인터랙티브미디어 프로젝트 최종 보고서

1716572 박혜림

<목차>

- **제목 <횡스크롤 격투게임>**
- **주제**
- **기능**
- **구현방법 (구현 방법 + 주요 스크립트)**
  - **(1) 캐릭터 선택 씬**
    - 배경, 캐릭터 구현 -3p
    - 캔버스, 버튼 관리 -5p
    - 카메라 관리 - 6p
    - 게임시작, 선택 정보 저장(ChoiceManager) - 8p
  - **(2) 전투 씬 -9p**
    - 오디오 관리(배경음, 효과음) -10p
    - StartManager (게임 시작 관리, 초기화) - 11p
    - 캔버스(hp, speed) -13p
    - 조작 (직접조작, 자동조작) -15p
    - 충돌/타격 감지 -20p
    - 아이템 생성 및 충돌 -22p
- **실행화면 -25p**

*** 제목 <횡스크롤 격투게임>**

*** 주제**

파이터, 매지션 중 원하는 직업을 선택하고 컴퓨터와 싸워보는 횡스크롤 격투게임입니다.

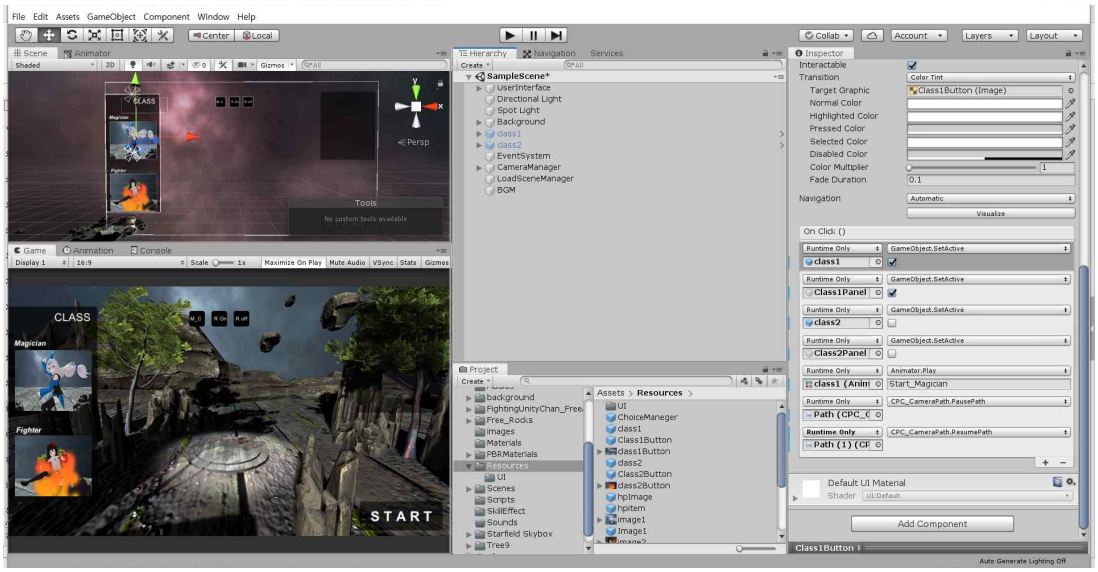
* 기능

캐릭터 선택 씬
• 플레이어가 원하는 직업 버튼을 클릭하면 가운데 무대에 직업이 생성됩니다.(SetActive) • 플레이어를 설명하는 패널 또한 SetActive됩니다. • 직업은 애니메이션을 시작합니다. • 직업을 비추는 카메라가 움직입니다. • Start버튼을 누르면 게임이 시작됩니다.

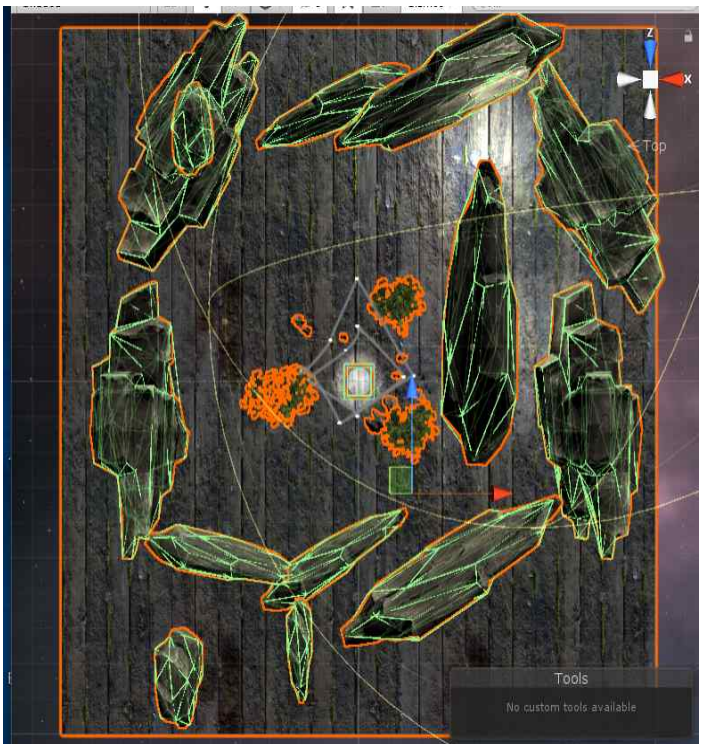
⇩ **- 플레이어의 선택 정보 저장**
- 전투 씬 로드

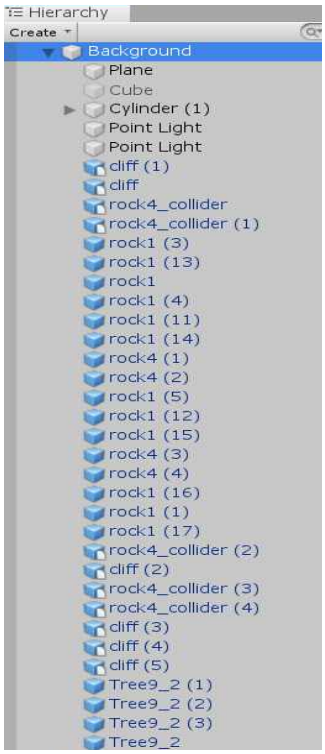
전투 씬
• 플레이어는 화면의 좌측에, 컴퓨터는 좌측에 배치됩니다. • 화면 상단에는 각 캐릭터의 체력과 스피드가 표시됩니다. • 플레이어는 방향키와 Ctrl, Alt로 조작하고, 컴퓨터는 자동 조작합니다. • 체력아이템, 스피드아이템, 슬로우아이템 3가지가 맵의 랜덤한 위치에 생성됩니다. • 상대의 체력이 0이 되면 승리합니다.

