컴가상 게임프로젝트 1조 보고서

- 1조(배박김조) :
    - 김동준(조장) : 게임 스크립트 작성 및 구현
    - 조태연 : 게임 사운드 선정 및 트러블 슈팅
    - 박지은 : 회의록 작성 및 맵 카드, PPT 제작
    - 배장원 : 게임 오브젝트 선정 및 맵 디자인

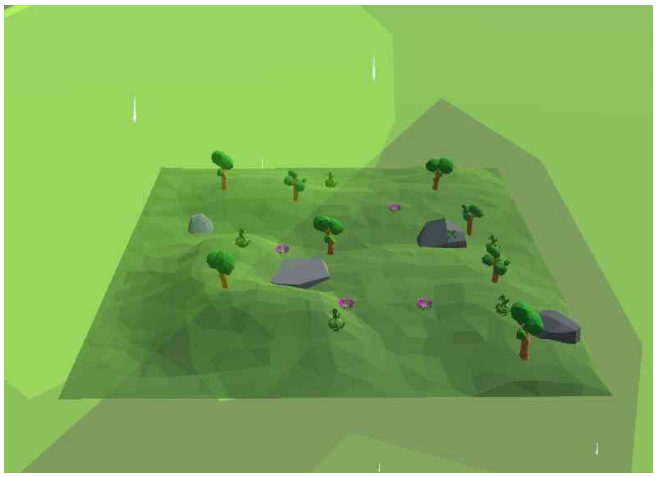
- 게임 제목 : 슬라임 vs 터틀

- 게임 개요 : 맵 카드 3개(숲, 사막, 암석지대)를 카메라로 인식하면 그 위에 맵과 아이템(사과, 가지)이 생성되고 두 개의 캐릭터(슬라임, 터틀)가 HP가 0이 될 때까지 PVP 배틀을 한다. 키보드로 플레이한다.

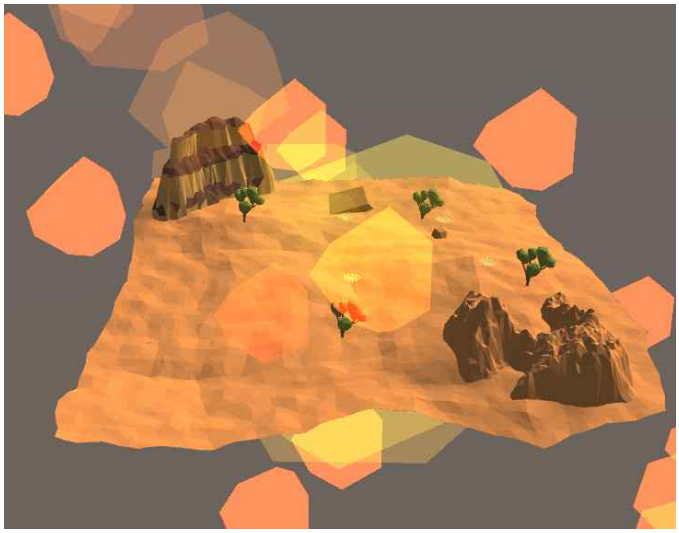
- 게임 스토리 : 지구에는 아직도 인류의 발길이 닿지 않은 곳이 있다. 캐나다에 동쪽에 위치한 늘 비가 오는 숲 지대, 카자흐스탄 서쪽에 위치한 바람이 많이 부는 암석 지대, 그리고 나미비아 남쪽에 위치한 불이 꺼지지 않는 사막지대. 이들 사이에는 한가지 공통점이 있다. 바로 미지의 생물 슬라임과 터틀이 산다는 것. 그들은 문명과 많이 동 떨어져 있는 만큼 그들 본연의 모습을 간직하고 있다. 동그란 몸통, 외눈박이. 튀는 색깔... 하지만 이들에게도 인류와 비슷한 것이 있다. 바로 사과를 좋아하고 브로콜리를 싫어하는 것! 그들에게 사과는 미식이자 생명이고 브로콜리는 독이자 죽음이다. 맛있는 사과를 먹기 위한 그들만의 치열한 전쟁! 악의는 없다! 그저 맛있는 사과를 먹기 위한 처절한 투쟁일뿐.

- 맵

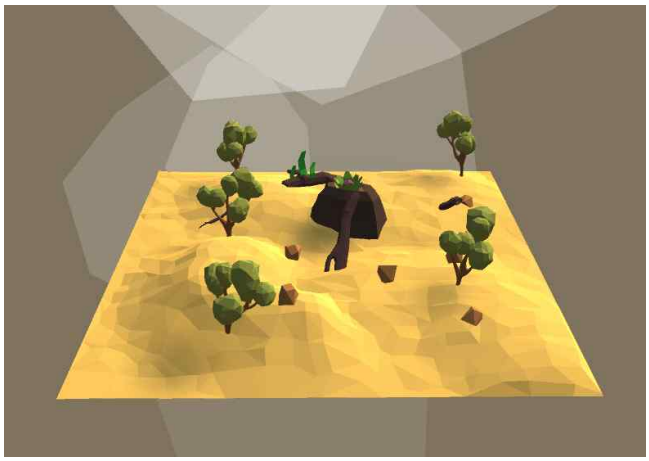
    - 숲 : 풀을 베이스로 나무, 돌, 꽃 오브젝트가 있다. 숲 느낌을 살리기 위해 초록색 파티클과 비가 내린다. 숲 테마 BGM이 나온다.



