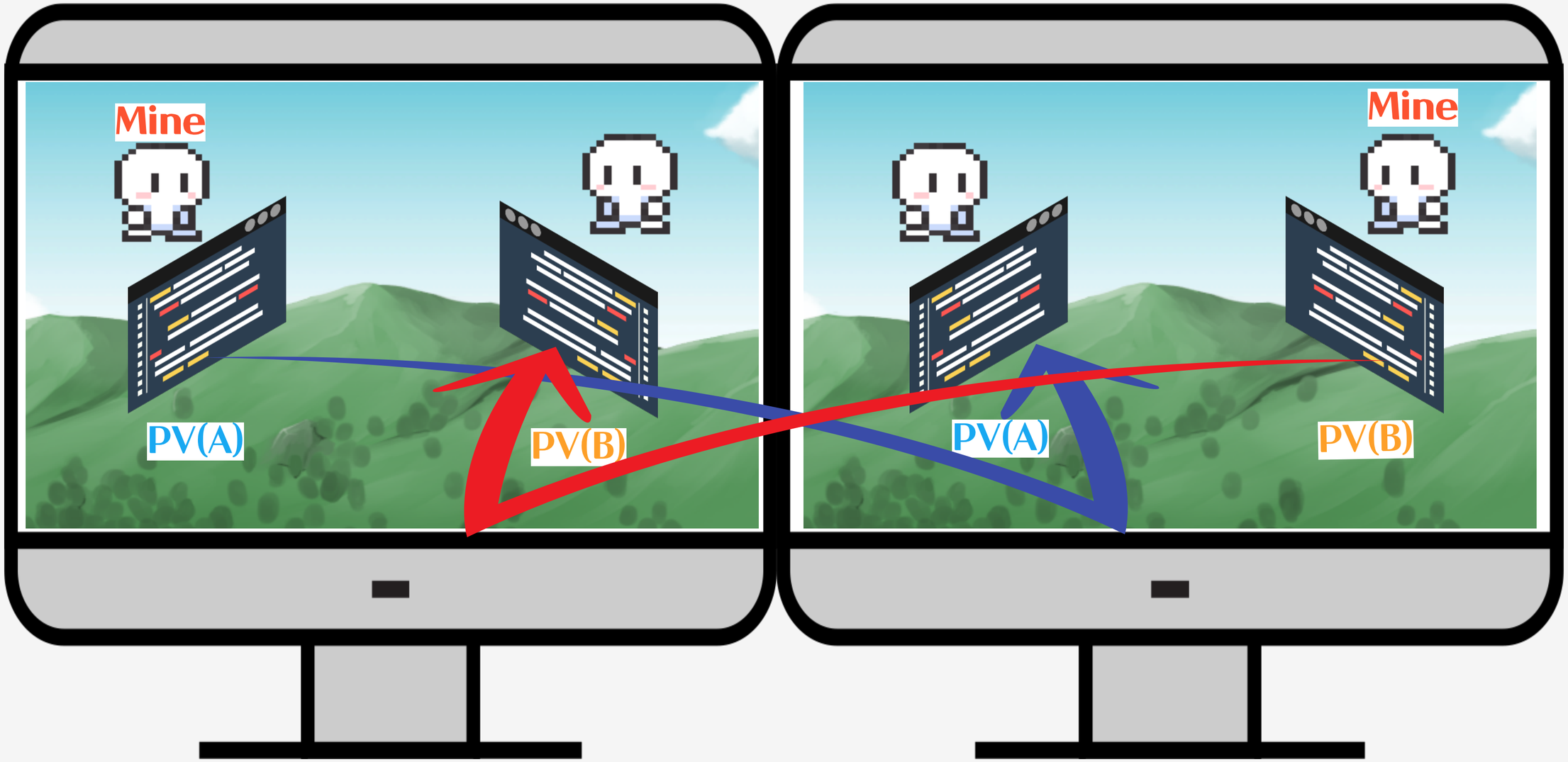
<PC 1>

<PC 2>

RPC



RPC

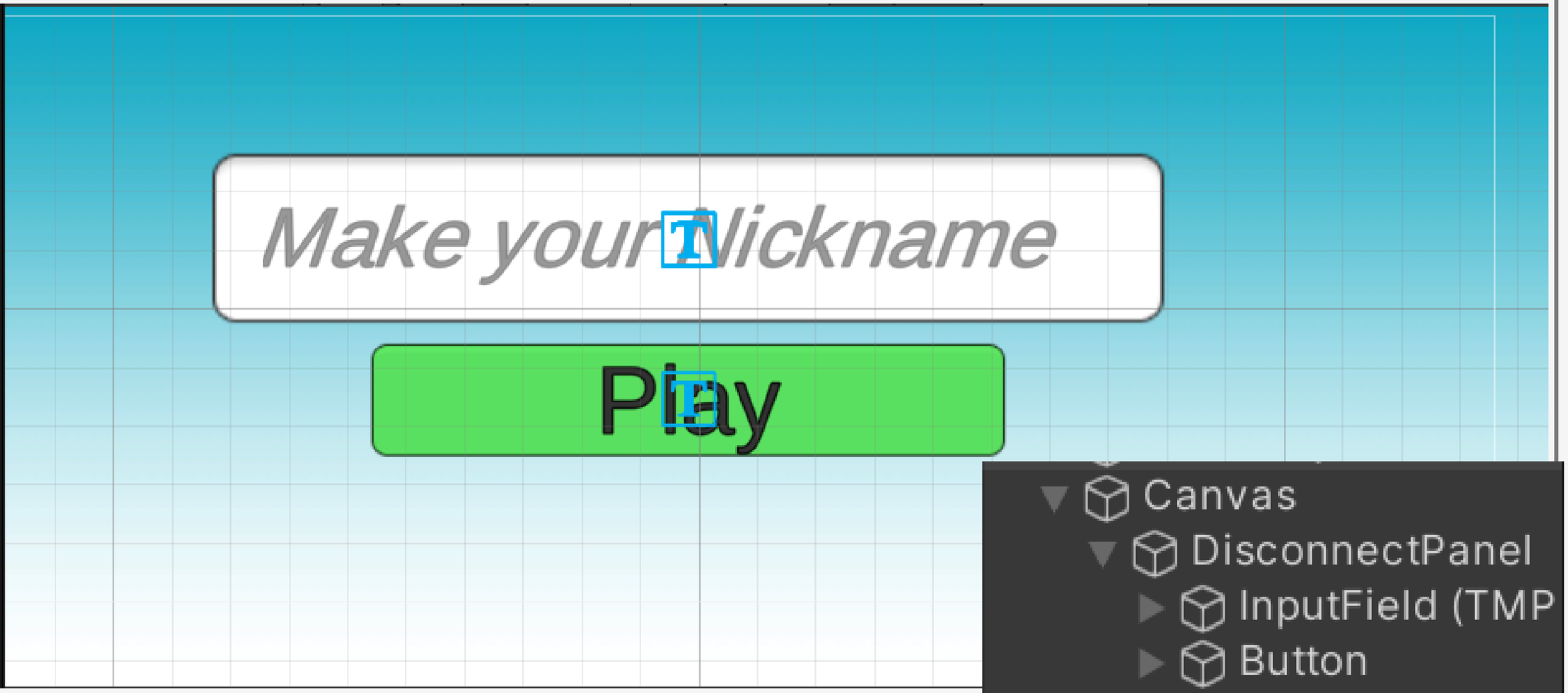


02

UI

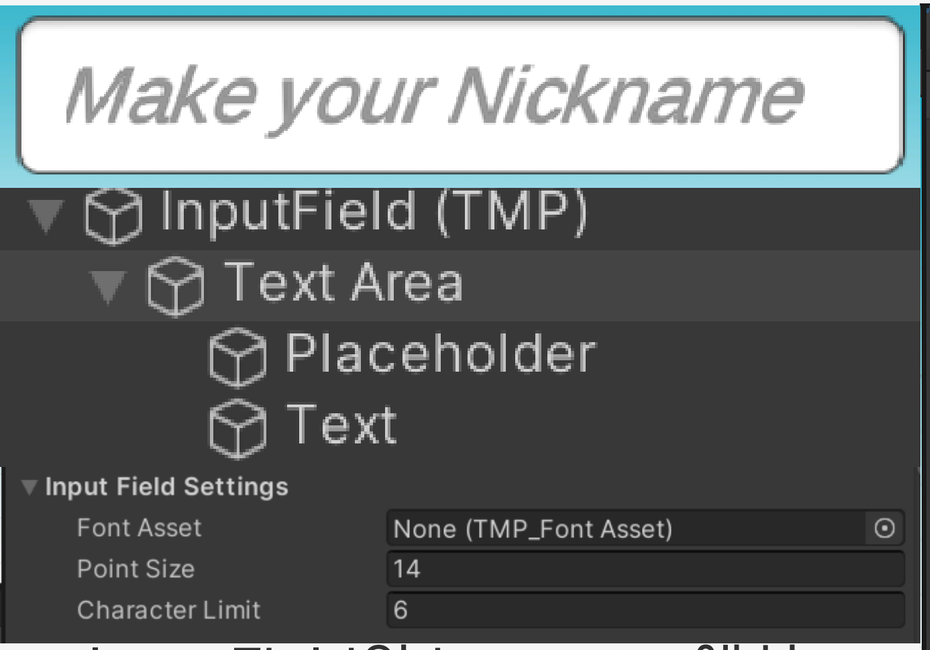
UI 만들기

UI(Disconnect Panel)

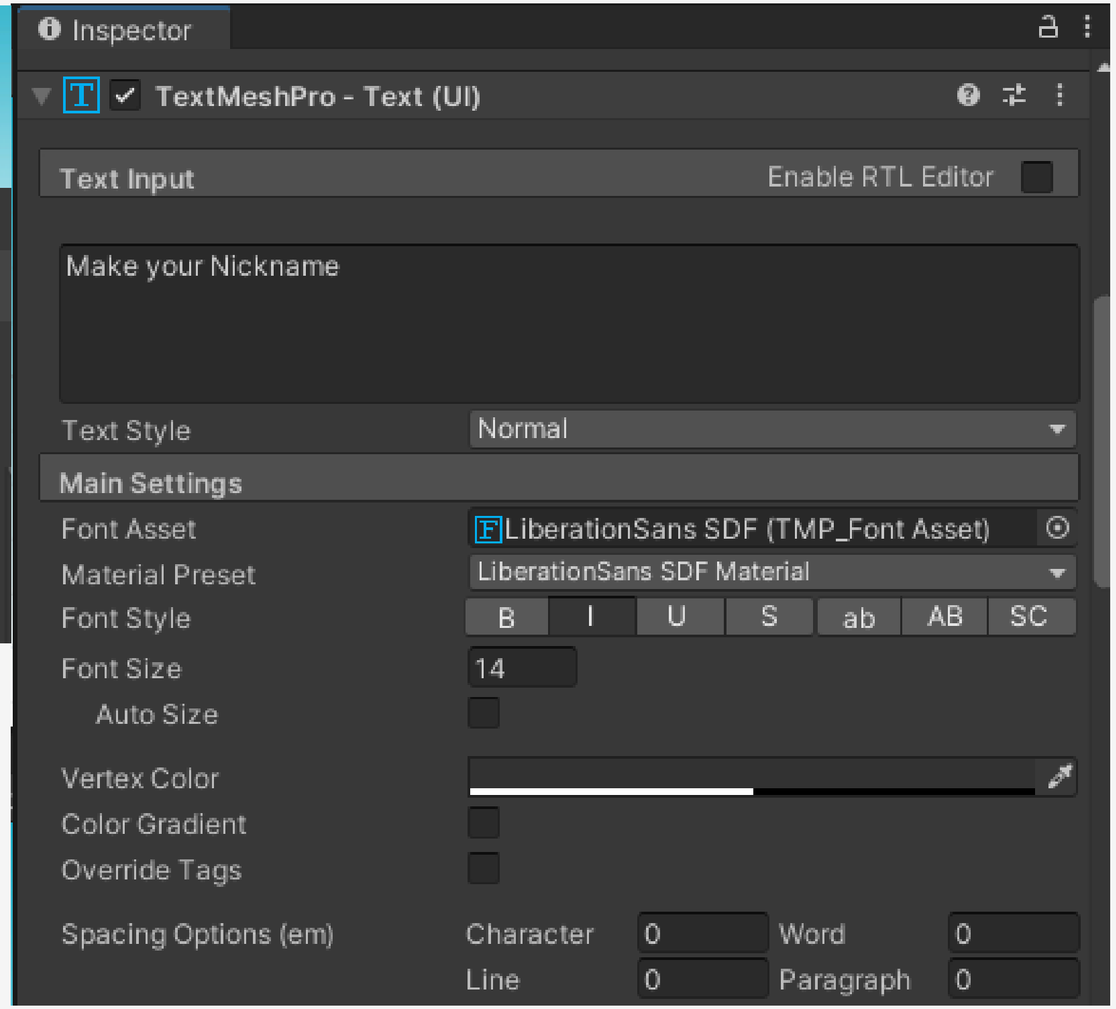


Canvas를 만든 후 UI의 배경이 될 Image를 넣고
그 안에 InputField와 Button을 만들어주었다.

UI(Disconnect Panel)

