PORT FOLIO

김일환(GIM IL HWAN)
Email : dodjungvv@naver.com
Phone Number : 010 - 4142 - 1442

1. Treasurer

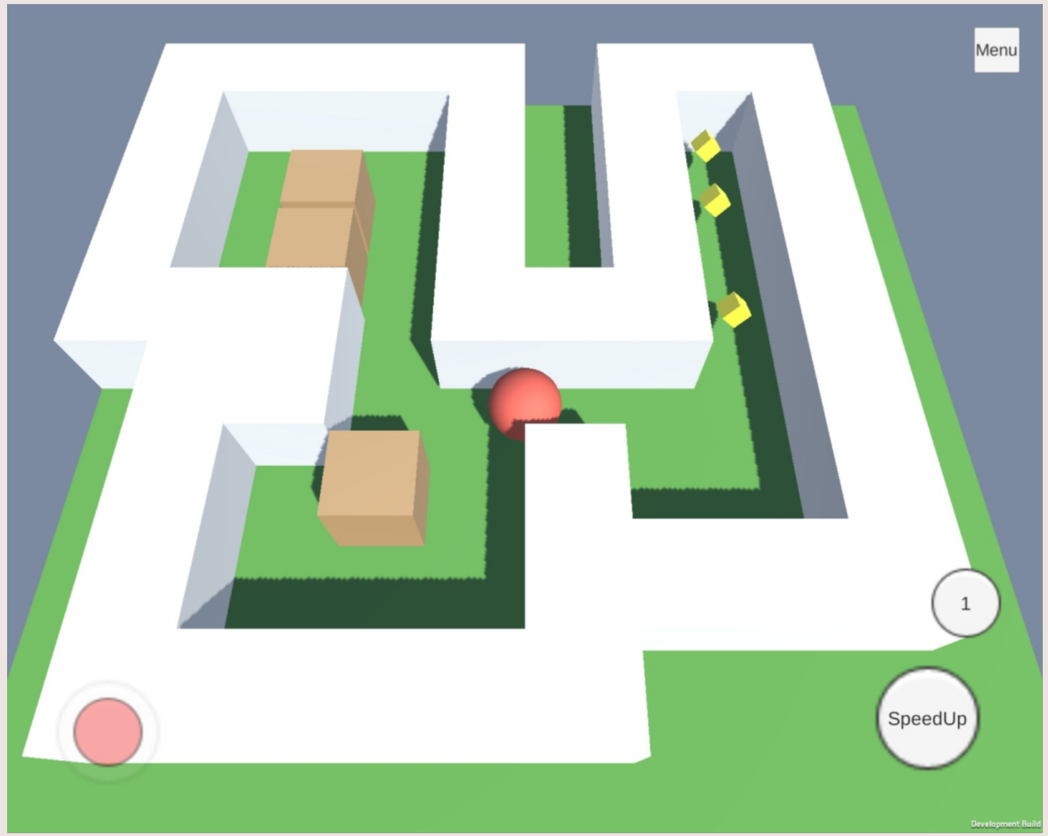
GIM ILHWAN

누구에게도 지지 않을 만큼 게임 개발에 진심인 사람입니다!

gim ilhwan

1. Treasurer Game

프로젝트에 대한 간단 설명

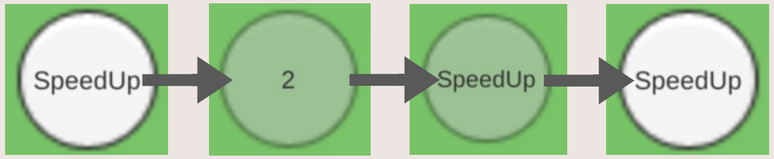


1. 조이스틱을 사용하여 빨간 구슬을 이동 할 수 있다.
2. 빨간 구슬을 이동하여 맵에 있는 박스를 지정된 지점에 전부 이동 시키면 게임이 끝난다.