* 구현방법 (캐릭터 선택 씬)

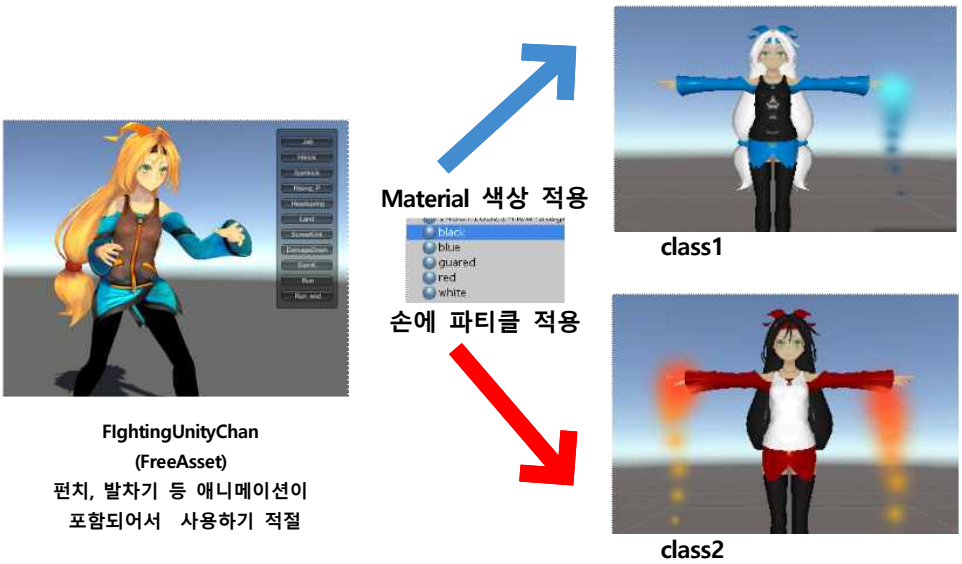


↳ 전체 화면 캡처

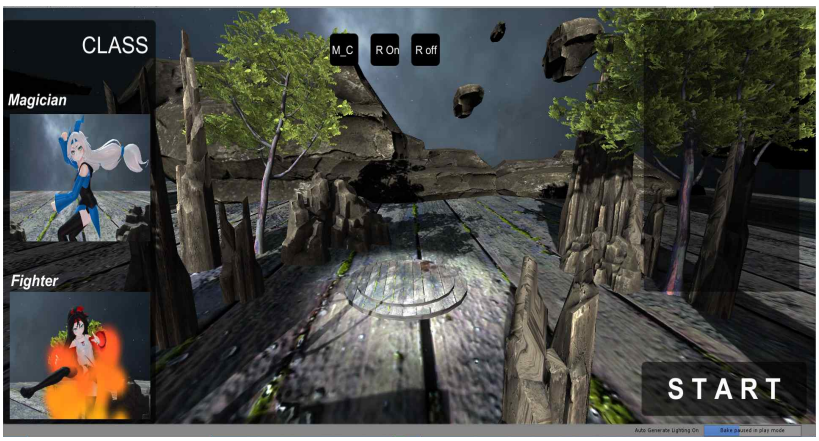




배경 구현


FlghtingUnityChan
(FreeAsset)
펀치, 발차기 등 애니메이션이
포함되어서 사용하기 적절
Material 색상 적용
black
blue
guared
red
white
손에 파티클 적용
class1
class2

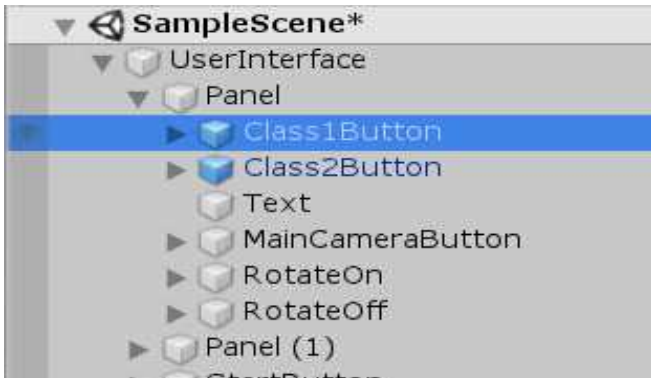
<캐릭터 구현>


CLASS
Magician
Fighter
M_C
R On
R off
START

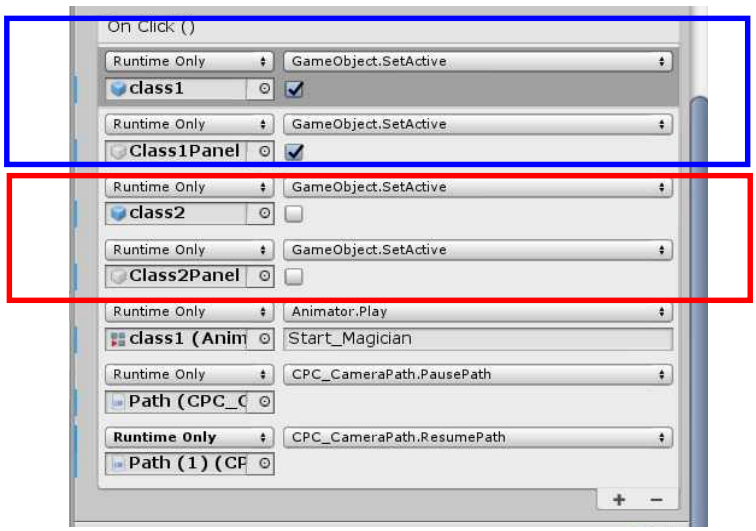
↳ 화면 구성

↳ 실행 화면

- 플레이어가 원하는 직업 버튼을 클릭하면 가운데 무대에 직업이 생성됩니다.(SetActive)
- 플레이어를 설명하는 패널 또한 SetActive됩니다.



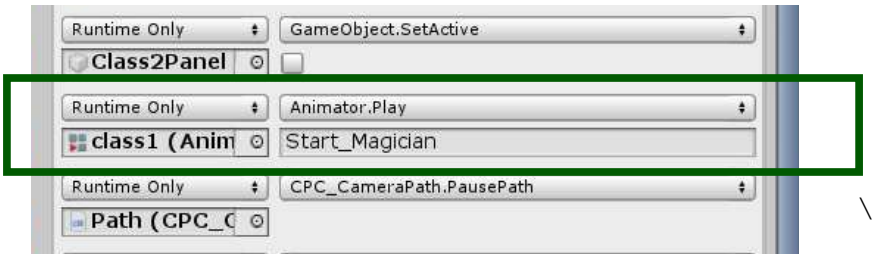
⇩



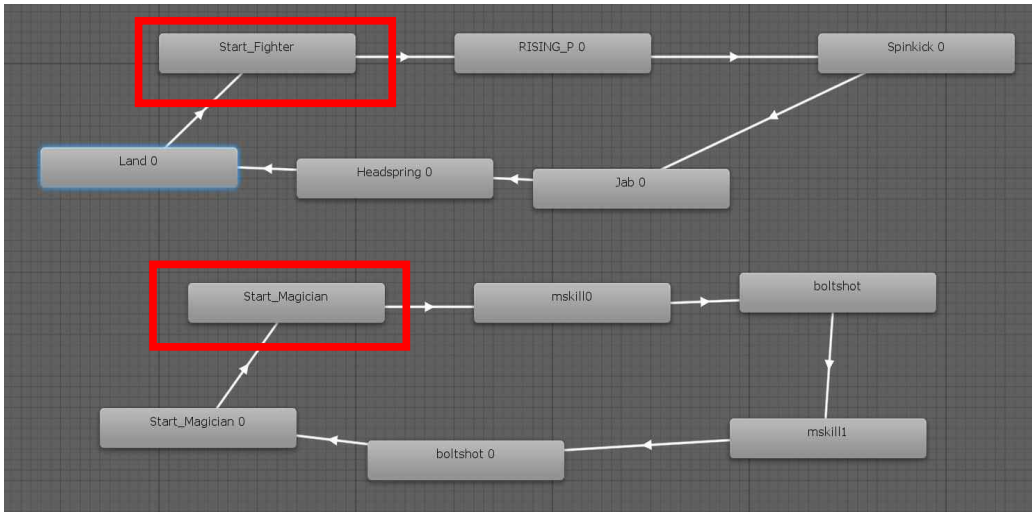
Class1Button 클릭시 >
캐릭터, 패널 SetActive(true)

Class2Button 클릭시 >
캐릭터, 패널 SetActive(true)

- **직업은 애니메이션을 시작합니다. > On Click()에서 Animator.Play이용**



⇩

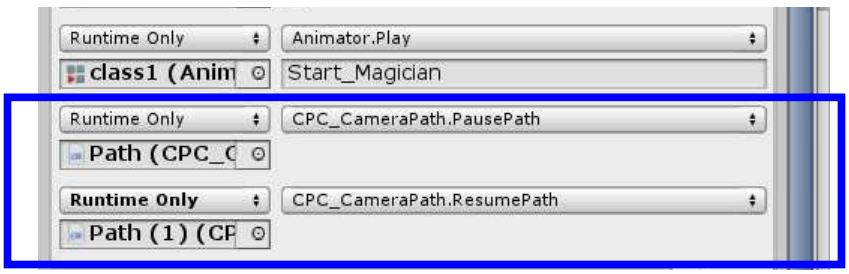


Animator.Play()에서 파라미터로 Start_Fighter, Start_Magician을 각각 주면 해당 애니메이터의 애니메이션이 순차적으로 실행된다.

