콘텐츠제작

무궁화 꽃이 피었습니다

인터넷 보안공학과
2021671029 박예서

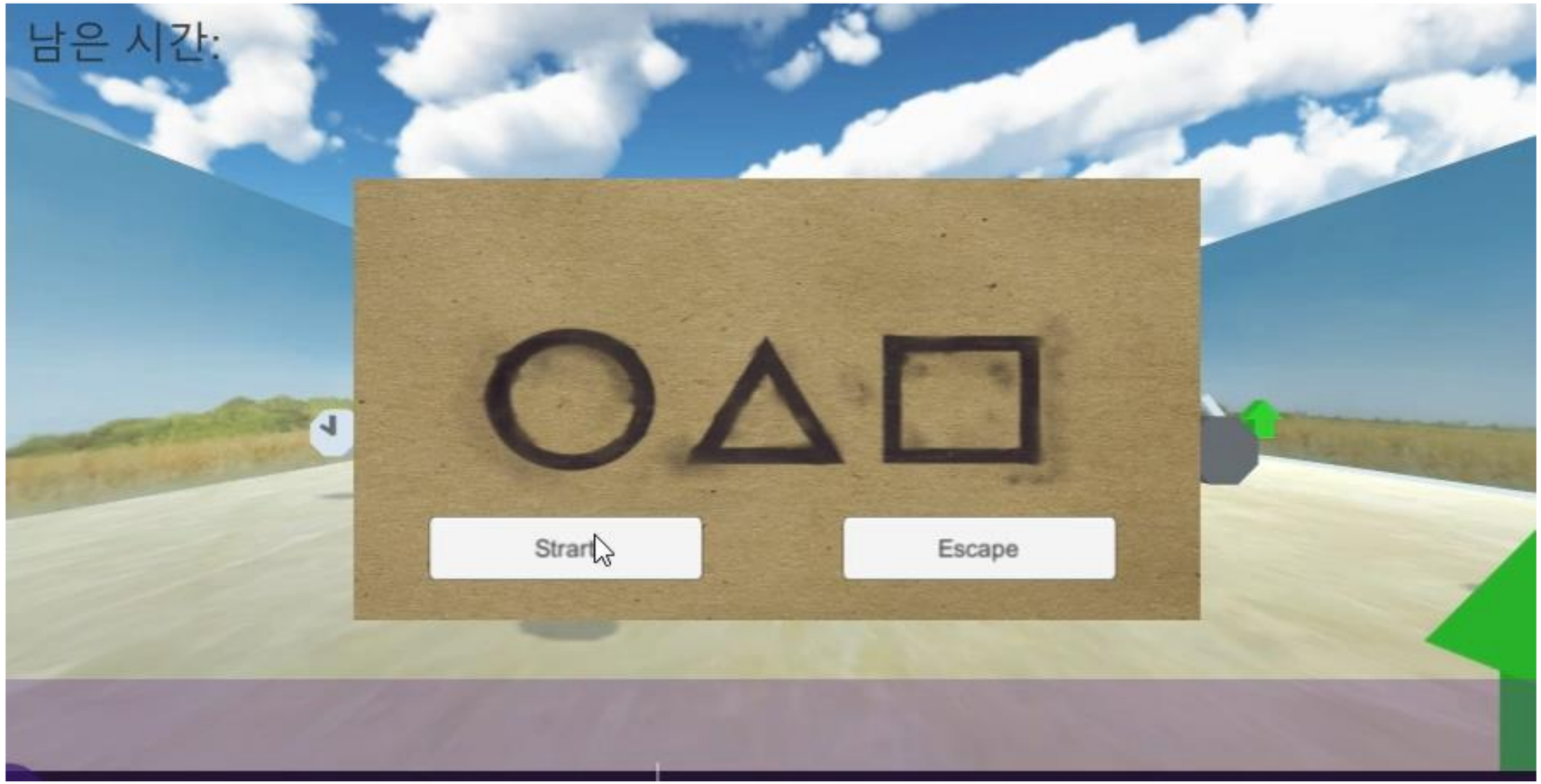
시작 >

목차

플레이 영상

01

플레이영상



https://youtu.be/p0BS99bd9xs

”