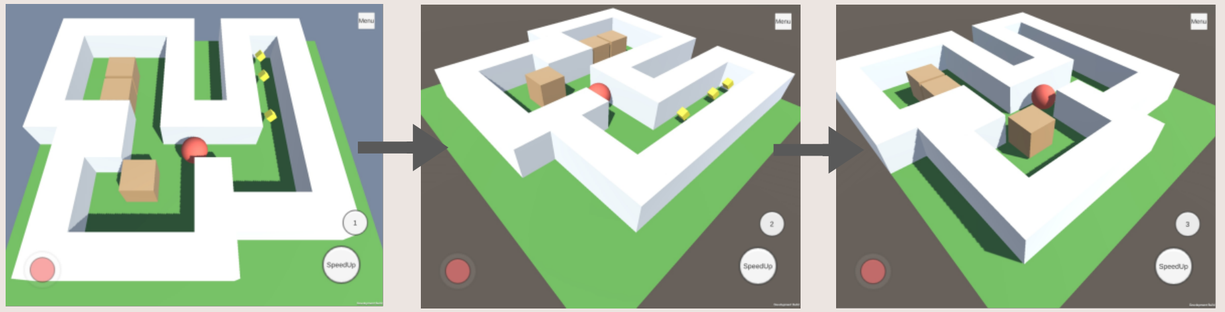
1. Game Control

1.
- 해당 조이스틱을 사용하여 붉은 구슬(Player) 를 이동 시켜 게임 화면의 박스를 이동 시킬 수 있다.

2.
- SpeedUp 버튼을 사용해 붉은 구슬(Player) 의 속도를 3초간 증가 시킬 수 있다.
- 버튼 사용 직후 남은 증가 시간이 나오고 일정 시간이 지나면 기존 Text 로 변경 된다.
- 버튼 사용 후 5초가 지나면 다시 해당 버튼이 활성화 된다.



3.
- 해당 버튼을 사용하여 카메라의 시점(카메라의 종류)을 변경 할 수 있다.
- 카메라의 시점에 따라 Text 의 숫자가 변경 된다.



2. Game Menu

1.
- 해당 기능을 사용하여 박스를 더이상 이동 할 수 없거나 문제가 발생 했을때 게임을 재시작 할 수 있다.

2.
- 게임 배경음악을 On/Off 할 수 있다.

3.
- 게임을 종료 할 수 있다 .
- Unity 환경과 실제 모바일 환경을 구분하여 종료 한다.

4.
- Menu 화면(Panel)을 활성 비활성 할 수 있다.

1. PlayerController

```
53      // Target의 이동 및 회전
54      void MoveAndRotate()////////////////////////////////////////////////////////////////////////////////////////
55      {
56          if (!canMove) return;
57
58          if(SpeedUp.speedUp_yn.Equals(true))
59          {
60              moveSpeed = 7 + speedUp;
61              Debug.Log("속도 올라감!" );
62          }
63          else
64          {
65              moveSpeed = 7;
66          }
67
68          // 이동
69          Vector2 normalVec = stickVector.normalized;
70          targetPlayer.position += new Vector3(normalVec.x, 0, normalVec.y) * stickDistRatio * moveSpeed * Time.deltaTime;
71
72          // 회전
73          Vector3 newRot = Vector3.up * Mathf.Atan2(normalVec.x, normalVec.y) * Mathf.Rad2Deg;
74          targetPlayer.rotation = Quaternion.Lerp(targetPlayer.rotation, Quaternion.Euler(newRot), rotSpeed * Time.deltaTime);
75      }
```

- 58 라인 Player의 SpeedUp 여부를 확인

- 69, 73 라인의 Vector2, Vector3 의 Vector 값을 이용하여 70, 74 라인의 Position, Rotation에 적용하여 각각 (x, y) 축 이동및 (x, y, z) 축 기준이로 Player 를 이동및 회전 시킨다.

