Game Engine Portfolio

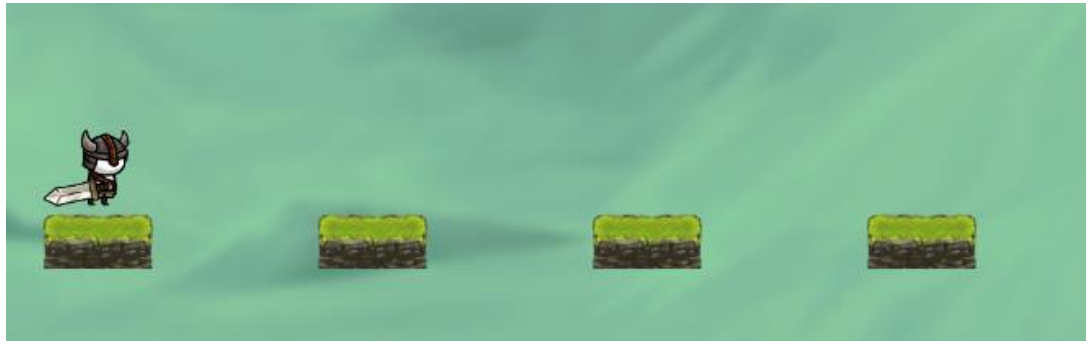
Joystick & Moving Implementation

Eugene Lee

Contents

1. Create Player

- Create Player Characters and simple terrain.
- Add Rigidbody 2D and Box Collider 2D components to player character.
- Check the z-axis of the Freeze Rotation in Rigidbody 2D to prevent the player from rotating the z-axis.
- Add the Box Collider2D Component to the terrain to stand on top of it.



```
new Rigidbody2D rigidbody2D;
Vector2 velocity = Vector2.zero;
Vector3 leftdir = new Vector3(-1, 1, 1);
public float moveSpeed = 3;
Joystick joystick;
Jump jump;
```

Add the variables to move the player character.

```
void Start()
{
    rigidbody2D = GetComponent<Rigidbody2D>();

    joystick = GameObject.FindObjectOfType<Joystick>();
    if (joystick != null)
    {
        joystick.Init();
    }

    jump = GameObject.FindObjectOfType<Jump>();
    if (jump != null)
    {
        jump.Init();
    }
}
```

Set the player script's start function to invoke the initialization function of other scripts.

```
public void Move(Vector2 dir)
{
    velocity = rigidbody2D.velocity;
    velocity.x = dir.x * moveSpeed;

    rigidbody2D.velocity = velocity;

    if (dir.x > 0)
        transform.localScale = Vector3.one;

    else if (dir.x < 0)
        transform.localScale = leftdir;
}

public void Jump()
{
    if (rigidbody2D.velocity.y != 0)
        return;
    rigidbody2D.AddForce(Vector2.up * 400);
}

void Update()
{
    Move(joystick.Dir);
}
```

- Add a function to move the character
- Use velocity.x because it will only move in the left or right direction through the joystick
- Set transform.localscale according to each direction to view the direction of movement.
- jump button also adds a function to jump.
- Gets the direction value of the joystick in the update function and processes it to move.

2. Create Joystick

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

public class Joystick2 : MonoBehaviour, IPointerUpHandler,
{
    private Transform btn;
    public float length = 58;
    private RectTransform background;
    private Vector2 direction = Vector2.zero;

    public Vector2 Dir
    {
        get
        {
            return direction;
        }
    }
```

- Add using UnityEngine.EventSystems; to use the UnityEventSystems.
- Inherits an interface to be used as an event system.
- Length is the maximum distance the joystick will move, which the editor declares public for adjustment.
- The direction value of the joystick is received as a read-only property.

6

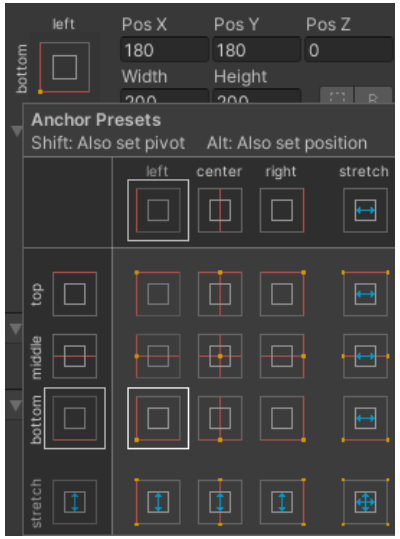
2. Create Joystick

- Override the function in the interface
- When the button is released, set the button to return to its original.
- Dragging a button saves the dragged direction as a vector of length 1.