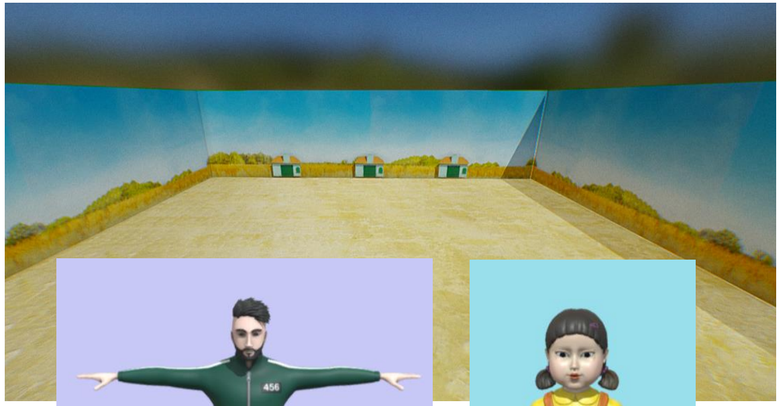
프로젝트 개요

02

게임 컨셉

- 게임명: 무궁화 꽃이 피었습니다.
- 장르: 캐주얼 게임*
- 플랫폼: PC
- ‘오징어 게임 ' 의 라운드 1에서 아이디어를 얻음
- 몰입감을 높이기 위해 극중 환경과 유사한 분위기를 연출

*간단한 조작으로 짧은 시간동안 즐길 수 있는 게임



제작 과정

- 제작기간 : 약 2주
- 폭포수 모델* 방법론을 적용
- 개발 진행 일정 및 프로젝트 수정 사항을 형상 관리도구인 깃허브와 노션를 이용

구분	추진내용	추진일정					
		8주	9주	10주	12주	13주	14주
계획	게임 주제 선정 및 계획서 작성						
분석	게임 주제에 대한 구체적인 조사						
설계	에셋 수집						
	전체적인 게임의 설계						
개발	전체적인 게임 개발						
	에셋 적용						
	오류 수정						
테스트	발표 준비 및 테스트						
종료	게임 제작 발표						

제작 과정

master ▾ 1 branch 0 tags Go to file

rexRUBY Delete ErrorDumy.unitypackage cae7b10 6

Project	Rename TimedDistroy to TimedDistroy.cs
SquidGame	Create f1
review	점수 그리고 씬 이동
Prototype.unitypackage	오징어게임 라운드 1 프로토타입
README.md	Update README.md

README.md

Unity 복습 및 미니 프로젝트

게임 개발 TODO

오징어 게임을 만들어 보자

Board View 표 + 보기 추가

할 일 3

구슬치기
알카노이드에서 아이디어를 얻어보자
라운드 4 구현

게임 오버 됐을때 시간이 느리게 흘러가도록
평소가 1임. 0.2로 설정하면 슬로우로 흘러감
Time.timeScale 사용

휴머노이드 타입으로 적용 안되는 문제
- [ ] 레거시 방식으로 에니메이션 컨트롤러
- [ ] 레거시 방식으로 c# 스크립트 짜기

에니메이션

+ 새로 만들기

진행 중 3

무궁화꽃이 피었습니다
- [x] 랜덤으로 아이템 생성
- [ ] 플레이어 에니메이션 추가
- [x] 아이템 먹으면 나타나는 효과
- [x] 남은 시간 흘러가는 시간 바제작
- [x] 타이머 제작

라운드1 구현

- [ ] 코드 설명(질문 공세 받아도 다 대답 해야 함.)
- [ ] 게임 개요
- [ ] 플레이 영상
- [ ] 배우지 않은 새로운 기능
- [ ] 느낀점