2. PlayerController

```
20    IEnumerator currCoroutine;                    // 현재 진행중인 코루틴 (OnJoyStickReset 정지하기 위해 사용)
```

```
97    // 조이스틱이 초기 위치로 돌아옴
98    IEnumerator OnJoyStickReset()//////////////////////////////////////////////////////////////////////////////
99    {
100       isReturn = true;
101
102       float time = Time.time;
103
104       while (Time.time < time + smoothTime)
105       {
106           joyStickButton.position = Vector3.SmoothDamp(
107               joyStickButton.position, joyStickBG.position, ref smoothVelocity, inverseSmoothSpeed);
108           yield return null;
109       }
110
111       isReturn = false;
112       joyStickButton.position = joyStickBG.position;
113   }
```

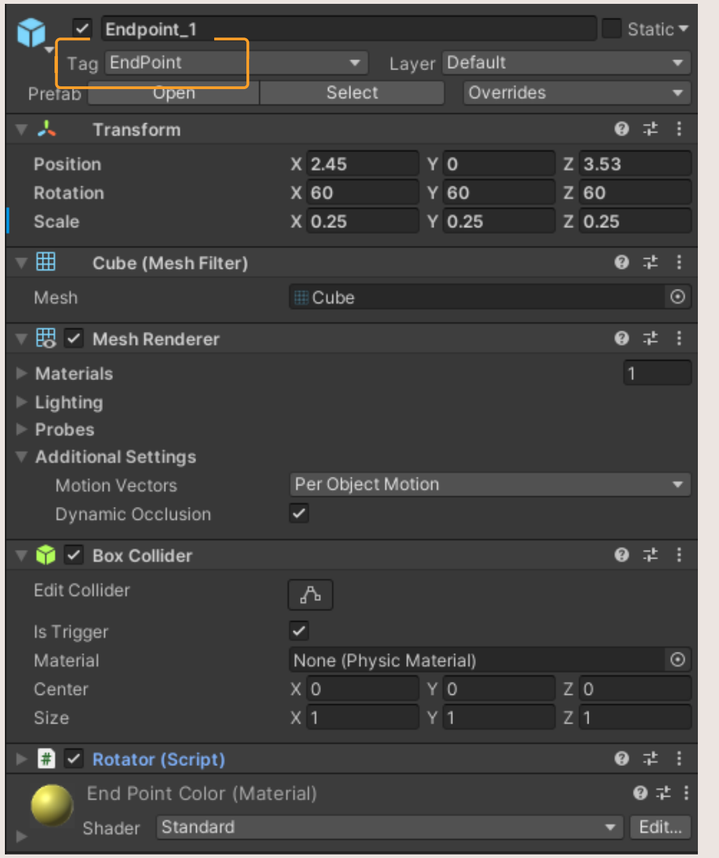
- 20, 108 라인을 사용하여 Update() 주기 (사용 기기의 사양) 과 상관 없이 일정한 시간을 측정하여 Event를 발생 시킨다. (사양에 따라 프레임이 변경 될 수 있기 때문에)

- 104 라인의 smoothTime 변수 값(초) 에 따라 조이스틱이 초기 위치로 돌아 오는 시간이 변경 된다 .

- 98 ~ 113 OnJoyStickRest() 함수는 조이스틱에서 손을 때는 순간 발동하여 조이스틱을 초기 위치로 초기화 한다.