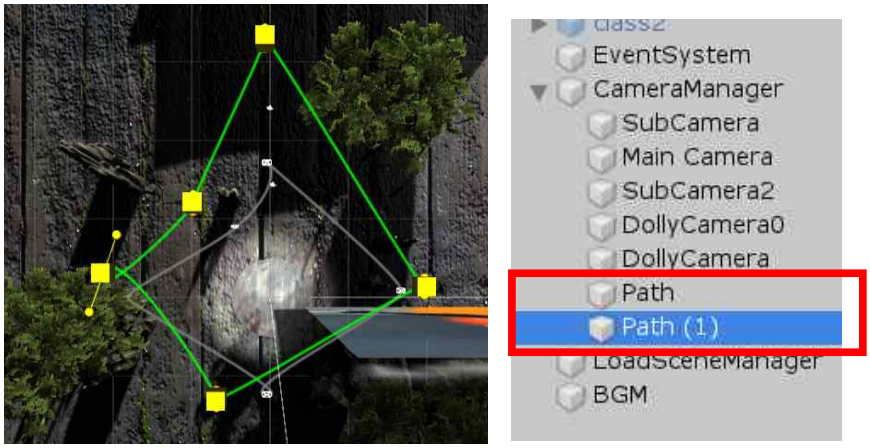
- **직업을 비추는 카메라가 움직입니다.**

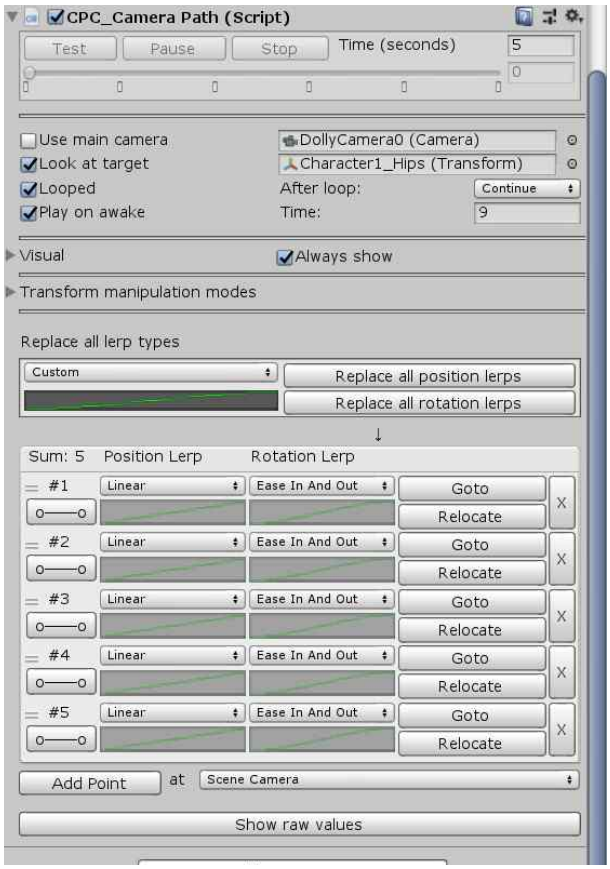
  > CameraManager 스크립트를 통해서 총 5개의 카메라의 ON/OFF를 관리

  1. 메인카메라
  2. class1 전용 움직이는 카메라
  3. class2 전용 움직이는 카메라
  4. class1 전용 서브 카메라
  5. class2 전용 서브 카메라



↳ 움직이는 카메라가 path를 따라서 움직이도록 명령





<Camera Path 생성 asset (CPC_Camera Path, free asset사용)>

- **Start버튼을 누르면 게임이 시작됩니다. >** LoadSceneManager

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class LoadSceneManager : MonoBehaviour
```

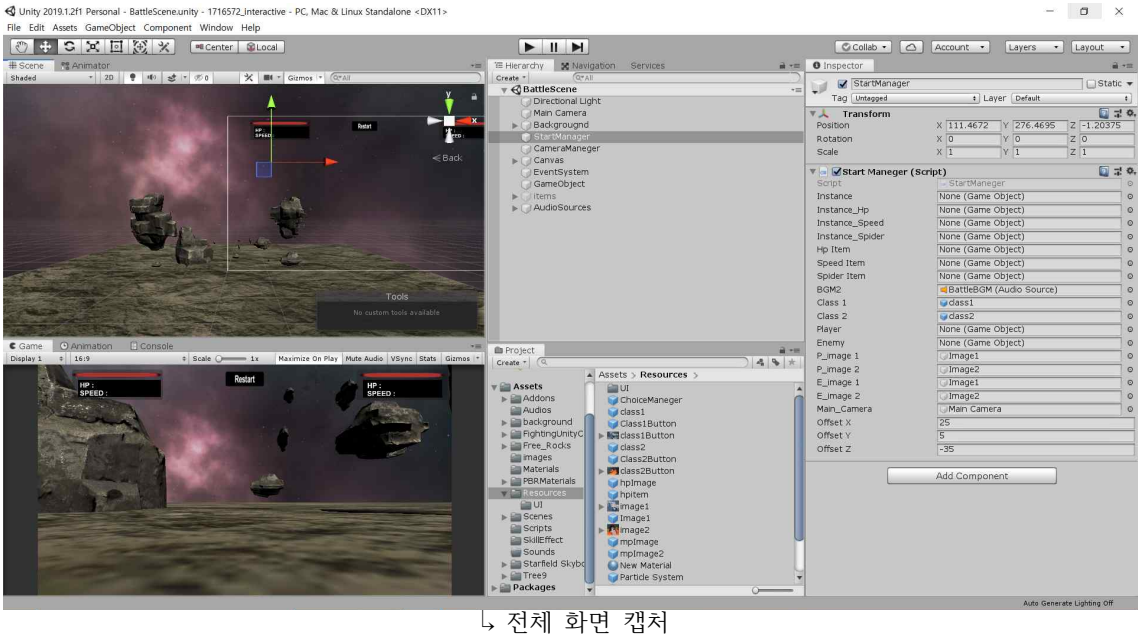
```
{
    public Button StartButton;