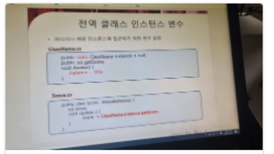
발표준비
2

- [x] 시작버튼
- [x] 종료버튼
- [ ] 튜토리얼
- [x] 명함 날라오는 UI

전체적인 UI

+ 새로 만들기

완료 2



타이머 구현

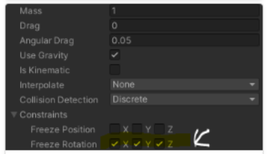
- [x] 카메라 추가
- [x] 1인칭 카메라 스크립트

SubCam

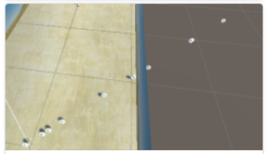
+ 새로 만들기

오류 수정 6

전역 클래스 변수
1



플레이어 넘어짐 수정



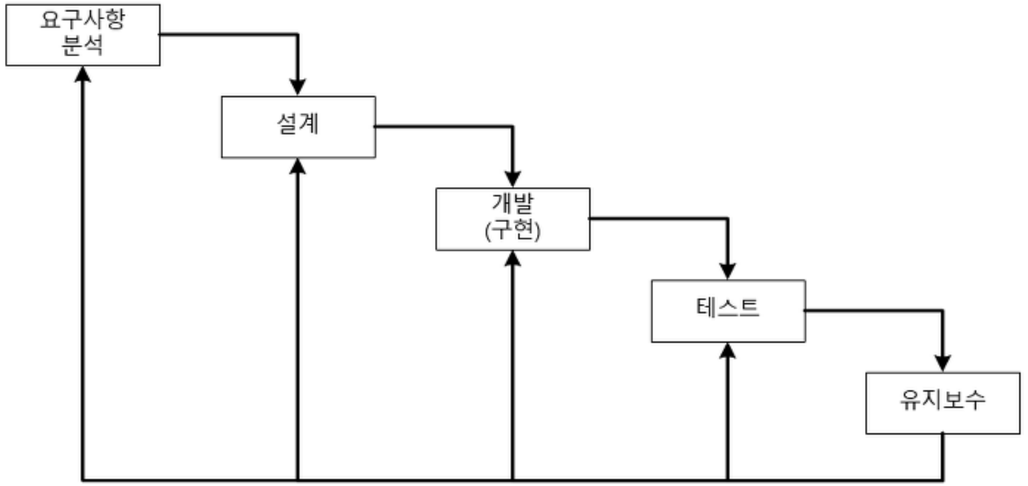
아이템 랜덤생성
2

마우스 스크롤이 조금 이상했음
어제고쳤냐면 내가 속도를 지정하지 않아서 생긴 오류였음
이렇게 수정해 주었음

플레이어 회전

* 폭포수 모델

- 폭포가 떨어지듯 각 단계가 끝나면 다음 단계로 진행 되는 모델
- 각 단계가 명확히 구분되어 프로젝트의 진행 단계가 명확함
- 앞 단계에서 문제가 발생했을 때 되돌아가기 어려움.



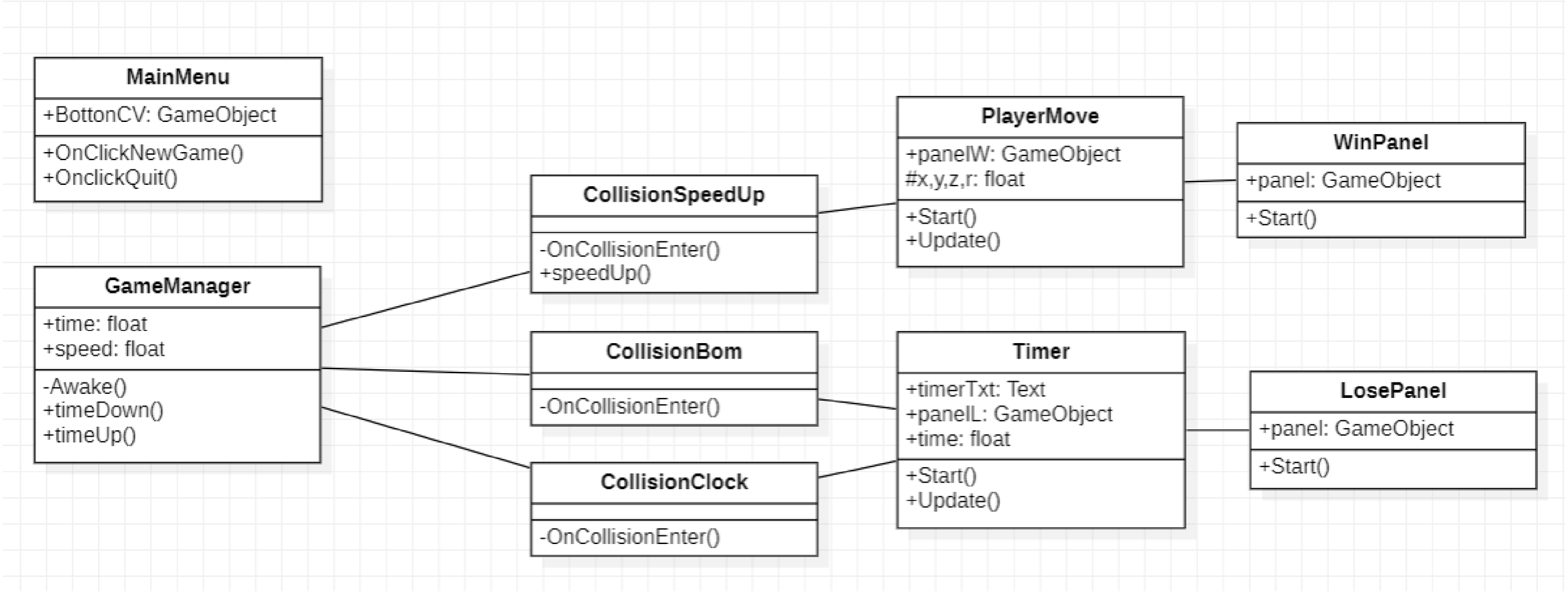
”