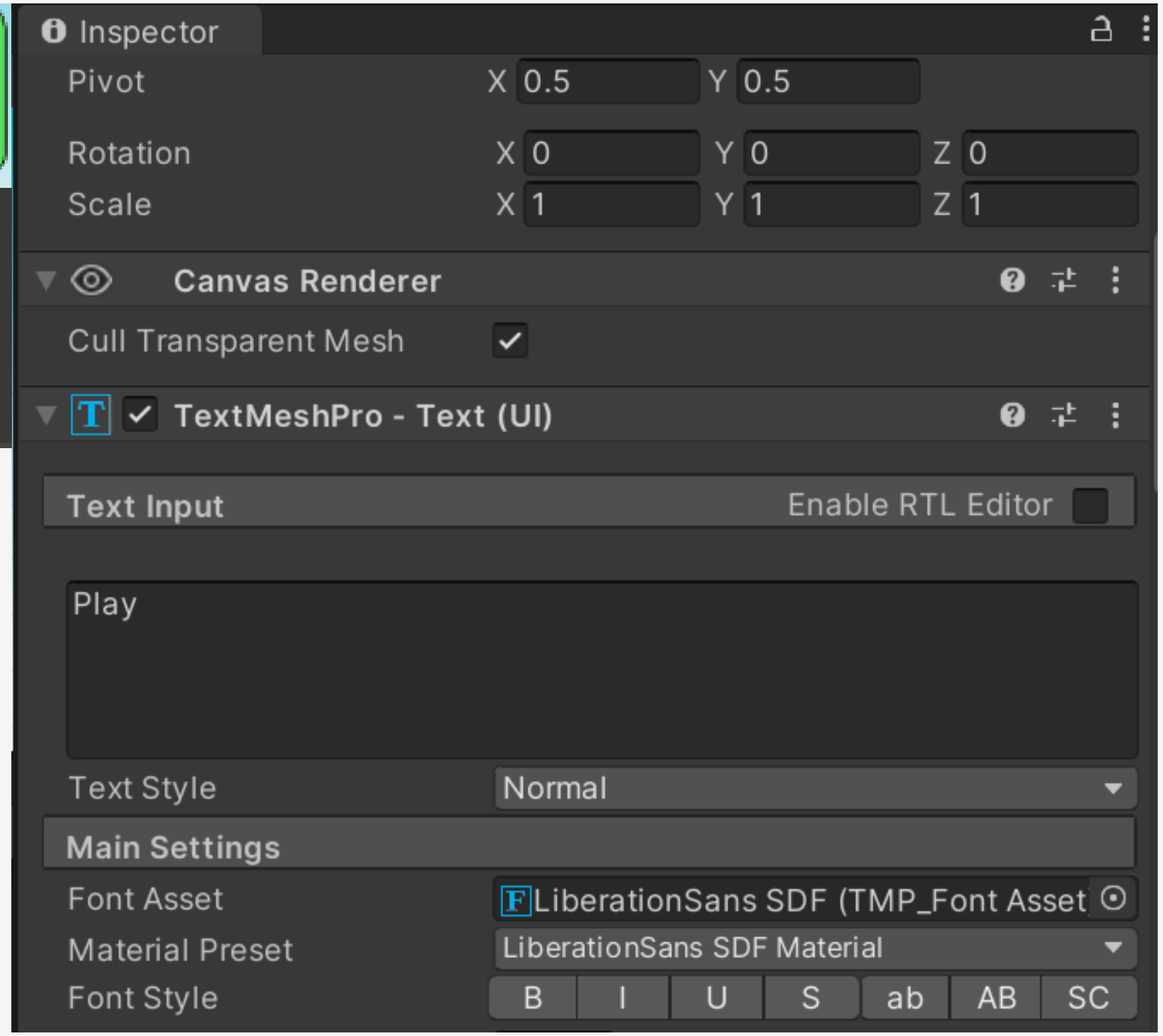


InputField의 inspector에서
글자 제한 설정



InputField를 만들면 자동으로 생기는 구성이고
Placeholder 부분에 닉네임을 만들라는 text를 넣었다.

UI(Disconnect Panel)



Button은 색을 바꾸고 원하는 text를 입력했다
TMP라고 돼있는 건 TextMeshPro 글자를 선명하게 보이게 해준다

UI(Respawn Panel)

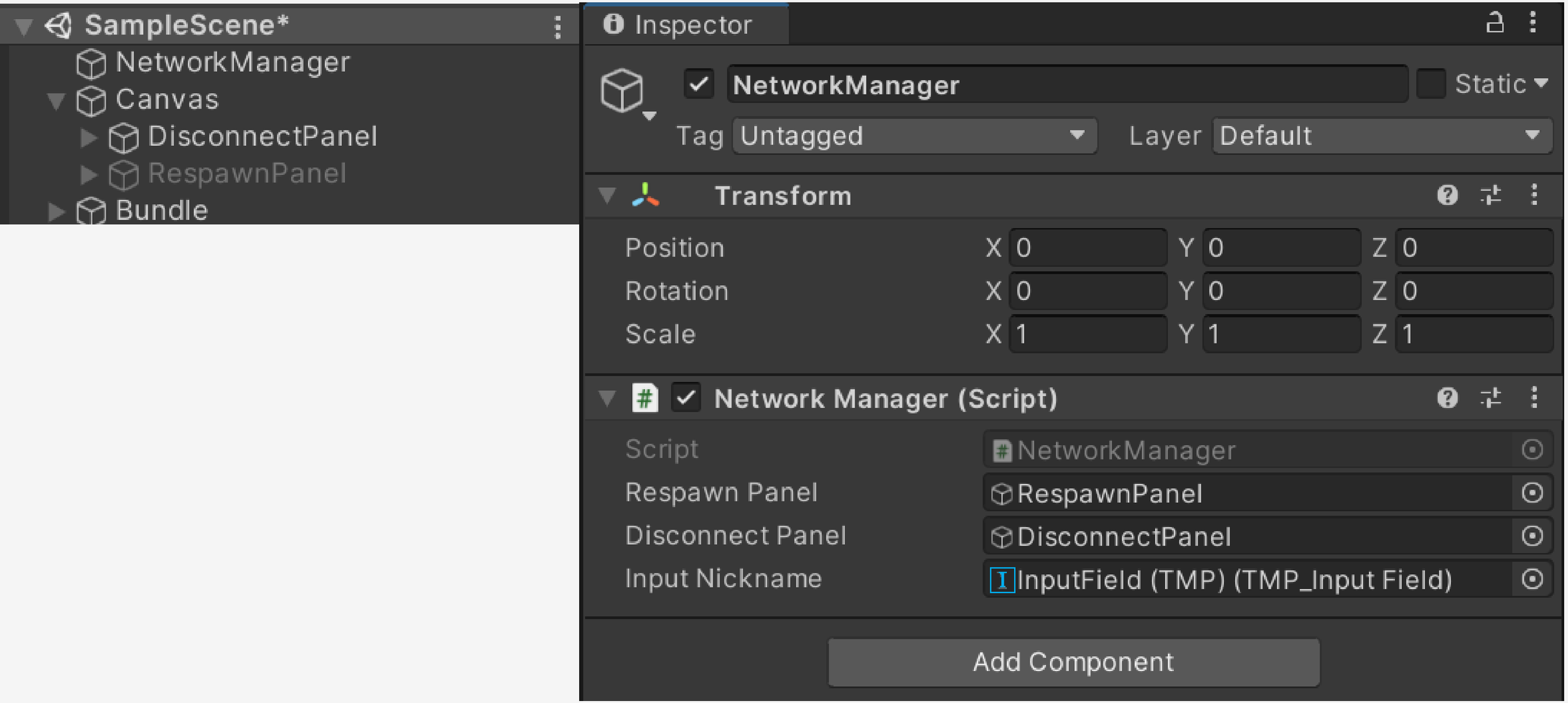
누르면 Respawn할 수 있는 버튼과
안내 메세지를 위한 Text UI로 구성했다.

UI(NetworkManger)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Photon.Pun;
using Photon.Realtime;
using UnityEngine.UI;
using TMPro;

Unity 스크립트|참조 0개
public class NetworkManager : MonoBehaviourPunCallbacks
{
    public GameObject RespawnPanel;
    public GameObject DisconnectPanel;
    public TMP_InputField InputNickname;


    Unity 메시지|참조 0개
    void Awake()
    {
        //server로 보내는 데이터율 설정? 잘 모르겠다. 쓰면 좋다고 한다..
        PhotonNetwork.SendRate = 60;
        PhotonNetwork.SerializationRate = 30;
        Screen.SetResolution(960, 540, false);
    }
```

Script에서 사용할 Object들을 public 변수로 선언해주고 Awake()로 시작할 때 설정을 해주었다.

UI(NetworkManger)



Object를 만든 후 Script를 넣어주어 NetworkManager 역할을 부여해주었다.
그리고 선언해놓은 변수들에 알맞은 Object들을 끌어놓아 할당해주었다.

UI(NetworkManger)

```
private void Update()
{
    //접속 중에 ESC 버튼을 누르면 접속 종료
    if (Input.GetKeyDown(KeyCode.Escape) && PhotonNetwork.IsConnected) PhotonNetwork.Disconnect();
}

참조 0개
public void Connect() => PhotonNetwork.ConnectUsingSettings();
참조 14개
public override void OnConnectedToMaster()
{
    PhotonNetwork.LocalPlayer.NickName = InputNickname.text;
    PhotonNetwork.JoinOrCreateRoom("Room", new RoomOptions { MaxPlayers = 6 }, null);
}

참조 21개
public override void OnJoinedRoom()
{
    PhotonNetwork.Instantiate("Player", Vector3.zero, Quaternion.identity);
    DisconnectPanel.SetActive(false);
}
```

Update로 종료 조건을 검사해주고
Connect 됐을 때의 함수들과 Connect 함수를 만들었다.

UI(NetworkManger)

```
private void Update()
{
    //접속 중에 ESC 버튼을 누르면 접속 종료
    if (Input.GetKeyDown(KeyCode.Escape) && PhotonNetwork.IsConnected) PhotonNetwork.Disconnect();
}

참조 0개
public void Connect() => PhotonNetwork.ConnectUsingSettings();
참조 14개
public override void OnConnectedToMaster()
{
    PhotonNetwork.LocalPlayer.NickName = InputNickname.text;
    PhotonNetwork.JoinOrCreateRoom("Room", new RoomOptions { MaxPlayers = 6 }, null);
}

참조 21개
public override void OnJoinedRoom()
{
    PhotonNetwork.Instantiate("Player", Vector3.zero, Quaternion.identity);
    DisconnectPanel.SetActive(false);
}
```

Instantiate는 객체를 모든 게임에 생성해준다.

UI(NetworkManger)