    void Start()
    {
        StartButton.onClick.AddListener(LoadingScene);
    }

    void LoadingScene()
    {
        SceneManager.LoadScene("BattleScene");
    }
}
```

- **유저의 선택 정보를 저장하고 관리하기 위해 ChoiceManager라는 스크립트를 생성하고, Resource폴더에 저장하였습니다.**

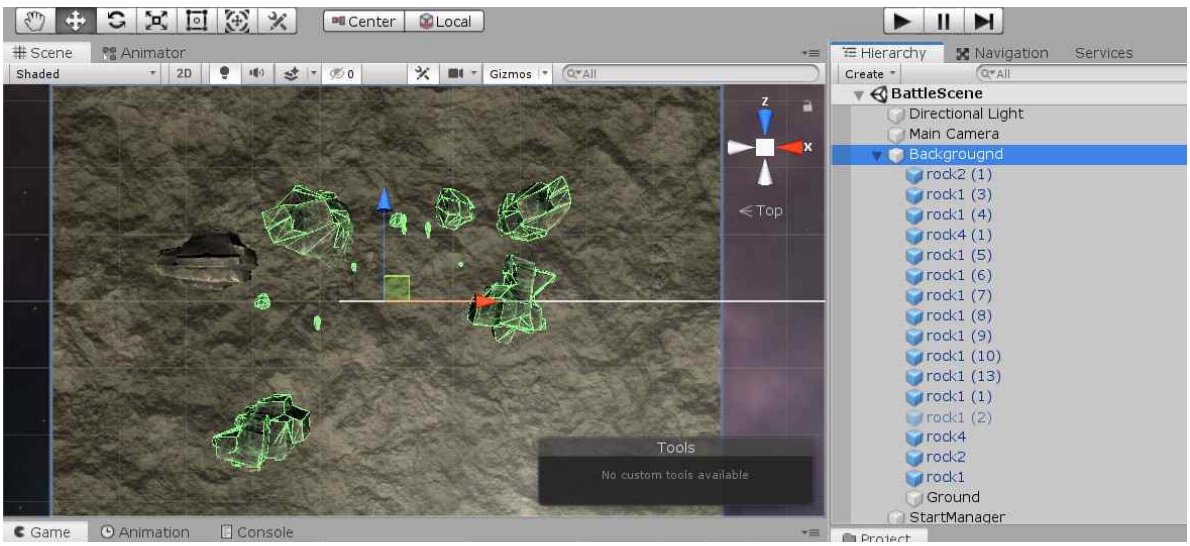
ChoiceManager.cs

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

public class ChoiceManager : MonoBehaviour
{
    public bool isClass1 = false; //캐릭터 선택 씬에서 격투 씬으로 넘어갈 때, 플레이어의 선택을
넘겨주고 관리한다.

    public void setClass1()
    {
        isClass1 = true;
    }
    public void offClass1()
    {
        isClass1 = false;
    }

    public bool getClass1()
    {
        return isClass1;
    }

}
```

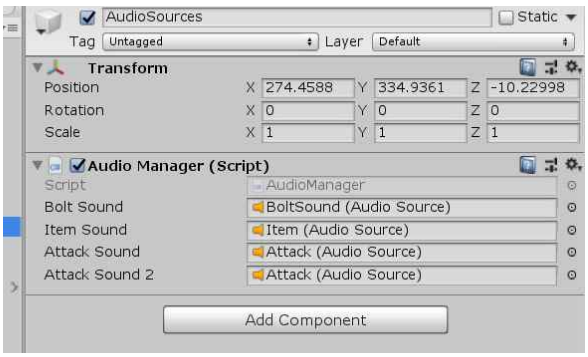
- **구현 방법 (전투 씬)**



↳ 전체 화면 캡처



↳ 배경 구현



***AudioManager.cs** (효과음 및 Audio 관리 스크립트)

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

public class AudioManager : MonoBehaviour //prefab에서 관리할 경우 오류로 인해 따로 관리
{
    public AudioSource BoltSound;
    public AudioSource ItemSound;
    public AudioSource AttackSound;
    public AudioSource AttackSound2;

    // Update is called once per frame
    void Update()
    {
        if (MoveControl_1.boltOn)
        {
            BoltSound.Play();
            MoveControl_1.boltOn = false;
        }

        if (itemCollider.itemOn)
        {
            ItemSound.Play();
            itemCollider.itemOn = false;
        }

        if (HitColliderM.attackOn)
        {
            AttackSound.Play();
            HitColliderM.attackOn = false;
        }

        if (HitCollider.attackOn1)
        {
            AttackSound2.Play();
            HitCollider.attackOn1 = false;
        }


    }
}
```

↳ 화면 구성

- 플레이어는 화면의 좌측에, 컴퓨터는 좌측에 배치

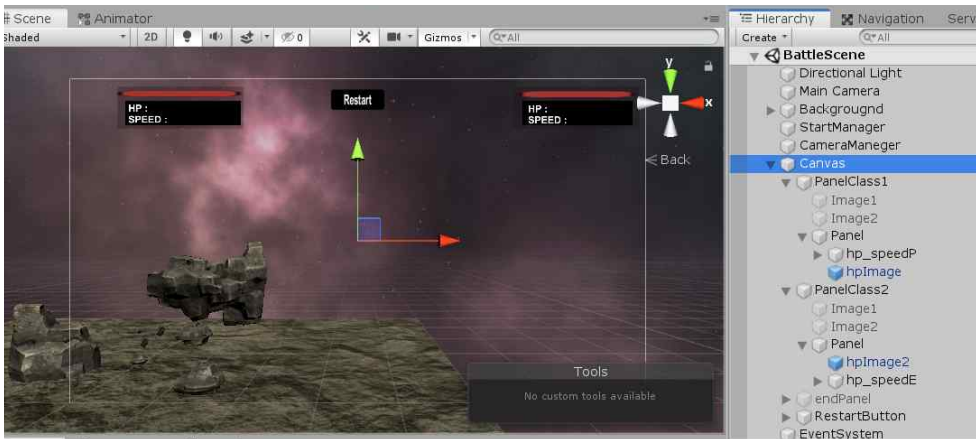
StartManager 스크립트

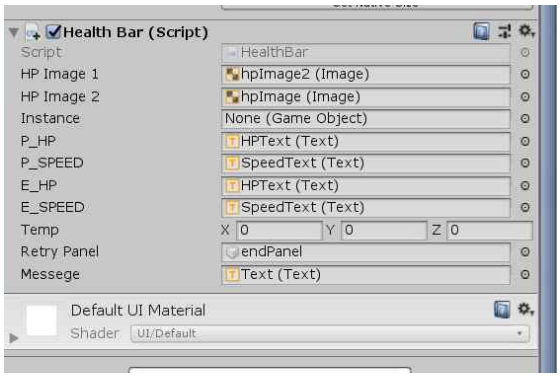
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class StartManeger : MonoBehaviour
{
    public GameObject instance; // class1을 선택했는지 알아보기 위한 게임오브젝트

    public GameObject instance_Hp;
    public GameObject instance_Speed;
    public GameObject instance_Spider;
    public GameObject hpItem; // HPitem
    public GameObject speedItem; //speed 증가아이템
    public GameObject spiderItem; //speed 감소아이템

    public AudioSource BGM2;

    public GameObject class1;
    public GameObject class2;
    public GameObject player;
    public GameObject enemy;

    public GameObject p_image1; // player 캐릭터 hp, mp 창 image
    public GameObject p_image2;

    public GameObject e_image1;
    public GameObject e_image2;

    public GameObject Main_Camera;

    public float offsetX;
    public float offsetY;
    public float offsetZ;



    void Start()
    {
        BGM2.Play(); //배경음악 재생
        instance = (GameObject)Resources.Load("ChoiceManeger");
```