```
public void OnPointerUp(PointerEventData eventData)
{
    btn.localPosition = Vector3.zero;
    direction = Vector2.zero;
}

public void OnDrag(PointerEventData eventData)
{
    Vector2 localPoint = Vector2.zero;

    if (RectTransformUtility.ScreenPointToLocalPointInRectangle(background,
                                                                 eventData.position,
                                                                 eventData.pressEventCamera,
                                                                 out localPoint))
    {
        btn.localPosition = localPoint;
        if (localPoint.magnitude >= length)
            btn.localPosition = localPoint.normalized * length;
    }

    direction = localPoint.normalized;
}

public void Init()
{
    btn = transform.Find("Btn");
    background = GetComponent<RectTransform>();
}
```


7

1. Create an image of the UI in the hierarchy window, then create another image at the bottom.
2. One is the joystick's background, and the other is a moving button.

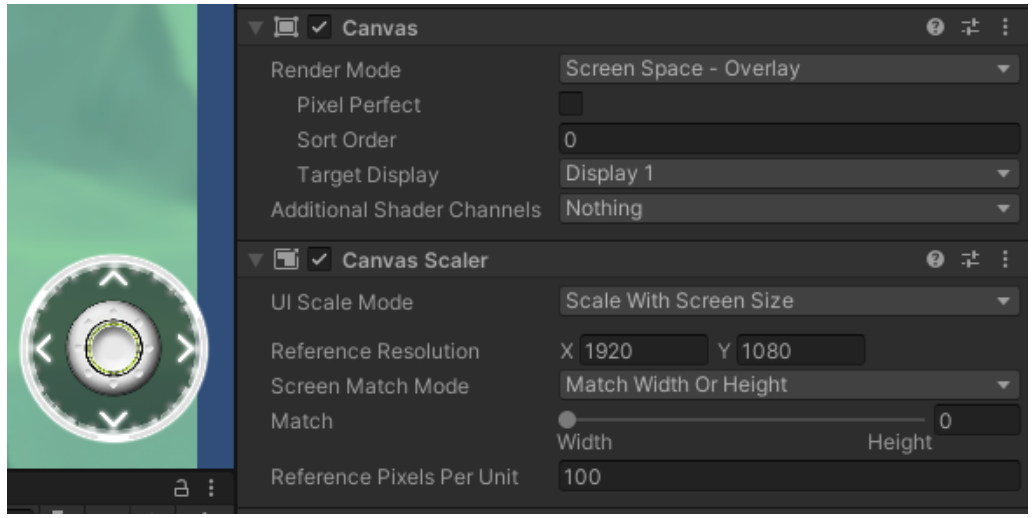


1. Set the anchor preset of the background image to the bottom left and adjust it through pos X, pos Y.

2. Set the joystick button image to center.

3. Joystick Setting

1) UI Configuration



1. Create a joystick UI to use to move players as shown in the figure.

2. Change Canvas UI Scale Mode to Scale With Screen Size to adjust 1920*1080 resolution and create a UI.

3. Place the joystick on the bottom left and resize it.

9

3. Joystick Setting

2) Character Movement Implementation – Source Code

```
public void OnPointerDown(PointerEventData eventData)
{
    print("DownEvent");
}
public void OnPointerUp(PointerEventData eventData)
{
    btn.position = startPos;
    // 조이스틱 버튼이 업 되었다면 방향을 0, 0으로 설정합니다.
    direction = Vector2.zero;
}
public void OnDrag(PointerEventData eventData)
{
    btn.position = eventData.position;

    Vector2 dir =  eventData.position - startPos;