```
private void Update()
{
    //접속 중에 ESC 버튼을 누르면 접속 종료
    if (Input.GetKeyDown(KeyCode.Escape) && PhotonNetwork.IsConnected) PhotonNetwork.Disconnect();
}

참조 0개
public void Connect() => PhotonNetwork.ConnectUsingSettings();
참조 14개
public override void OnConnectedToMaster()
{
    PhotonNetwork.LocalPlayer.NickName = InputNickname.text;
    PhotonNetwork.JoinOrCreateRoom("Room", new RoomOptions { MaxPlayers = 6 }, null);
}

참조 21개
public override void OnJoinedRoom()
{
    PhotonNetwork.Instantiate("Player", Vector3.zero, Quaternion.identity);
    DisconnectPanel.SetActive(false);
}
```

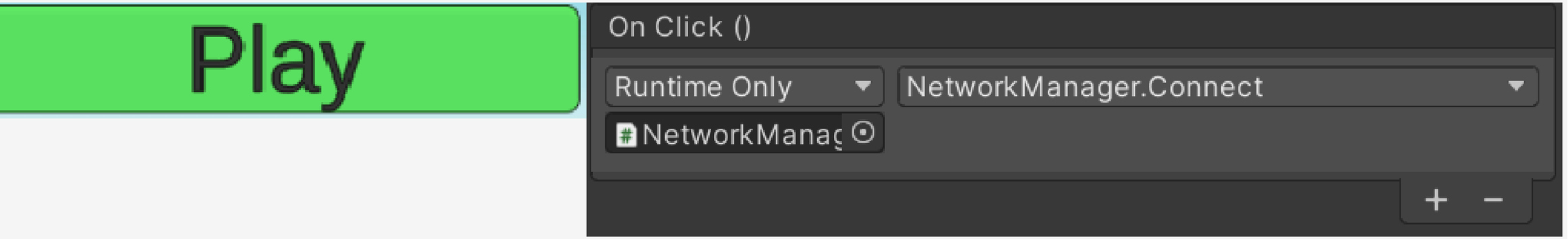
Instantiate는 객체를 모든 게임에 생성해준다.

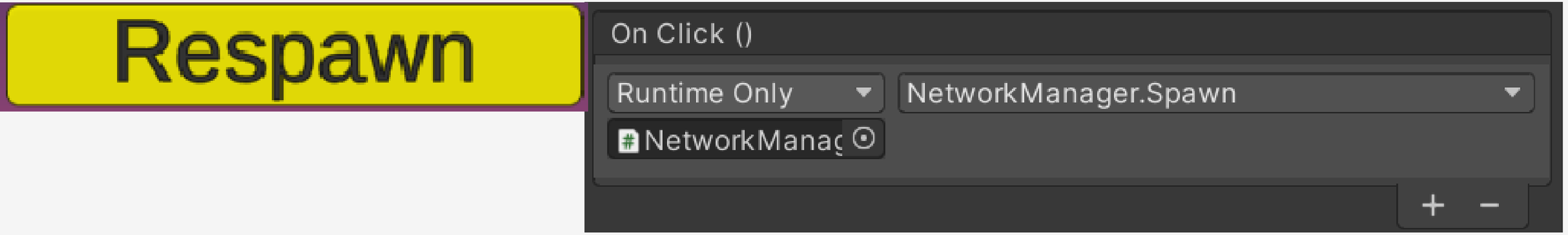
UI(NetworkManger)

```
public void Spawn()
{
    PhotonNetwork.Instantiate("Player", Vector3.zero, Quaternion.identity);
    RespawnPanel.SetActive(false);
}