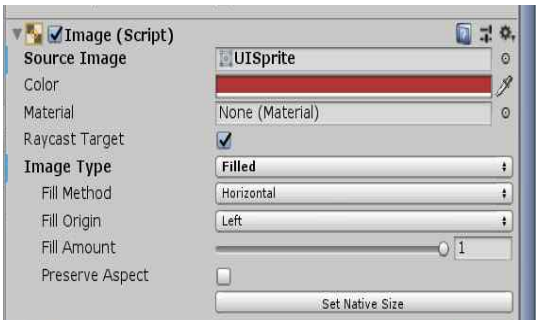
```
        hpItem = (GameObject)Resources.Load("hpitem");
        speedItem = (GameObject)Resources.Load("speeditem");
        spiderItem = (GameObject)Resources.Load("slowItem");

        InvokeRepeating("CreateHPItem", 5f, 10f); //3가지 아이템들을 꾸준하게 생성
        InvokeRepeating("CreateSpeedItem", 10f, 20f);
        InvokeRepeating("CreateSpiderItem", 15f, 25f);


        class1.SetActive(true);
        class2.SetActive(true);

        if (instance.GetComponent<ChoiceManager>().getClass1())
         //플레이어가 choice Scene에서 클래스 1 선택시 getClass -> true값 return
        {
            player = (GameObject)Instantiate(class1, new Vector3(0, 5, 0),
Quaternion.Euler(new Vector3(0,90,0)));
            enemy = (GameObject)Instantiate(class2, new Vector3(50, 5, 0),
Quaternion.Euler(new Vector3(0, -90, 0)));  //알맞은 위치에 플레이어와 적을 생성

            p_image1.SetActive(true);
            p_image2.SetActive(false);

            e_image2.SetActive(true);
            e_image1.SetActive(false);

        }
        else //플레이어가 choice Scene에서 클래스 2 선택
        {
            player = (GameObject)Instantiate(class2, new Vector3(0, 5, 0),
Quaternion.Euler(new Vector3(0, 90, 0)));
            enemy = (GameObject)Instantiate(class1, new Vector3(50, 5, 0),
Quaternion.Euler(new Vector3(0, -90, 0))); //알맞은 위치에 플레이어와 적을 생성

            p_image2.SetActive(true);
            p_image1.SetActive(false);

            e_image1.SetActive(true);
            e_image2.SetActive(false);


        }

    }

    void Update()
    {
        Vector3 FixedPos =new Vector3(player.transform.position.x + offsetX,
player.transform.position.y + offsetY, player.transform.position.z + offsetZ);
        Main_Camera.transform.position = FixedPos; //player 따라다니는 카메라
    }


    void CreateHPItem() //아이템 생성 함수들
    {
        int x = Random.Range(0, 125);
        instance_Hp = (GameObject)Instantiate(hpItem, new Vector3(x-10 , 2, 0),
Quaternion.Euler(0, 180, 0));

    }
    void CreateSpeedItem()
    {
        int x2 = Random.Range(0, 125);
        instance_Speed = (GameObject)Instantiate(speedItem, new Vector3(x2-10 , 2, 0),
Quaternion.Euler(0, 180, 0));
```

```
    }
    void CreateSpiderItem()
    {
        int x3 = Random.Range(0, 125);
        instance_Spider = (GameObject)Instantiate(spiderItem, new Vector3(x3-10 , 2, 0),
Quaternion.Euler(0, 180, 0));

    }



}
```

- **화면 상단에는 각 캐릭터의 체력과 스피드가 표시됩니다.**







- **HealthBar.cs** <체력관리, 캔버스의 Text관리, 게임 재시작 관리 스크립트 >

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class HealthBar : MonoBehaviour //각 캐릭터의 Hpbar관리 + 게임 재시작 관리 스크립트
{
    public Image HPImage1;
    public Image HPImage2;
    public GameObject instance;

    public Text p_HP;
```

```
    public Text p_SPEED;

    public Text e_HP;
    public Text e_SPEED;

    public Vector3 temp;
    public GameObject retryPanel;
    public Text messege;


    public void RetryButtonDown() //재시작 버튼을 눌렀을 경우 > 캐릭터 선택 씬으로 이동한다.
    {
        Debug.Log("재시작");
        Application.LoadLevel("SampleScene");
        MoveControl_1.class1HP = 100f;
        MoveControl_2.class2HP = 100f;

        MoveControl_2.speed = 20f;
        MoveControl_1.speed = 10f;

    }

    // Start is called before the first frame update
    void Start()
    {
        retryPanel.SetActive(false);
        instance = (GameObject)Resources.Load("ChoiceManeger");

        if (instance.GetComponent<ChoiceManager>().getClass1()) { //유저와 적의 hpBar위치를
설정한다.
            temp = new Vector3(HPImage1.transform.position.x,
HPImage1.transform.position.y, HPImage1.transform.position.z);
            HPImage1.transform.position = HPImage2.transform.position;
            HPImage2.transform.position = temp;

        }
    }


    // Update is called once per frame
    void Update() //hp bar와 hp, speed text 설정
    {
        HPImage1.fillAmount = MoveControl_1.class1HP / 100f;
        HPImage2.fillAmount = MoveControl_2.class2HP / 100f;

        if (instance.GetComponent<ChoiceManager>().getClass1()) {
            p_HP.text = "HP = " + (int)MoveControl_1.class1HP;
            e_HP.text = "HP = " + (int)MoveControl_2.class2HP;

            p_SPEED.text = "SPEED = " + (int)MoveControl_1.speed;
            e_SPEED.text = "SPEED = " + (int)MoveControl_2.speed;
        }
        else
        {
            p_HP.text = "HP = " + (int)MoveControl_2.class2HP;
            e_HP.text = "HP = " + (int)MoveControl_1.class1HP;

            p_SPEED.text = "SPEED = " + (int)MoveControl_2.speed;
            e_SPEED.text = "SPEED = " + (int)MoveControl_1.speed;
        }




        if ((MoveControl_2.class2HP <= 0) || (MoveControl_1.class1HP <= 0)) //게임이 종료,
```

```
종료 패널 띄우고 승리/패배 메시지를 띄움
        {
            if ((instance.GetComponent<ChoiceManager>().getClass1()) &&
(MoveControl_2.class2HP <= 0)) //유저가 1선택, 2가 죽은 경우
            {
                retryPanel.SetActive(true);
                messege.text = "YOU WIN";
                MoveControl_2.ani.SetBool("DamageDown", true);
            }
            else if ((instance.GetComponent<ChoiceManager>().getClass1() == false) &&
(MoveControl_1.class1HP <= 0)) //유저가 2 선택, 1이 죽은 경우
            {
                retryPanel.SetActive(true);
                messege.text = "YOU WIN";
                MoveControl_1.ani.SetBool("DamageDown", true);
            }
            else if ((instance.GetComponent<ChoiceManager>().getClass1()) &&
(MoveControl_1.class1HP <= 0)) //유저가 1선택, 유저가 죽은 경우
            {
                retryPanel.SetActive(true);
                messege.text = "YOU LOSE";
                MoveControl_1.ani.SetBool("DamageDown", true);
            }

            else if ((instance.GetComponent<ChoiceManager>().getClass1() == false) &&
(MoveControl_2.class2HP <= 0)) //유저가 2 선택, 유저가 죽은 경우
            {
                retryPanel.SetActive(true);
                messege.text = "YOU LOSE";
                MoveControl_2.ani.SetBool("DamageDown", true);
            }


        }
    }
}
```

- **플레이어는 방향키와 Ctrl, Alt로 조작하고, 컴퓨터는 자동 조작합니다.**

  *** MoveControl_1 스크립트, (MoveControl_2 또한 같은 원리로 작성)**

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

public class MoveControl_1 : MonoBehaviour //class1을 위한 조작 스크립트
{

    Rigidbody rbody;
    public static Animator ani;

    public static float speed = 15f;
    public static float class1HP = 100f; //체력 관리


    private GameObject instance2;
    public GameObject skill1;

    public GameObject MainC;
    public GameObject instance;

    public bool updateClass = false; //Start 함수에서 auto기능 on/off를 위함

    private bool toBack=false;
    private bool changeDirection = false; //캐릭터 앞 뒤 방향 전환을 위한 변수
```