03

기능소개 & 스크립트

04

클래스들의 연관 관계


MainMenu
+BottonCV: GameObject
+OnClickNewGame()
+OnclickQuit()
GameManager
+time: float
+speed: float
-Awake()
+timeDown()
+timeUp()
CollisionSpeedUp
-OnCollisionEnter()
+speedUp()
CollisionBom
-OnCollisionEnter()
CollisionClock
-OnCollisionEnter()
PlayerMove
+panelW: GameObject
#x,y,z,r: float
+Start()
+Update()
Timer
+timerTxt: Text
+panelL: GameObject
+time: float
+Start()
+Update()
WinPanel
+panel: GameObject
+Start()
LosePanel
+panel: GameObject
+Start()

시작 화면

- 시작 버튼 및 종료 버튼


```
public class MainMenu : MonoBehaviour
{
    public GameObject BottonCV;

    public void OnClickNewGame()
    {
        BottonCV.SetActive(false);
    }

    public void OnclickQuit()
    {
#if UNITY_EDITOR
        UnityEditor.EditorApplication.isPlaying=false;
#else
        Appliction.Quit();
#endif
    }
}
```

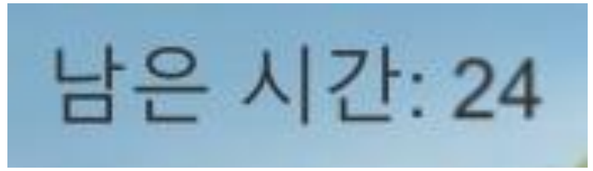
게임 메니저 - 1

```
public static GameManager instance; //클래스 내 모든 변수 및 함수 대상

public static GameManager Instance
{
    get { return instance; }
}

public float time=30f; //게임 내에서 유지하려고 하는 시간
public float speed = 0.3f; //게임 내에서 유지하려고 하는 속도

private void Awake()  //Awake는 스크립트에서 가장 먼저 실행됨
{
    if (instance) //instance가 내가 아니라면 이미 instance가 하나 존재하고 있다는 의미
    {
        Destroy(gameObject); //둘 이상 존재하면 안되는 객체이니 방금 AWake된 자신을 삭제
    }
    else if (!instance) //instance가 시스템상에 존재하고 있지 않을때
    {
        instance = this;
        DontDestroyOnLoad(gameObject);
    }
}
```

게임 메니저 - 2

```
public void timeUp()
{
    time += 3f;
    Timer.time += 3;
}

public void timeDown()
{
    time -= 3f;
    Timer.time -= 3;
}
```

타이머 – 텍스트 (UI)




```
public class Timer : MonoBehaviour
{
    public Text timerTxt;
    public GameObject panelL;
    public static float time;  //전역클래스 인스턴스 변수

    void Start()
    {
        time= 30f;
    }
    void Update()
    {
        if (Mathf.Floor(time) <= 0) //Count 0일때 동작할 함수, Mathf.Floor 작거나 같은 정수값 반환
        {
            panelL.SetActive(true);
            Time.timeScale = 0; //게임 종료
        }
        else
        {
            time -= Time.deltaTime; //초의 흐름
            timerTxt.text = Mathf.Floor(time).ToString(); //컴포넌트 이기 때문에 Txt.text를 해 줌
        }
    }
}
```

아이템 – 폭탄

```
public class CollisionBom : MonoBehaviour
{
    private void OnCollisionEnter(Collision coll)
    {
        if (coll.gameObject.tag == "Player")
        {
            GameManager.Instance.timeDown();
            Destroy(gameObject);
        }
    }
}
```

```
public void timeDown()
    {
        time -= 3f;
        Timer.time -= 3;
    }
```

← Game Manager

아이템 – 시계

```
public class CollisionClock : MonoBehaviour
{
    private void OnCollisionEnter(Collision coll)
    {
        if (coll.gameObject.tag == "Player")
        {
            GameManager.Instance.timeUp();
            Destroy(gameObject);
        }
    }
}
```

```
public void timeDown()
    {
        time += 3f;
        Timer.time += 3;
    }
```