참조 16개
public override void OnDisconnected(DisconnectCause cause)
{
    DisconnectPanel.SetActive(true);
    RespawnPanel.SetActive(false);
}
```

Respawn을 눌렀을 때의 Spawn함수와
Disconnect될 때의 함수를 만들었다.

UI(NetworkManger)



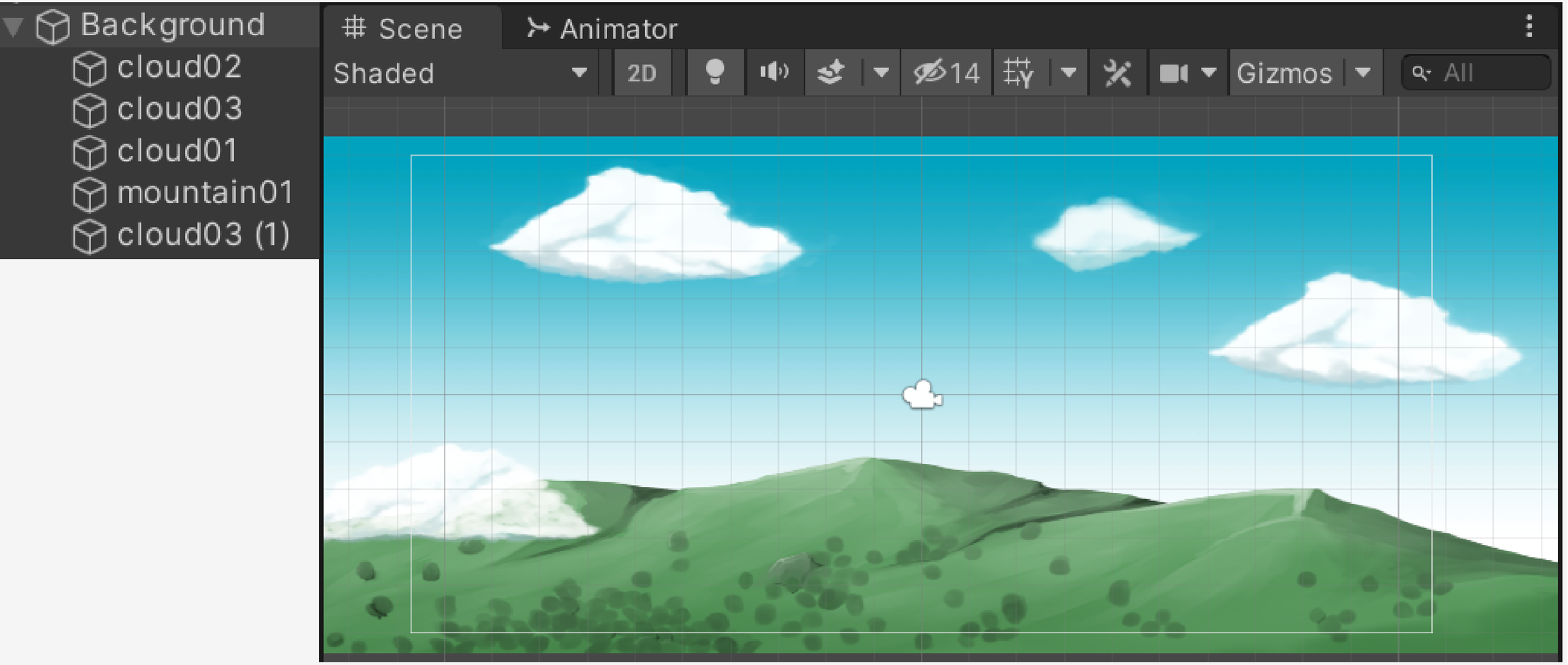


각 버튼의 컴포넌트에 On Click시 실행할 함수를 넣어주었다.

03

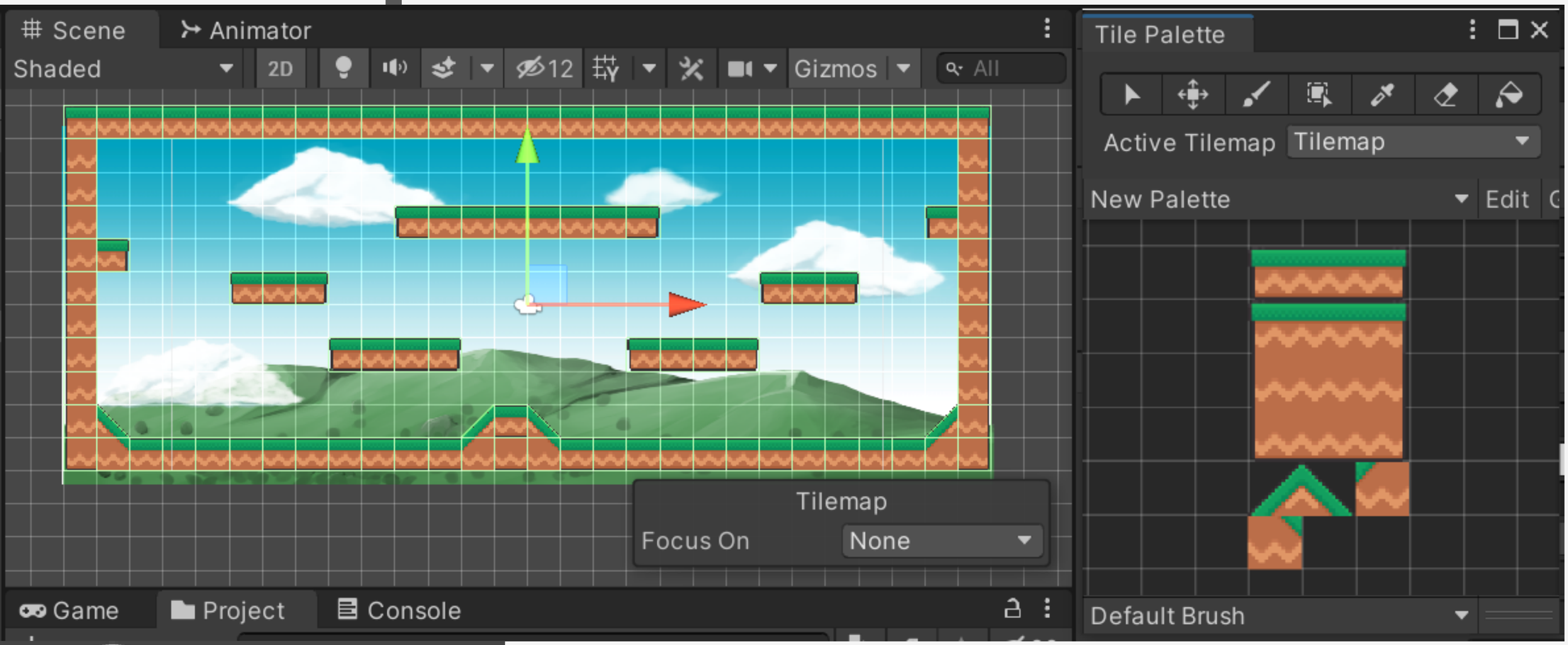
Background

Background



AssetStore에 있는 무료 에셋 Image들을 활용해서 배경을 만들었다.

Tilemap



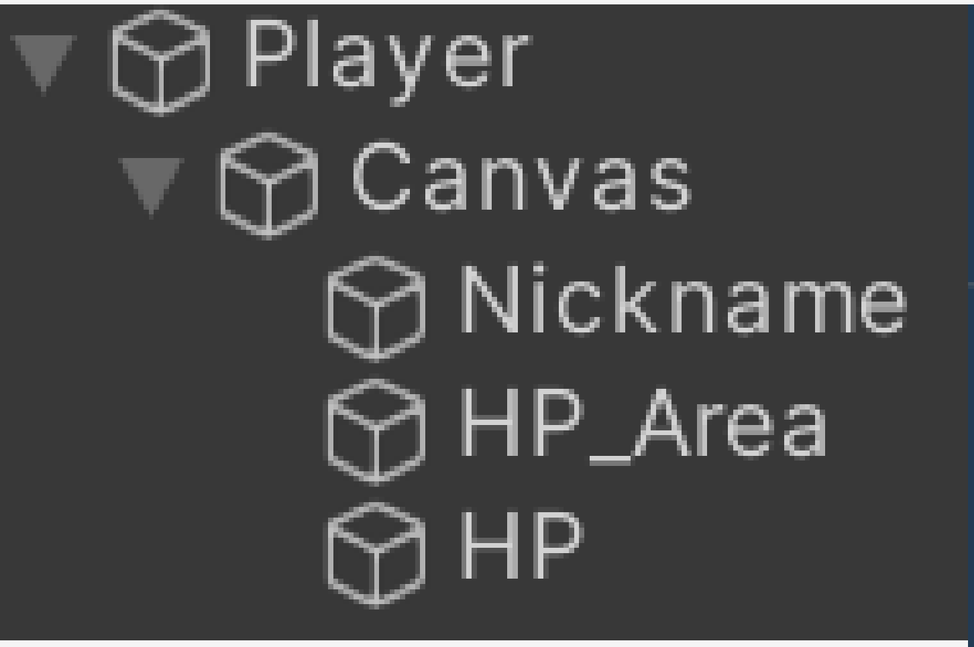
TilePallete를 사용하여 물리법칙이 작용할 맵을 만들어주었다.
Tilemap에는 Rigid body를 적용하여
다른 오브젝트와 물리법칙이 적용되게 하였다.

04