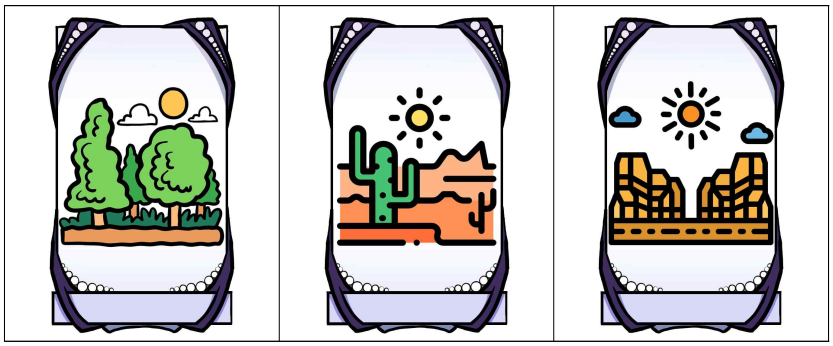
    - 사막 : 모래 베이스로 산, 돌, 작은 나무 오브젝트가 있다. 사막 느낌을 살리기 위해 불, 홍염 파티클을 사용한다. 사막 테마 BGM이 나온다.



- 암석지대 : 모래 베이스로 그루터기, 암석, 언덕 오브젝트가 있다. 암석지대 느낌을 살리기 위해 모래바람, 연기, 나뭇잎 파티클을 사용한다. 암석지대 테마 BGM이 나온다.



- 맵 카드 : 뷰포리아에서 Rating 5 에 해당하는 사진들을 찾아 테마에 맞춰 맵 카드를 제작했다.

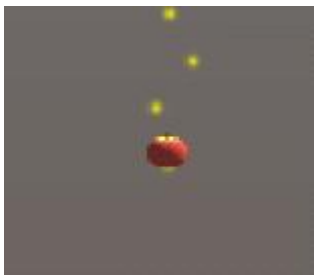


- 캐릭터
  - 슬라임 : 이동, 공격, 회피, 죽음 애니메이션과 사운드를 갖고 있다.

- 터틀 : 이동, 공격, 회피, 죽음 애니메이션과 사운드를 갖고 있다.

- 아이템

- 사과 : 노란색 파티클을 갖고 있으면 먹으면 HP를 10 회복한다.

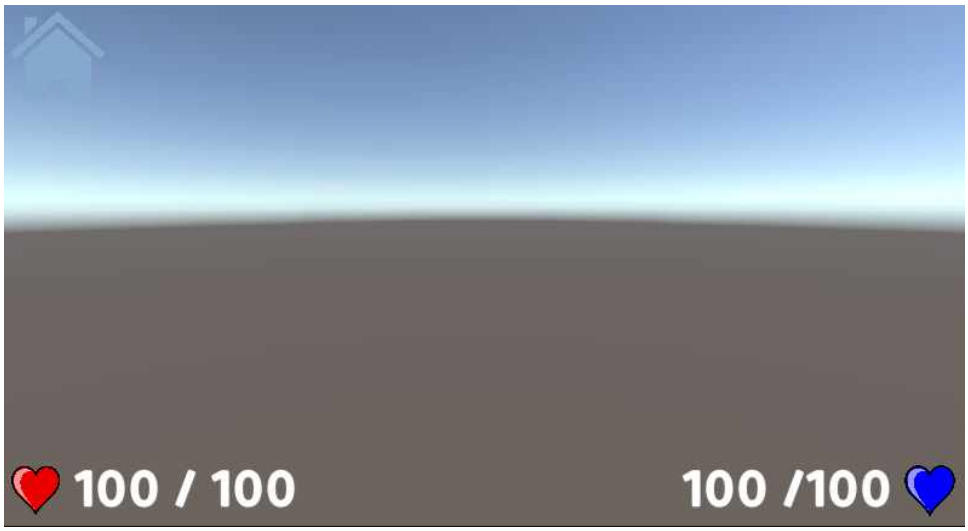


- 가지 : 보라색 파티클을 갖고 있으며 먹으면 HP를 10 감소한다.

- UI

- 메뉴 : 반투명 패널 위에 가운데에 'START' 버튼이 있다. 반투명 버튼이 있으므로 맵이 보인다. 'START' 버튼을 누르면 BGM과 함께 플레이어들이 나오고 인게임 UI가 나온다.

- 인게임 : 왼쪽 위에는 홈버튼이 있다. 누르면 메뉴 UI로 돌아간다. 좌측 하단과 우측 하단에는 각각 슬라임과 터틀의 체력이 나온다. 죽으면 HP가 XXX가 된다.