3. ItemBox

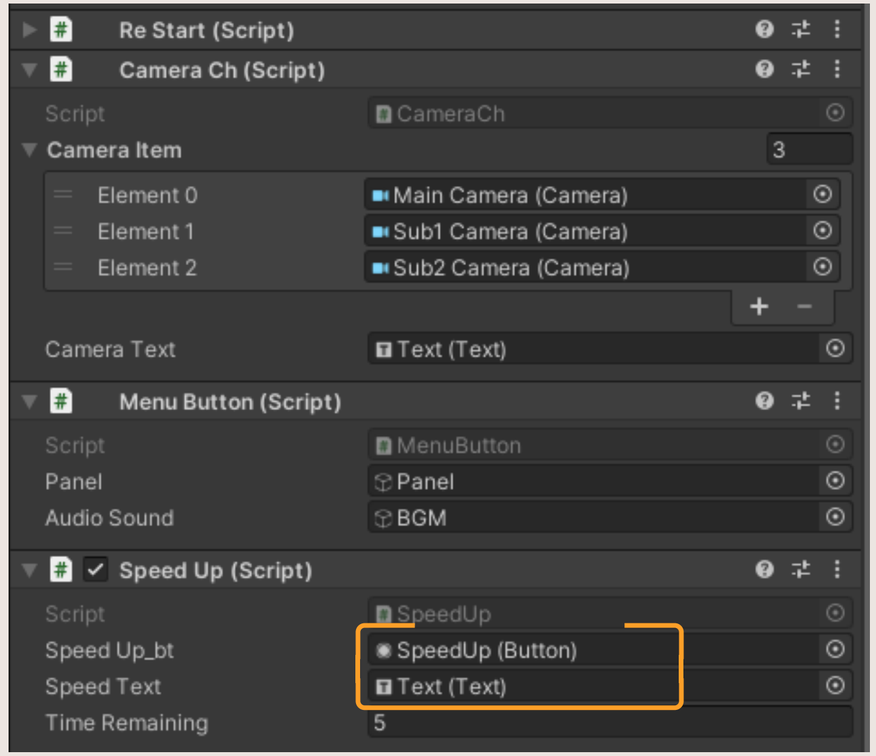
```
29      //Enter은 충돌을 한순간!
30      void OnTriggerEnter(Collider other)//충돌하게 되면 발동한다.
31      {
32          if (other.tag.Equals("EndPoint"))
33          {
34              isOveraped = true;
35              myRenderer.material.color = touchColor;
36          }
37          //충돌했을때 시행 할 이벤트 작성!
38          //통과 가능한 충돌 인경우
39      }
40
41      //Exit 충돌이 해제 끝났을때? 실행됨
42      void OnTriggerExit(Collider other)
43      {
44          if (other.tag.Equals("EndPoint"))
45          {
46              isOveraped = false;
47              myRenderer.material.color = originalColor;
48          }
49      }
50
51      //stay 충돌 중일때 발동함
52      void OnTriggerStay(Collider other)
53      {
54          if (other.tag.Equals("EndPoint"))
55          {
56              isOveraped = true;
57              myRenderer.material.color = touchColor;
58          }
59      }
```



- 30, 42, 52 라인 함수들을 이용하여 특정 Object 와의 충돌 여부를 확인 할 수 있다.
- 32, 44, 54 라인의 other.tag 를 판단하여 상황에 맞는 Event를 처리한다.
- 이때 ItemBox 는 Rigidbody 를 사용하여 기본적인 물리 법칙에 영향을 받는다.

4. SpeedUp

```
19      void Update()
20      {
21          if (SpeedUp_bt.interactable.Equals(false))
22          {
23              timeRemaining -= Time.deltaTime;
24              if (timeRemaining > 2) {
25                  speedText.GetComponent<Text>().text = ( Mathf.FloorToInt(timeRemaining) - 1).ToString();
26              }
27              else
28              {
29                  speedText.GetComponent<Text>().text = "SpeedUp";
30              }
31
32          }
33          if (timeRemaining <= 2 && timeRemaining > 0)
34          {
35              SpeedUp.speedUp_yn = false;
36          }
37          else if (timeRemaining <= 0)
38          {
39              SpeedUp_bt.interactable = true;
40              timeRemaining = 5;
41          }
42      }
```