Prefab

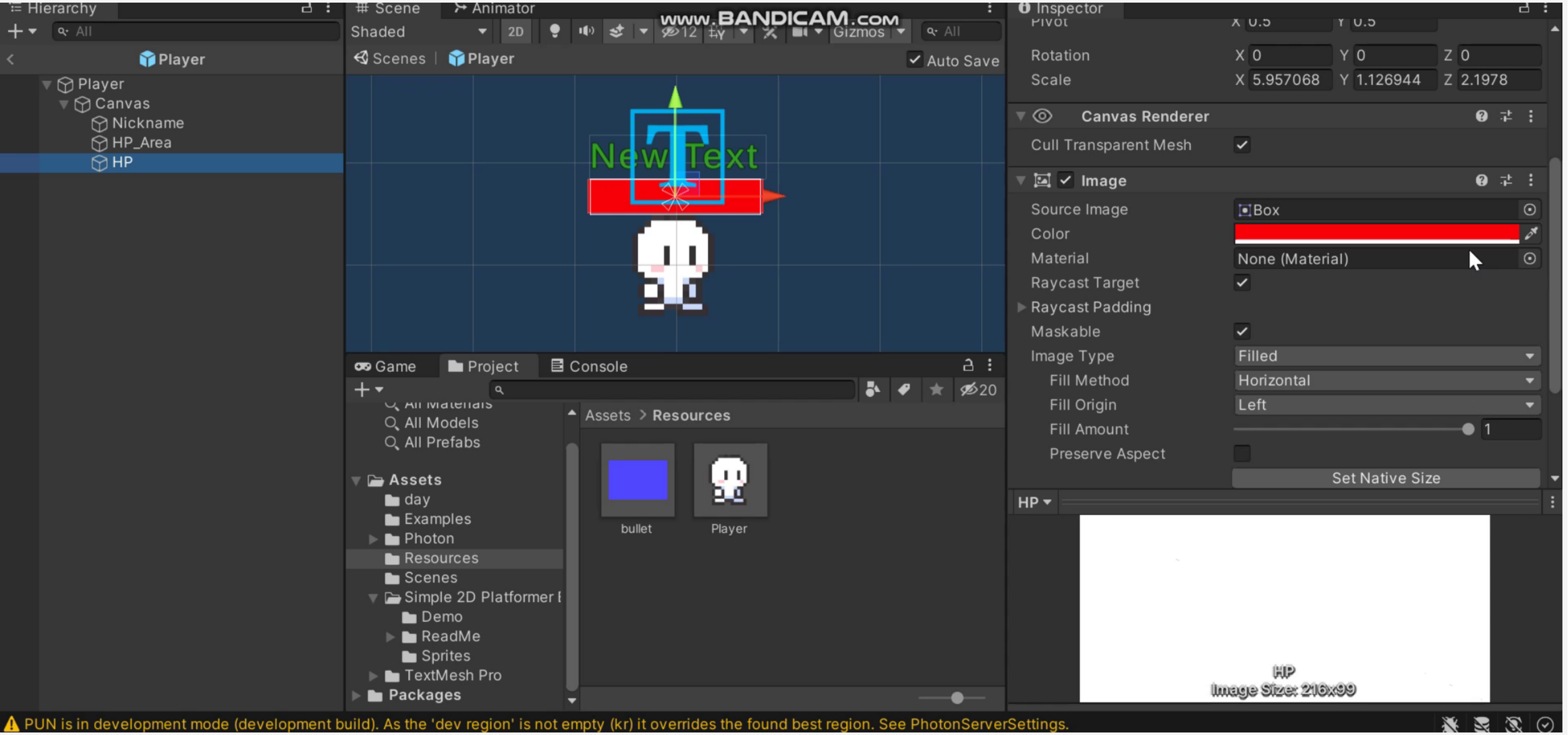
Player&Bullet의 Inspector, Script,

Player

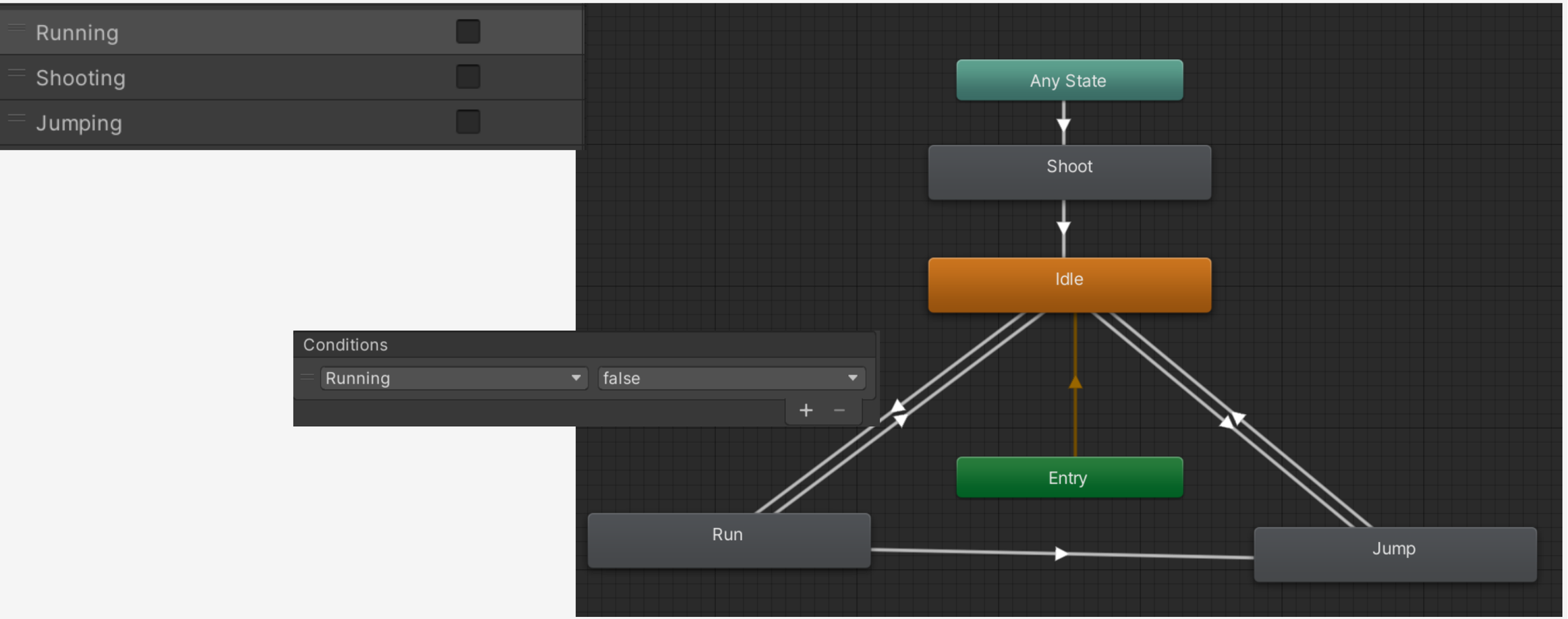


닉네임이 들어갈 Text UI와
빨간색, 하얀색의 Box Image로 HP를 표현했다.

Player

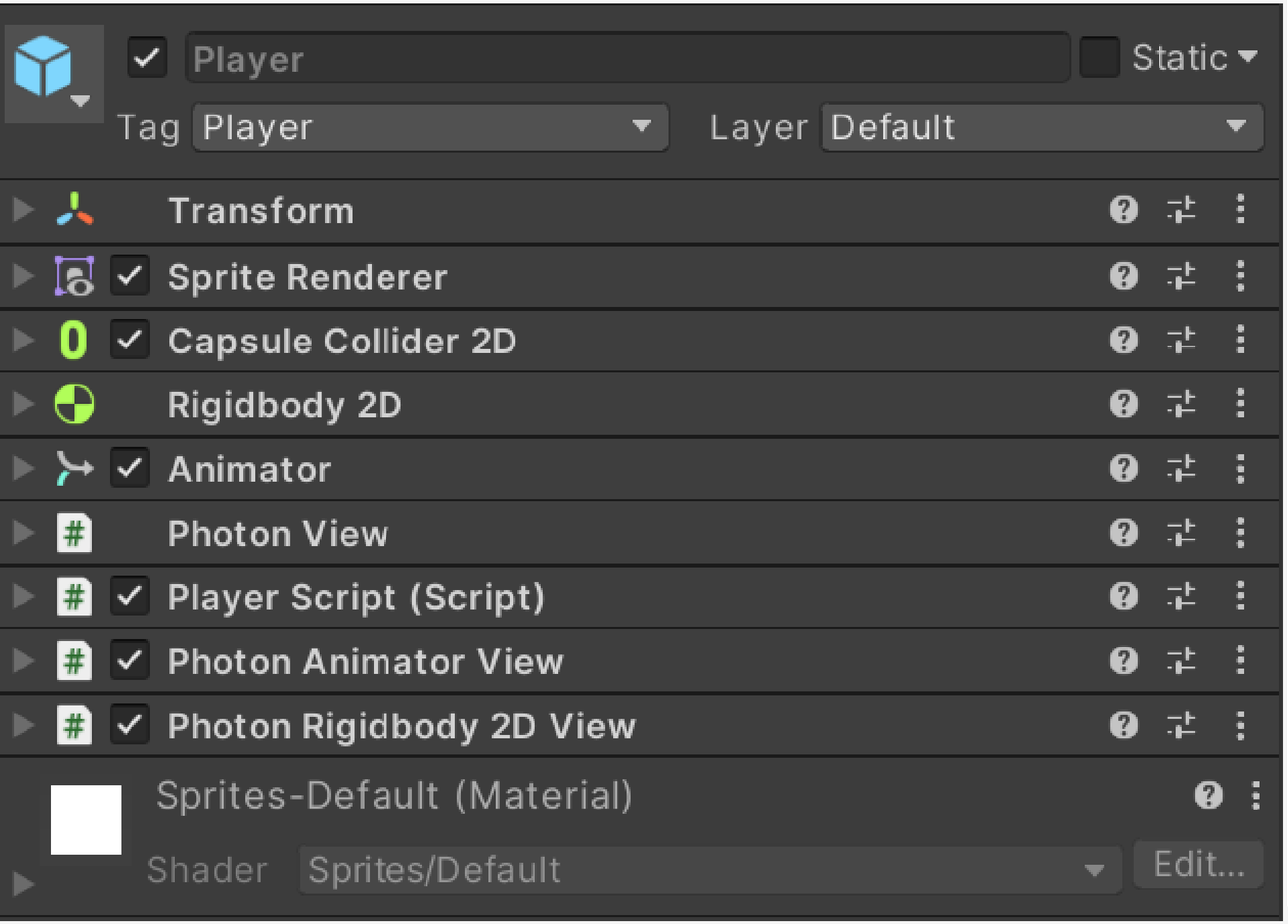

Hierarchy
Player
Canvas
Nickname
HP_Area
HP
Scene
Animator
Shaded
2D
Gizmos
Scenes
Player
Auto Save
New Text
Game
Project
Console
All Models
All Prefabs
Assets
day
Examples
Photon
Resources
Scenes
Simple 2D Platformer
Demo
ReadMe
Sprites
TextMesh Pro
Packages
Assets > Resources
bullet
Player
Inspector
Rotation
Scale
Canvas Renderer