- C# 스크립트

- GameManager : 게임내의 UI를 통제하기 위한 스크립트.

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

public class GameManager : MonoBehaviour
{
    public Slime slime;
    public Turtle turtle;
    public GameObject gamePanel;
    public GameObject menuPanel;
    public Text slimeHealth;
    public Text turtleHealth;

    public void GameStart()
    {
```

```
        menuPanel.SetActive(false);
        gamePanel.SetActive(true);
        slime.gameObject.SetActive(true);
        turtle.gameObject.SetActive(true);
        slime.health = 100;
        turtle.health = 100;
    }

    public void home()
    {
        menuPanel.SetActive(true);
        gamePanel.SetActive(false);
        slime.gameObject.SetActive(false);
        turtle.gameObject.SetActive(false);
    }
    void LateUpdate()
    {
        slimeHealth.text = slime.health + " / " + slime.maxHealth;
        if (slime.health <= 0)
        {
            slimeHealth.text = "XXX";
        }

        turtleHealth.text = turtle.health + " / " + turtle.maxHealth;
        if (turtle.health <= 0)
        {
            turtleHealth.text = "XXX";
        }
    }
}
```

- Item : 아이템을 회전 시키기 위한 스크립트.

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

public class Item : MonoBehaviour
{
    Vector3 offset;

    void Update()
    {
        transform.Rotate(Vector3.up * 70 * Time.deltaTime);
    }
}
```

- Slime(Turtle) : 슬라임을 통제하기 위한 스크립트.(slime을 turtle로 바꾸면 turtle 스크립트)

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

public class Slime : MonoBehaviour
{

    public int lifeLevel = 0;

    public AudioSource attackSound;
    public AudioSource deathSound;
    public AudioSource dodgeSound;

    public Vector3 reactVec;
```

```
public float speed;
public GameObject[] weapons;
public bool[] hasWeapons;
public Turtle turtle;

public int health;
public int maxHealth;

BoxCollider boxCollider;
CapsuleCollider capsuleCollider;

MeshRenderer[] meshs;

float hAxis;
float vAxis;

bool wDown;
bool jDown;
bool iDown;
bool fDown;
bool attack;

bool isJump;
bool isDodge;
bool isFireReady;
Vector3 moveVec;

Rigidbody rigid;
Animator anim;

GameObject nearObject;
Weapon equipWeapon;
float fireDelay;

void Awake()
{
    rigid = GetComponent<Rigidbody>();
    anim = GetComponentInChildren<Animator>();
    boxCollider = GetComponent<BoxCollider>();
    capsuleCollider = GetComponent<CapsuleCollider>();
    meshs = GetComponentsInChildren<MeshRenderer>();

}

void Update()
{
    GetInput();
    Move();
    Turn();
    Attack();
    Dodge();
    Die();
}

void GetInput()
{
    hAxis = Input.GetAxisRaw("Horizontal");
    vAxis = Input.GetAxisRaw("Vertical");
    wDown = Input.GetButton("Walk");
    jDown = Input.GetButtonDown("Jump");
    iDown = Input.GetButtonDown("Interation");
    fDown = Input.GetButtonDown("Fire1");
}
```

```
void Move()
{
    moveVec = new Vector3(hAxis, 0, vAxis).normalized;

    transform.position += moveVec * speed * (wDown ? 0.3f : 1f) * Time.deltaTime;

    transform.position += moveVec * speed * Time.deltaTime;

    anim.SetBool("isRun", moveVec != Vector3.zero);
    anim.SetBool("isWalk", wDown);
}

void Turn()
{
    transform.LookAt(transform.position + moveVec);
}

void Die()
{
    if (health <= 0)
    {
        lifeLevel += 1;
        anim.SetBool("isDie", true);
    }
    else
    {
        anim.SetBool("isDie", false);
    }

    if(lifeLevel == 1)
    {
        deathSound.Play();
    }
}

void Attack()
{
    if (fDown && !isDodge)
    {
        anim.SetTrigger("doSwing");
        attackSound.Play();
    }
    if(fDown && !isDodge && attack == true)
    {
        turtle.health -= 10;
        attack = false;
    }
}

void OnCollisionEnter(Collision collision)
{
    if (collision.gameObject.tag == "Turtle")
    {
        attack = true;
    }
    if (collision.gameObject.tag == "Plus")
    {
        health += 10;
        nearObject = collision.gameObject;
        Destroy(nearObject);
    }
    if (collision.gameObject.tag == "Minus")
```

```
        {
            health -= 10;
            nearObject = collision.gameObject;
            Destroy(nearObject);
        }
    }

    void Dodge()
    {
        if (jDown && moveVec != Vector3.zero)
        {
            speed *= 2;
            anim.SetTrigger("doDodge");
            isDodge = true;
            dodgeSound.Play();

            Invoke("DodgeOut", 0.4f);
        }
    }


    void DodgeOut()
    {
        speed *= 0.5f;
        isDodge = false;
    }

}
```