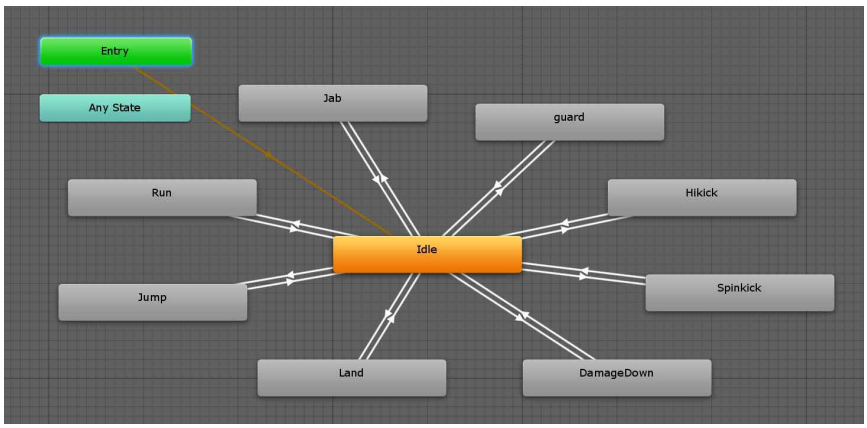
```
    public static bool boltOn = false;
    public static bool guarded = false; //방어

    private bool IsJumping = false; //점프 판정 위한 변수
    private bool doSomething = false; //동작 중첩 방지를 위한 변수
    public GameObject kickCol;

    private bool canEnergyBolt = true;

    private bool canB = true; //자동이동시 과도한 가속도 방지
    private bool canF = true;

    private bool autoPunch = false;
    private bool autoGuard = false;


    private float timer = 0f;

    void Start()
    {
        kickCol.SetActive(false);
        instance = (GameObject)Resources.Load("ChoiceManeger");
        skill1 = (GameObject)Resources.Load("skill_1");
        rbody = GetComponent<Rigidbody>();
        rbody.isKinematic = false;
        rbody.useGravity = true;


        ani = GetComponent<Animator>();

        if (instance.GetComponent<ChoiceManager>().getClass1()) //자동이동을 사용할지 ,
플레이어가 직접 조종할지 Start함수에서 결정
        {
            updateClass = true; //유저가 직접 조종
        }
        else {

            Debug.Log("class1 autoMove, Start Coroutine");
            StartCoroutine(auto()); //자동 이동 이용
        }

    }
    IEnumerator auto() //자동 이동 구현
    {
        speed = 15f;
        while (class1HP>0) //죽을때까지 반복
        {
            int x = Random.Range(0, 5);  //자동이동 구현  random number 생성해서 움직임
조절 0:공격 1:가드 2:발차기 3:오른쪽 이동 4: 왼쪽 이동
            yield return new WaitForSeconds(1);
            kickCol.SetActive(false);
            Debug.Log(x);
            ani.SetBool("Run", false);

            if (toBack)
            {
                transform.rotation *= Quaternion.Euler(0, 180f, 0);
                toBack = false;
            } //뒤 돌고 있다면 다시 앞을 보도록


            switch (x) {
                case 0: //에너지볼트
                    if (canEnergyBolt)
```

```
                    {
                        canEnergyBolt = false;
                        boltOn = true;
                        ani.SetBool("Jab", true);
                        Vector3 pos = new Vector3(gameObject.transform.position.x,
gameObject.transform.position.y + 6, gameObject.transform.position.z);
                        instance2 = (GameObject)Instantiate(skill1, pos, Quaternion.identity);
                        instance2.GetComponent<Rigidbody>().AddForce(transform.forward *
800);
                        yield return new WaitForSeconds(1);
                        boltOn = true;
                        ani.SetBool("Jab", true);
                        Vector3 pos1 = new Vector3(gameObject.transform.position.x,
gameObject.transform.position.y + 6, gameObject.transform.position.z);
                        instance2 = (GameObject)Instantiate(skill1, pos1, Quaternion.identity);
                        instance2.GetComponent<Rigidbody>().AddForce(transform.forward *
800);

                    }
                    break;

                case 1: //가드
                    ani.SetBool("guard", true);
                    guarded = true;
                    yield return new WaitForSeconds(2);
                    ani.SetBool("guard", false);
                    guarded = false;

                    break;

                case 2://발차기 + 펀치
                    kickCol.SetActive(true);
                    ani.SetBool("Hikick", true);
                    yield return new WaitForSeconds(1);
                    ani.SetBool("Jab", true);


                    break;

                case 3: //뒤로가기
                    if (canB)
                    {
                        transform.rotation *= Quaternion.Euler(0, 180f, 0);
                        ani.SetBool("Run", true);
                        rbody.AddForce(Vector3.right * 1300f * (speed * 0.1f),
ForceMode.Impulse);
                        toBack = true;
                        canB = false;
                        canF = true;
                    }
                    break;

                case 4: //앞으로 가기
                    if (canF)
                    {
                        ani.SetBool("Run", true);
                        rbody.AddForce(Vector3.left * 1300f * (speed * 0.1f),
ForceMode.Impulse);
                        canB = true;
                        canF = false;
                    }
                    break;

                default:
                    break;
            }
```

```
    }

}

    // Update is called once per frame
    void Update()
{
    timer += 1f*Time.deltaTime;
    if (timer > 5f) //에너지볼트를 5초에 한번만 쓸 수 있도록 타이머 설정
    {
        canEnergyBolt = true;
        timer = 0f;
    }

   if (updateClass)  //유저가 직접 조종
    {

        float h = Input.GetAxisRaw("Horizontal");

        if (h == 0) ani.SetBool("Run", false);
        else ani.SetBool("Run", true);

        // 앞뒤 방향 전환 체크(y축 180도 회전 유무 파악을 위해 )
        if ((h < 0) && (toBack == false))
        {
            changeDirection = true;
            toBack = true;
        }

        if ((h > 0) && (toBack == true)) // 앞뒤 회전
        {
            changeDirection = true;
            toBack = false;
        }

        if (changeDirection)
        {
            Debug.Log("회전");
            transform.rotation *= Quaternion.Euler(0, 180f, 0);
            changeDirection = false;
        }


        //캐릭터 움직임      (좌우)
        if ((Input.GetAxisRaw("Horizontal") < 0) && (doSomething == false))
        {
            kickCol.SetActive(false);
            transform.position += Vector3.left * speed * Time.deltaTime;

        }
        else if ((Input.GetAxisRaw("Horizontal")) > 0 && (doSomething == false))
        {
            kickCol.SetActive(false);
            transform.position += Vector3.right * speed * Time.deltaTime;

        }

        //방어(꾹 누르기)
        if (Input.GetKey(KeyCode.DownArrow))
        {
            kickCol.SetActive(false);
            ani.SetBool("guard", true);
            doSomething = true;
```