Cull Transparent Mesh
Image
Source Image
Box
Color
Material
None (Material)
Raycast Target
Raycast Padding
Maskable
Image Type
Filled
Fill Method
Horizontal
Fill Origin
Left
Fill Amount
Preserve Aspect
Set Native Size
HP
Image Size: 216x99
PUN is in development mode (development build). As the 'dev region' is not empty (kr) it overrides the found best region. See PhotonServerSettings.

Player



필요한 Animation들과 각 Animation에 Transition을 만들고
Transtion을 조종하기 위해 변수를 만들어 주었다.

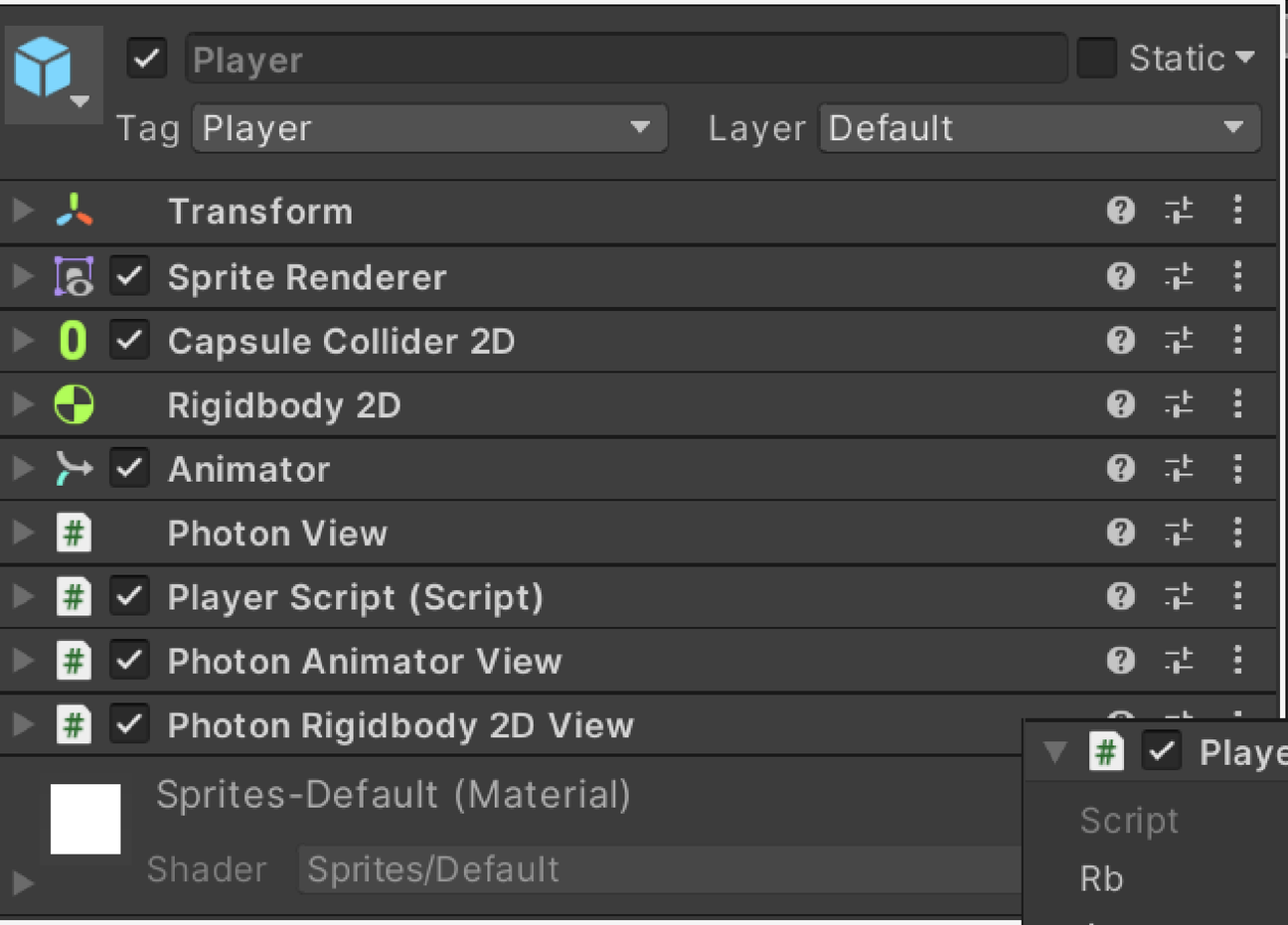
Player



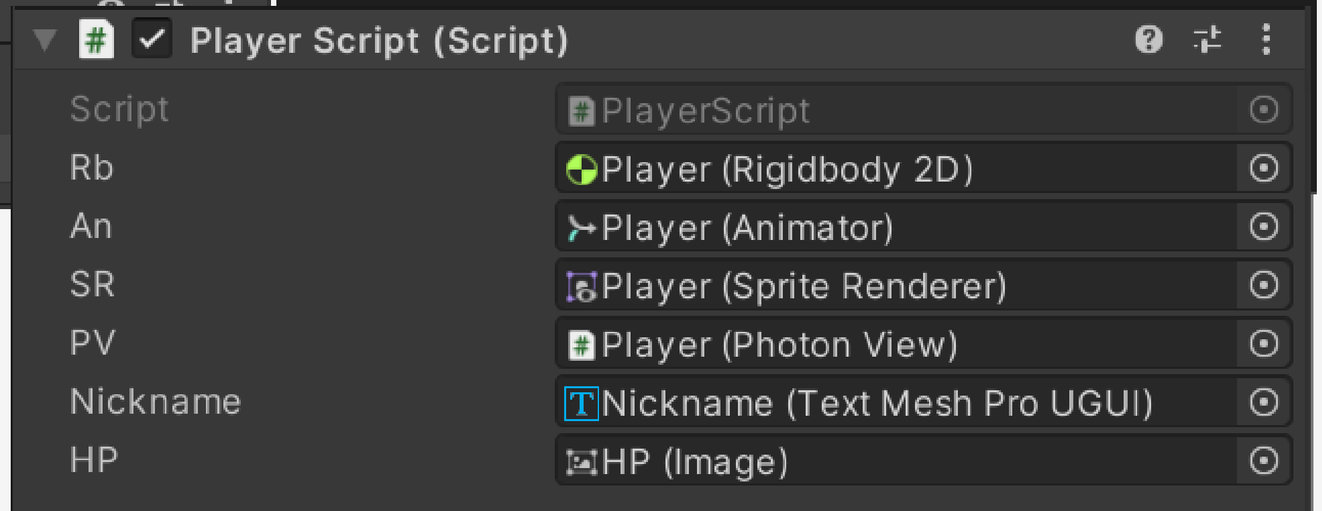