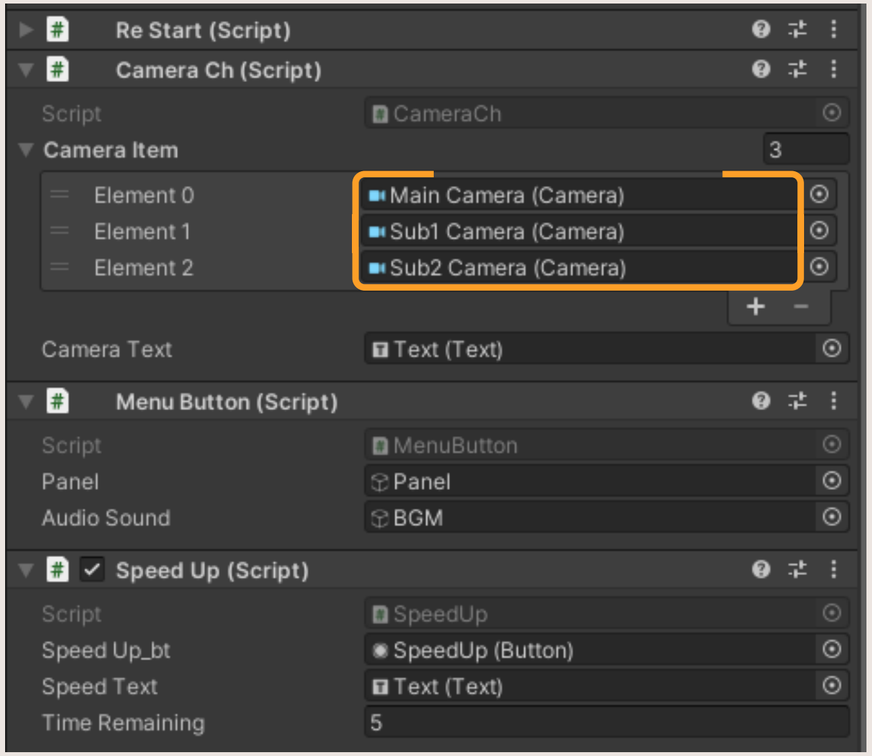


- SpeedUp 버튼을 클릭하면 Button Component 비활성화및 Text Component 를 timeRemaining 변수 값(초) 에따라 Text 를 변경한다.

- 코루틴을 이용하여 Update() 주기 수정 필요...