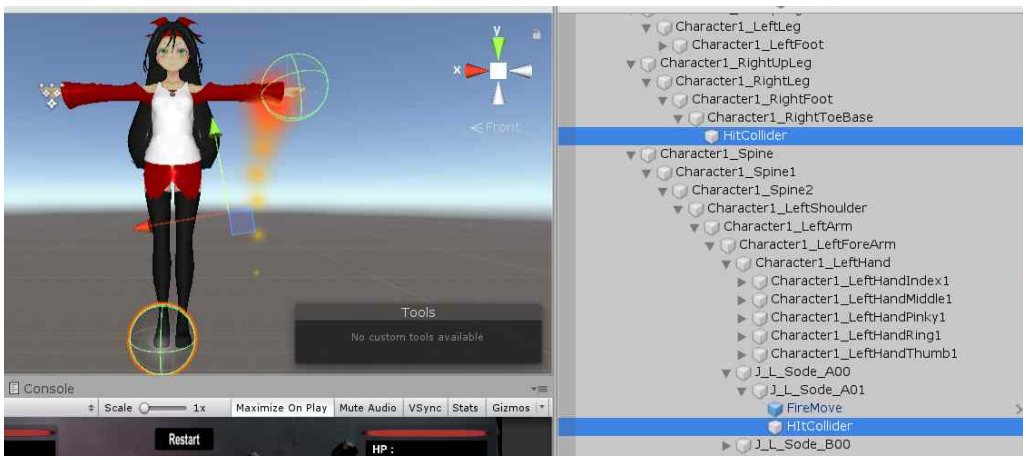
```
                guarded = true;
            }
            else {
                ani.SetBool("guard", false);

                doSomething = false; //가드와 움직임 동시 불가능
                guarded = false;
            }

            //공격 키 (에너지볼트)
            if (Input.GetKeyDown(KeyCode.LeftControl) && (doSomething ==
false)&&(canEnergyBolt))
            {
                 kickCol.SetActive(false);
                boltOn = true;
                canEnergyBolt = false;    //에너지볼트를 5초에 한번씩만 사용할 수 있도록
bool변수 이용
                 Vector3 pos = new Vector3(gameObject.transform.position.x,
gameObject.transform.position.y + 6, gameObject.transform.position.z);
                 ani.SetBool("Jab", true);
                 instance2 = (GameObject)Instantiate(skill1, pos, Quaternion.identity);
                 instance2.GetComponent<Rigidbody>().AddForce(transform.forward * 800);
                 transform.position += Vector3.up * speed * Time.deltaTime; // 타격감

            }

            if (Input.GetKeyDown(KeyCode.LeftAlt))
            {
                kickCol.SetActive(true); //발차기 collider Active 시킴
                ani.SetBool("Hikick", true);

            }

            if (Input.GetKeyDown(KeyCode.UpArrow)) //점프
            {

                if (!IsJumping)
                {
                    IsJumping = true;
                    Debug.Log("점프");
                    rbody.AddForce(Vector3.up * 1100f, ForceMode.Impulse);
                }

            }



                if (class1HP <= 0) //플레이어 죽음
            {
                ani.SetBool("DamageDown", true);
                speed = 0f;
            }

        }


    }



    private void OnCollisionEnter(Collision other) //점프를 여러번 할 수 없도록 설정
    {
        if (other.transform.tag == "ground")
        {
            IsJumping = false;
        }
```

```
    }

}
```



<사용한 애니메이터>

- **충돌/타격 감지**



***HitColldier.cs** (파이터 전용 HItCollider)

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

public class HitCollider : MonoBehaviour
{

    public string HitName; //파이터의 손과 발에 적용된 collider의 충돌을 감지하는 스크립트이다.
    public static bool attackOn1 = false;


    void OnTriggerEnter(Collider other)
    {
        if (other.transform.tag == "class1")
        {

            if (!MoveControl_1.guarded) //매지션이 가드하고 있지 않은 경우 데미지가 적용된다.
            {
                if (HitName == "punch") {
```

```
                    MoveControl_1.class1HP -= 10f;

                    attackOn1 = true;
                }

                else if (HitName == "kick") {
                    MoveControl_1.class1HP -= 20f;

                    attackOn1 = true;
                }

            }
            else Debug.Log("guarded");
        }

    }

}
```

HitColldierM.cs (매지션 전용 충돌 감지 스크립트)

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

public class HitColliderM : MonoBehaviour
{
    public string HitName; //매지션의 에너지볼트와 발에 적용된 collider의 충돌을 감지하는 스크립
트이다.
    public static bool attackOn = false;


    void OnTriggerEnter(Collider other)
    {
        if (other.transform.tag == "class2")
        {
            if (!MoveControl_2.guarded) //파이터가 가드하고 있지 않은 경우, 데미지가 적용된다.
            {
                if (HitName == "energy")
                {
                    MoveControl_2.class2HP -= 20f;

                    attackOn = true;
                    Destroy(gameObject);
                }

                else if (HitName == "kick")
                {
                    MoveControl_2.class2HP -= 10f;
                    attackOn = true;


                }

            }

            else Debug.Log("guarded");
        }
    }}
```

- 체력아이템, 스피드아이템, 슬로우아이템 3가지가 맵의 랜덤한 위치에 생성됩니다.

<원기둥, 큐브 등 3D 오브젝트로 제작>

* StartManager에서 아이템 생성 관련 코드

```
        ....
    InvokeRepeating("CreateHPItem", 5f, 10f); //3가지 아이템들을 꾸준하게 생성
   InvokeRepeating("CreateSpeedItem", 10f, 20f);
   InvokeRepeating("CreateSpiderItem", 15f, 25f);
        ....
  void CreateHPItem() //아이템 생성 함수들
    {
        int x = Random.Range(0, 125);
        instance_Hp = (GameObject)Instantiate(hpItem, new Vector3(x-10 , 2, 0),
Quaternion.Euler(0, 180, 0));

    }
    void CreateSpeedItem()
    {
        int x2 = Random.Range(0, 125);
        instance_Speed = (GameObject)Instantiate(speedItem, new Vector3(x2-10 , 2, 0),
Quaternion.Euler(0, 180, 0));

    }
    void CreateSpiderItem()
    {
        int x3 = Random.Range(0, 125);
        instance_Spider = (GameObject)Instantiate(spiderItem, new Vector3(x3-10 , 2, 0),
Quaternion.Euler(0, 180, 0));

    }
```

* ItemCollider.cs (아이템에 적용된 충돌 관리 함수)

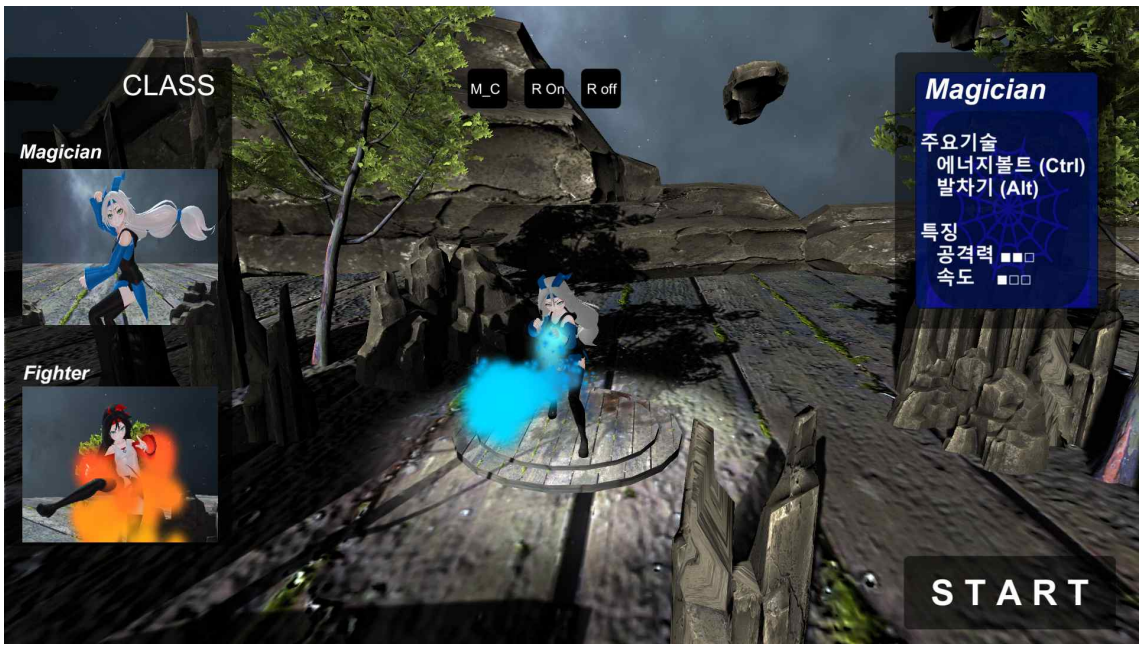
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class itemCollider : MonoBehaviour //누가 어떤 아이템과 접촉하였는지 Tag로 관리, 효과
적용
{

    public GameObject instance;
    public static bool itemOn=false;


    void Start() {
        instance = (GameObject)Resources.Load("ChoiceManeger");