    // 조이스틱이 향하는 방향을 얻습니다.( 길이가 1인 벡터 )
    direction = dir.normalized;

    if( dir.magnitude > length )
    {
        dir.Normalize();
        btn.position = startPos + dir * length;
    }
}

// Start is called before the first frame update
public void Init()
{
    btn = transform.Find("Btn");
    startPos = btn.position;

    Transform pivot = transform.Find("Pivot");
    Vector2 pivotPos = pivot.position;
    length = (pivotPos - startPos).magnitude;
}
```

- Sets the event function that occurs according to the mouse
- Gets the mouse position value and sets the position value of the joystick
- When the joystick is facing up, set the direction to 0,0 so that it does not rise up
- Obtain the difference from the joystick's original position value to obtain the direction the joystick is facing
- Controls the image of the joystick so that it does not deviate more than a certain distance

10

3. Joystick Setting

3) Result of Movement

11

4. Create Jump Button

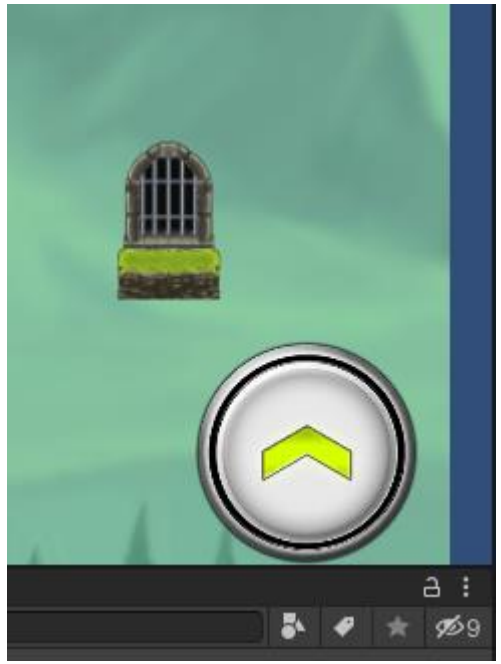
- Create a button for the UI in the highlight window

```
PlayerController player;

public void Init()
{
    Transform t = transform.Find("Jump");
    if(t!=null)
    {
        Button btn = t.GetComponent<Button>();
        if (btn != null)
            btn.onClick.AddListener(OnClickJump);
    }
    player = GameObject.FindObjectOfType<PlayerController>();
}

void OnClickJump()
{
    if (player != null)
    {
        player.Jump();
    }
}
```

- Press the jump button to create a fuction to link, and click on the button. Use the AddListener function to connect
- Press the jump button to invoke the player's jump function script.

5. Jump Setting

1-1)Jump UI Configuration, Source Code



```
public void Jump()
{
    if (rigidbody2D.velocity.y != 0)
        return;

    rigidbody2D.AddForce(Vector2.up * 400);

}
```

Place the button UI in the lower right corner and create a jump fuction

5. Jump Setting

1-2) Source code

```
public void Init()
{
    Transform t = transform.Find("Jump");
    if (t != null)
    {
        Button btn = t.GetComponent<Button>();
        if (btn != null)
        {
            // 점프 버튼을 클릭하였을 때 실행할 함수를 연결합니다.
            // 함수는 리턴 타입이 void이고 매개변수가 없는 형태만 연결될 수 있습니다.
            btn.onClick.AddListener(OnClickJump);
        }
    }

    playerController = GameObject.FindObjectOfType<PlayerController>();

    // 지금까지 실패한 카운트를 ui에 출력합니다.
    textMeshPro =  GetComponentInChildren<TMP_Text>(true);
    textMeshPro.SetText("x " + GameData.failedCount);
}

//
void OnClickJump()
{
    // 게임이 클리어된 상태에서는 클릭되어도 점프되지 않아야 합니다.
    if (GameData.inputState == false)
        return;

    if (playerController != null)
    {
        playerController.Jump();
    }
}
```

- Connects to a button via AddListener within a script
- Run the jump function when the OnClickJump jump button is pressed and does not work when the game is not cleared.

14

5. Jump Setting

2) Result of Moving

15

Thank you