```
public class PlayerScript : Mo
{
    public Rigidbody2D Rb;
    public Animator An;
    public SpriteRenderer SR;
    public PhotonView PV;
    public TMP_Text Nickname;
    public Image HP;

    bool isGround;
    Vector3 curPos;
```

Player에 필요한 컴포넌트들을 추가해주고
Script에서 사용하기 위해 변수화 했다.

Player

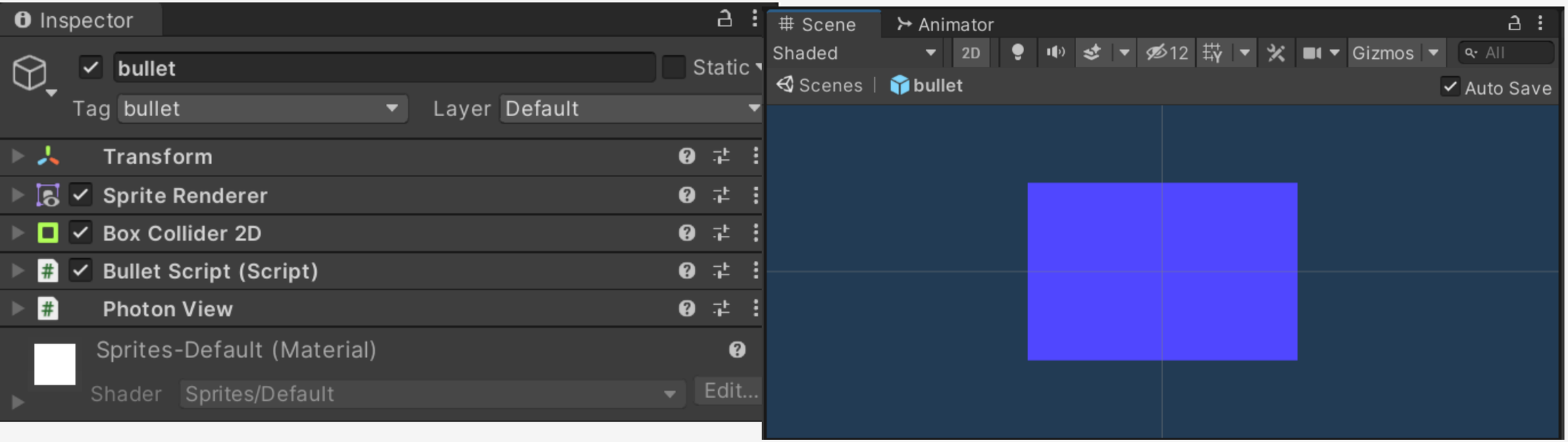