    }



    void OnTriggerEnter(Collider other)
    {
        if ((transform.tag == "hpitem") && (other.transform.tag == "class1")) //hp아이템 > hp를
10 올려준다
        {
            if(MoveControl_1.class1HP<100) MoveControl_1.class1HP += 10f;
            itemOn = true;
            Destroy(gameObject);


        }
        else if((transform.tag == "hpitem") && (other.transform.tag == "class2"))
        {
            if (MoveControl_2.class2HP < 100) MoveControl_2.class2HP += 10f;
            itemOn = true;
            Destroy(gameObject);
        }
        if ((transform.tag == "speeditem") && (other.transform.tag == "class1")) //속도 아이템 >
속도를 10 올려준다
        {
            if (MoveControl_1.speed < 25) //MAX값 설정
            {
                MoveControl_1.speed += 10f;
            }
            else Debug.Log("MAX speed!");
            itemOn = true;

            Destroy(gameObject);
        }

        else if ((transform.tag == "speeditem" )&& (other.transform.tag == "class2")) {
            if (MoveControl_2.speed < 25)
            {
                MoveControl_2.speed += 10f;
            }
            else Debug.Log("MAX speed!");
            itemOn = true;
            Destroy(gameObject);
        }


        if ((transform.tag == "spideritem") && (other.transform.tag == "class1")) //거미줄 아이템
```

> 속도를 5로 만든다.

```
        {

            MoveControl_1.speed = 5f;
            itemOn = true;
            Destroy(gameObject);

        }
        else if ((transform.tag == "spideritem") && (other.transform.tag == "class2"))
        {

            MoveControl_2.speed= 5f;

            itemOn = true;
            Destroy(gameObject);
        }

    }

}
```

- **실행화면 캡처**

<게임 시작>



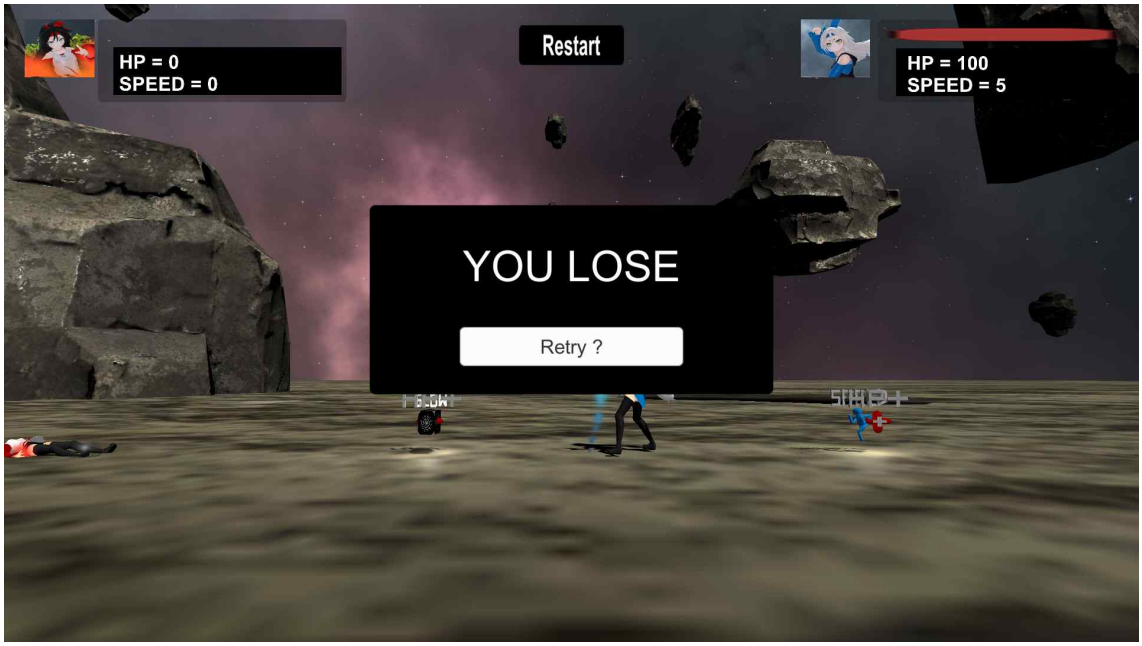
<매지션 버튼 클릭>

<파이터버튼 클릭>

<게임 시작>

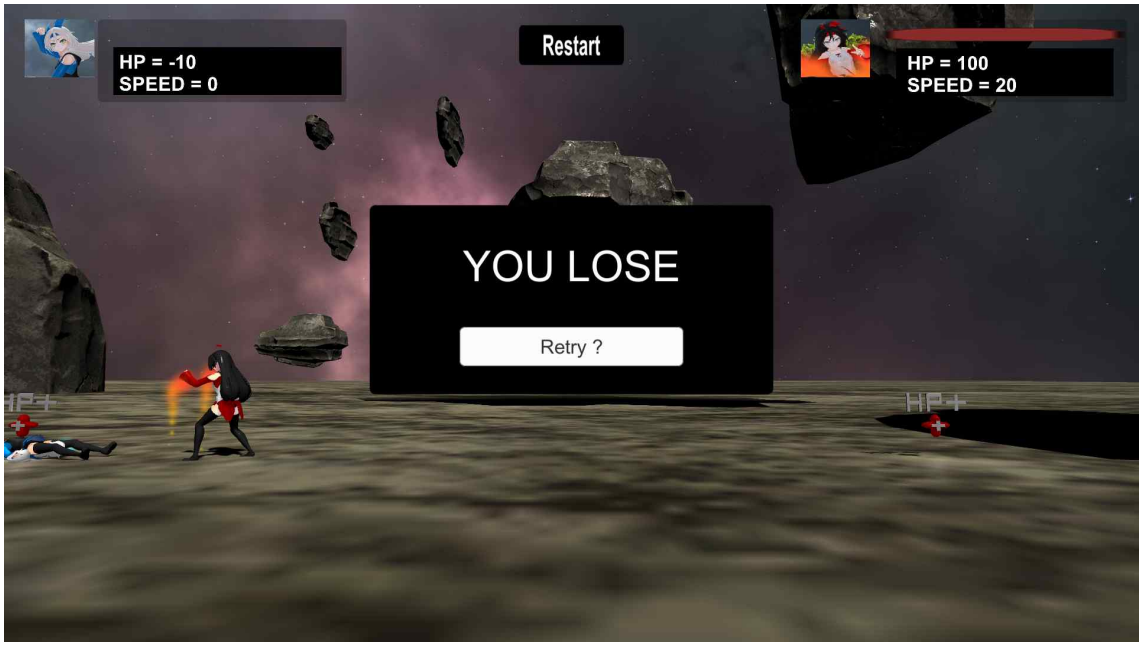
<맵에 생성된 아이템들>

<승리한 화면>



<패배한 화면>

<매지션으로 승리한 화면>



<매지션으로 패배한 화면>