5. CameraCh

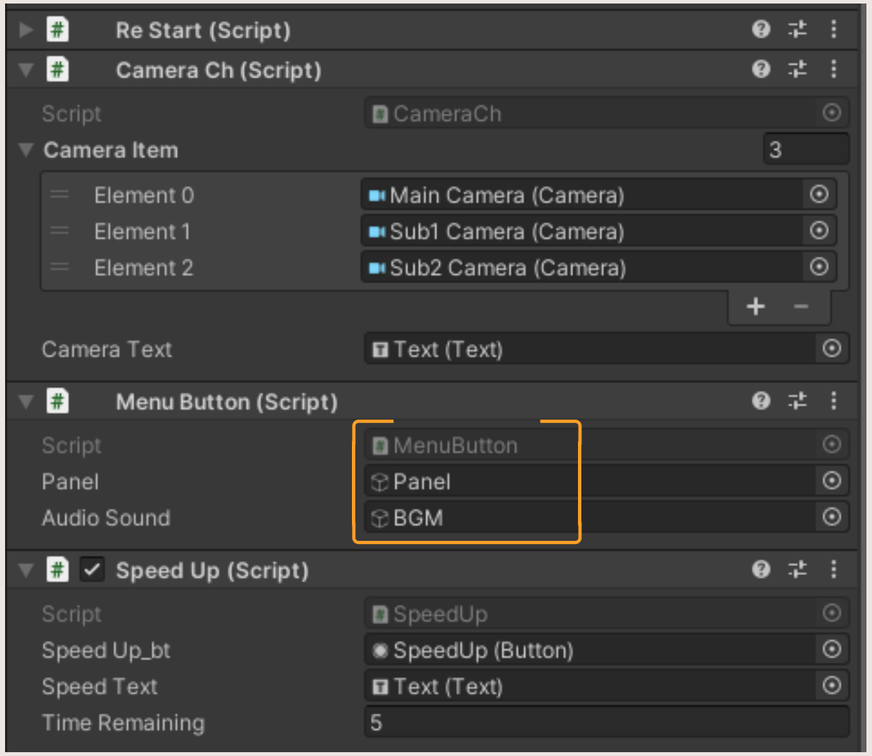
```
8   public Camera[] cameraItem;//카메라 활성 비활성화
9
10  int cameraCount = 3;//카메라 갯수
11  int nowCamera = 0;// 활성화된 카메라 번호
12
13  public Text cameraText;
14
15  public void CameraChange()
16  {
17      ++nowCamera;
18
19      if (nowCamera >= cameraCount)
20      {
21          nowCamera = 0;
22          cameraText.GetComponent<Text>().text = "1";
23      }
24      for (int i = 0; i < cameraItem.Length; ++i)
25      {
26          if (i == nowCamera)
27          {
28              cameraItem[i].enabled = true;
29              cameraText.GetComponent<Text>().text = (i + 1).ToString();
30          }
31          else
32          {
33              cameraItem[i].enabled = false;
34          }
35      }
36  }
```



- *8 라인에 Camera[] 를 사용하여 Camera Object를 등록 후 특정 배열 위치에 있는 Camera를 Control 하기 위해 선언*
- *CameraChange() 함수를 호출 할때마다 19 라인에서 등록된 Camera 개수를 초과 하지 않도록 확인 후 Text 를 변경한다.*
- *24 라인에 for()문 을 사용하여 활성및 비활성화 Camera를 변경 할 수 있다.*

6. CameraCh

```
9     public GameObject panel;
10    public GameObject audioSound;
11
12    public void ReStart()
13    {
14        SceneManager.LoadScene("treasurer");
15    }
16    public void SoundCtr()
17    {
18        if (audioSound.active.Equals(true))
19            audioSound.SetActive(false);
20        else
21            audioSound.SetActive(true);
22    }
23    public void GameExit()
24    {
25        #if UNITY_EDITOR
26            UnityEditor.EditorApplication.isPlaying = false;
27        #else
28            Application.Quit(); // 어플리케이션 종료
29        #endif
30    }
31
32    public void PanelCtr()
33    {
34        if (panel.active.Equals(true))
35            panel.SetActive(false);
36        else
37            panel.SetActive(true);
38    }
```



- *9, 10 라인 GameObject 를 사용하여 Menu Panel 및 Game BGM 을 관리 한다.*
- *ReStart() 함수의 14라인의 LoadScene 을 사용하여 "Treasurer" Scene을 불러와 게임을 재시작 한다.*
- *SoundCtr(), PanelCtr() 함수의 SetActive 를 사용하여 Audio를 종료하거나 실행하고 Panel을 활성화 하거나 비활성화 한다.*
- *GameExit() 함수를 사용하여 게임을 종료 할 수 있다.*

- 앞으로 나아갈 길...

1. 기본 적인 UI 정리 및 다양한 모바일 환경에서 Play 가능하도록 UI 구상
(현재는 Button 사이즈, 위치 만 최적화 되어 있습니다.)

2. 클리어에 걸린 시간 확인 기능 추가.

3. MySql 연동 후 Table을 생성하여 클리어 시간및 기타정보 저장, 확인 기능 추가.

마지막으로...

이번 게임은 안드로이드 환경에서 플레이 가능하도록 Apk파일이 준비 되어 있습니다.

끝 까지 읽어 주셔서 감사합니다!