```
public class PlayerScript : Mo
{
    public Rigidbody2D Rb;
    public Animator An;
    public SpriteRenderer SR;
    public PhotonView PV;
    public TMP_Text Nickname;
    public Image HP;
```



Public 변수에
알맞은 컴포넌트 할당 필수

Bullet



Bullet은 파란색 Box Image로 생성했다.
필요한 컴포넌트들도 할당해주었다.

Player

```
void Awake()
{
    //Nickname
    Nickname.text = PV.IsMine ? PhotonNetwork.NickName : PV.Owner.NickName;
    Nickname.color = PV.IsMine ? Color.green : Color.red;
}
```

처음 Script가 실행될 때 Script를 가진 Player에 맞는 닉네임 할당
자신에게는 초록색으로 보이고 남에게는 빨간색으로 보인다.

Player

```
//Master면 실행
if (PV.IsMine)
{
    //이동
    float axis = Input.GetAxisRaw("Horizontal");
    Rb.velocity = new Vector2(4 * axis, Rb.velocity.y);


    //좌우반전
    //RPC 함수는 모든 플레이어에게 정해진 함수를 실행하도록 하는 함수
    //AllBuffered를 사용하여 버퍼에 마지막 종료 모습?을 저장해놓고 다시 불러올 때 사용
    if (axis != 0)
    {
        An.SetBool("Running", true);
        PV.RPC("FlipXRPC", RpcTarget.AllBuffered, axis);
    }
    else An.SetBool("Running", false);
```

```
[PunRPC]
참조 0개
void FlipXRPC(float axis) => SR.flipX = axis == -1;
```

플레이어 이동, 좌우회전을 Update에 넣어 변화가 있으면 바로 적용한다

Player

```
//Jump
isGround = Physics2D.OverlapCircle((Vector2)transform.position + new Vector2(0, -0.5f), 0.07f, 1 << LayerMask.NameToLayer("Ground"));
An.SetBool("Jumping", !isGround);
if (Input.GetKeyDown(KeyCode.UpArrow) && isGround) PV.RPC("JumpRPC", RpcTarget.All);

//Shoot
if (Input.GetKeyDown(KeyCode.Space))
{
    //GameObject 반환
    PhotonNetwork.Instantiate("bullet", transform.position + new Vector3(SR.flipX ? 0.4f : 0.4f, -0.11f, 0), Quaternion.identity)
    .GetComponent<PhotonView>().RPC("DirRPC", RpcTarget.All, SR.flipX ? -1 : 1);
    An.SetTrigger("Shooting");
}
```

```
[PunRPC]
참조 0개
void JumpRPC()
{
    Rb.velocity = Vector2.zero;
    Rb.AddForce(Vector2.up * 700);
}

[PunRPC]
참조 0개
void DestroyRPC() => Destroy(gameObject);
```

isGround로 공중인지 확인하고 아니라면 Jump

방향에 따라 bullet 객체를 만들어주고
만들어진 객체는 bullet.cs에서 control

Player

```
//Jump
isGround = Physics2D.OverlapCircle((Vector2)transform.position + new Vector2(0, -0.5f), 0.07f, 1 << LayerMask.NameToLayer("Ground"));
An.SetBool("Jumping", !isGround);
if (Input.GetKeyDown(KeyCode.UpArrow) && isGround) PV.RPC("JumpRPC", RpcTarget.All);

//Shoot
if (Input.GetKeyDown(KeyCode.Space))
{
    //GameObject 반환
    PhotonNetwork.Instantiate("bullet", transform.position + new Vector3(SR.flipX ? 0.4f : 0.4f, -0.11f, 0), Quaternion.identity)
    .GetComponent<PhotonView>().RPC("DirRPC", RpcTarget.All, SR.flipX ? -1 : 1);
    An.SetTrigger("Shooting");
}
```

```
[PunRPC]
참조 0개
void JumpRPC()
{
    Rb.velocity = Vector2.zero;
    Rb.AddForce(Vector2.up * 700);
}

[PunRPC]
참조 0개
void DestroyRPC() => Destroy(gameObject);
```

isGround로 공중인지 확인하고 아니라면 Jump

방향에 따라 bullet 객체를 만들어주고
만들어진 객체는 bullet.cs에서 control

Bullet

```
public PhotonView PV;
int dir;

// Start is called before the first frame update
Unity 메시지 | 참조 0개
void Start()
{
    Destroy(gameObject, 3.5f);
}

// Update is called once per frame
Unity 메시지 | 참조 0개
void Update()
{
    transform.Translate(Vector3.right * 7 * Time.deltaTime * dir);
}
```

```
[PunRPC]
참조 0개
void DirRPC(int dir) => this.dir = dir;
```

만들어진 후부터 3.5초가 지나면 파괴된다.
만들어지고 난 후부터 Update가 계속 실행되기 때문에
방향대로 계속 직진한다.

Bullet

```
void OnTriggerEnter2D(Collider2D other)
{
    if (other.tag == "Ground") PV.RPC("DestroyRPC", RpcTarget.AllBuffered);
    if(!PV.IsMine && other.tag == "Player" && other.GetComponent<PhotonView>().IsMine)
    {
        other.GetComponent<PlayerScript>().Hit();
        PV.RPC("DestroyRPC", RpcTarget.AllBuffered);
    }
}
```

〈PlayerScript.cs〉

```
public void Hit()
{
    HP.fillAmount -= 0.1f;
    if (HP.fillAmount <= 0)
    {
        GameObject.Find("Canvas").transform.Find("RespawnPanel").gameObject.SetActive(true);
        PV.RPC("DestroyRPC", RpcTarget.AllBuffered);
    }

}
```

지형에 부딪히면 Bullet Object 파괴

Player에게 부딪히면 Player의 Hit 함수 실행

HP가 10%씩 줄어들고 0이 되면 RespawnPanel 활성화

Player

〈PlayerScript.cs〉

```
참조 12개
public void OnPhotonSerializeView(PhotonStream stream, PhotonMessageInfo info)
{
    //IsMine인 경우 쓴다,넘겨준다.
    if(stream.IsWriting)
    {
        stream.SendNext(transform.position);
        stream.SendNext(HP.fillAmount);
    }
    //IsMine이 아니면 받는다.
    else
    {
        curPos = (Vector3)stream.ReceiveNext();
        HP.fillAmount = (float)stream.ReceiveNext();

    }
}
```

각 PhotonView들의 변수 동기화
자기 PV라면 자신의 변수값을 다른 PV에게 넘겨주고
자신의 PV가 아니라면 값을 받는다.

Play

발표를 들어주셔서

감사합니다 :)