← Game Manager

아이템 – 스피드

```
public class CollisionSpeedUp : MonoBehaviour
{
    public void OnCollisionEnter(Collision coll)
    {
        if (coll.gameObject.tag == "Player")
        {
            Destroy(gameObject);
            GameManager.instance.speed=1.5f;
            StartCoroutine(speedUp());
            GameManager.instance.speed = 0.3f;
        }
    }

    IEnumerator speedUp()
    {
        yield return new WaitForSeconds(3f);
    }
}
```

코루틴

- 코루틴은 실행을 중지하여 Unity에 제어권을 돌려주고, 다시 실행 할 때는 다음 프레임에서 중지한 곳부터 실행을 계속할 수 있는 기능
- Update 함수가 반환된 후 수행
- 반환형은 IEnumerator, 리턴은 yield return
- 코루틴을 실행하려면 StartCoroutine 함수를 사용
- yield return 행은 실행을 중지하고 다음 프레임에서 실행을 재개할 수 있는 지점을 의미
  - __yield__ 코루틴은 다음 프레임에서 모든 Update 함수가 호출된 후에 속행합니다.
  - __yield WaitForSeconds__ 프레임에 대한 모든 Update 함수가 호출된 후 지정된 시간 지연 후에 속행합니다.
  - __yield WaitForFixedUpdate__ 모든 스크립트에서 모든 FixedUpdate 호출 후 속행합니다.
  - __yield WWW__ WWW 다운로드 완료 후 속행합니다.
  - __yield StartCoroutine__ 코루틴을 연결하고 MyFunc 코루틴이 먼저 완료될 때까지 기다립니다.

아이템 무작위 생성

```
public class ItemDrop : MonoBehaviour
{
    public GameObject Item;
    void Start()
    {
        InvokeRepeating("Spawnitem", 0, 3);

    }

    void Spawnitem()
    {
        float random1 = Random.Range(-2f, 25f);
        float random2 = Random.Range(-2f, 25f);

        if (true)
        {
            //Debug.Log("생성");
            GameObject item = (GameObject)Instantiate(Item, new Vector3(random1, 1f, random2),
Quaternion.identity);
        }
    }
}
```

아이템 무작위 생성

```
public class ItemDrop : MonoBehaviour
{
    public GameObject Item;
    int n;

    IEnumerator Spawnitem(float loopTime)
    {
        while (true)
        {
            float random1 = Random.Range(-10f, 10f);
            float random2 = Random.Range(-19f, 35f);

            GameObject item = (GameObject)Instantiate(Item, new Vector3(random1, 1f, random2),
Quaternion.identity);

            yield return new WaitForSeconds(loopTime);
        }
    }
}
```

패널 - 이긴 경우

```
public class WinPanel : MonoBehaviour
{
    public GameObject panel;

    void Start()
    {
        panel.SetActive(false);
    }
}
```

```
public class PlayerMove : MonoBehaviour
{
    public GameObject panelW;

    float x, y, z;
    float r;
    void Start()
    {
        x = 0;
        y = 0;
        z = 0;
    }

    void Update()
    {
        z = Input.GetAxis("Vertical") * GameManager.instance.speed * Time.deltaTime;
        transform.Translate(x, y, z);

        x = Input.GetAxis("Horizontal") * GameManager.instance.speed * Time.deltaTime;
        transform.Translate(x, y, z);

        if (transform.position.z > 29.33)
        {
            panelW.SetActive(true);
            Time.timeScale = 0;
        }
    }
}
```

패널 - 진 경우

```
public class LosePanel : MonoBehaviour
{
    public GameObject panel;

    void Start()
    {
        panel.SetActive(false);
    }
}
```

```
public class Timer : MonoBehaviour
{
    public Text timerTxt;
    public GameObject panelL;
    public static float time;
    void Start()
    {
        time= 30f;
    }

    void Update()
    {
        if (Mathf.Floor(time) <= 0)
        {
            panelL.SetActive(true);
            Time.timeScale = 0;
       else
        {
            time -= Time.deltaTime;
            timerTxt.text = Mathf.Floor(time).ToString();
        }
    }
}
```

Q&A

05

감사합니다.

”

한 학기동안 수고 많으셨습니다☺

끝!

콘텐츠 제작 무궁화 꽃이 피었습니다 2022.06.03