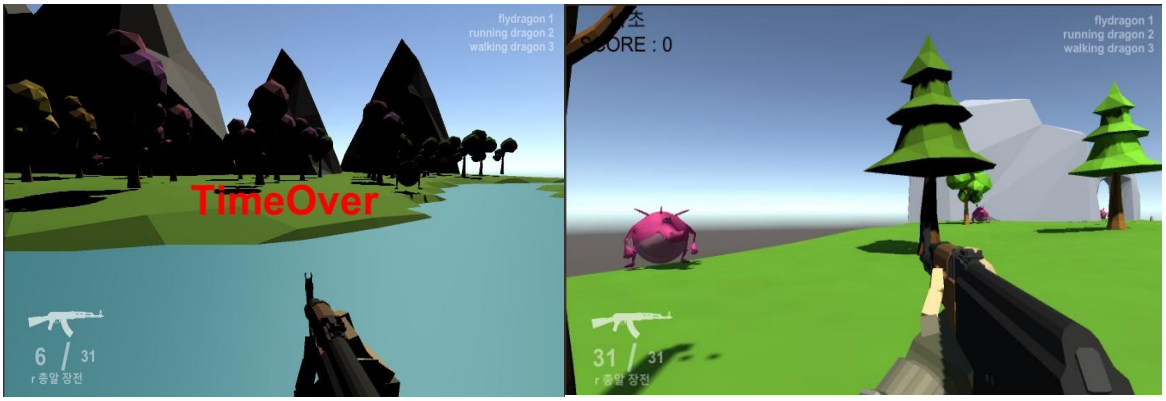
기말프로젝트 레포트

많은 프로젝트를 배우고 실습했지만 그중 나에게 가장 큰 뿌듯함과 재미를 주었던 프로젝트는 VR 프로젝트이었다. VR은 프로젝트 실습에서 어려움을 주었지만 이를 수정하고 프로젝트를 완성해 나갈 때 나에게 큰 성취감도 주었다. 그래서 내가 경험한 VR 게임을 토대로 유니티 VR FPS 게임을 만들어 보고 싶었다. 개인적으로 기존의 VR FPS 게임들은 어렵게만 느껴졌다. 나의 수준에 맞는 게임을 직접 제작해보고 싶어서 이 프로젝트를 기획하게 되었다. VR 콘텐츠 제작에서 장애물, 지형 만들기, tag 설정, script 작성, 안드로이드 build를 이 프로젝트의 개발범위로 잡았다.

✓ 기존프로젝트의 목표

기존 프로젝트는 지형 만들기, 장애물의 스크립트 작성, UI로 점수판 만들기를 목표로 잡았다. Emergency 경고가 나오면서 Start 버튼을 누르면 게임이 시작하도록 기획했다. Play를 시작하면 wasd 키를 사용해 플레이어를 움직이도록 스크립트를 작성하고 플레이어를 제작한다. 장애물 10개 정도 만들어 방향을 정하고 총에 1번만 맞으면 사라지도록 스크립트를 작성한다. 드래곤의 색을 두 가지로 나누고 각각 다른 속도를 정해주어 난이도를 조절한다. 플레이어가 장애물을 전부 다 맞추게 되었을 때 게임을 종료한다. 이 프로젝트의 큰 핵심은 점수로 순위를 정하는 것이었다.

✓ 제작된 프로젝트 설명



완성된 프로젝트는 VR fps 게임이기 때문에 main camera를 삭제해주었다. 플레이어의 프리팹에는 카메라, 움직임, 총알 장전, 총 쏘기 등 스크립트가 작성되어있어서 따로 제작하지 않았다. Asset store에서 player의 총, 드래곤, terrain을 만들 나무, 풀 등 다운받았다. 다운로드한 프리팹으로 직접

Terrain을 제작하고 box collider을 만들어 player가 지형 밖으로 나가지 못하도록 설정했다. 게임을 시작함과 동시에 시간제한이 60초인 timer가 시작되도록 스크립트를 작성했다. Player의 자식 계층에 canvas를 만들어 score, timer, 총알 개수, 총알 맞는 횟수, 총알 장전, time over를 만들어 주었다. Timer 밑에 Score가 있도록 스크립트를 작성해 시간과 점수를 동시에 확인할 수 있게 만들어 주었다. 그 다음 'Timer 스크립트'를 'player canvas', 'Time over' text를 'Timer 스크립트'의 'game over text'에 연결해주었다. 시간이 0이 되면 "time over"라고 화면 중앙에 뜨면서 게임이 멈추도록 게임을 개발했다.

장애물은 asset에서 다운로드한 드래곤 프리팹을 사용했다. 드래곤의 움직임의 종류가 다양했지만 이 프로젝트에서는 fly, run, walk 3가지의 움직임을 가진 드래곤만 사용했다. 드래곤은 최대 12마리로 정했다. 드래곤의 움직임에 따라 총을 맞는 횟수가 다르도록 기획했다. Fly dragon은 1번, running dragon은 2번, walking dragon은 3번 맞으면 사라지도록 스크립트를 작성했다. 날아다니거나 달리는 드래곤은 정확히 맞추기에 어려움이 있었고 총을 맞는 횟수에 차이를 두면 충분히 난이도 조절이 가능하다고 생각했다. 작성한 'Dragon move 스크립트'는 모든 드래곤 프리팹에 연결해 주었다. 드래곤은 각각 다른 위치에 있도록 제작했다. 드래곤이 좌우로만 움직이는 것이 아니라 드래곤마다 다른 속도로 player에 다가오도록 만들어 스릴감을 더해주었다.

✓ 프로젝트 핵심코드 설명

DragonMove.cs | Timer.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.AI;
public class DragonMove : MonoBehaviour
{

    public GameObject Player;
    public NavMeshAgent agent;
    public int energy = 2;
    // Start is called before the first frame update
    void Start()
    {
        agent = GetComponent<NavMeshAgent>();
    }

    // Update is called once per frame
    void Update()
    {
        agent.SetDestination(Player.transform.position);
    }

    void OnCollisionEnter(Collision coll)
    {
        if (coll.collider.CompareTag("Bullet"))
        {
            energy -= 1;

        }
        else return;
        if (energy <= 0)
        {
            Timer.score += 10;
            Destroy(this.gameObject, 0.0f);
        }
    }
}
```

Timer.cs

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

public class Timer : MonoBehaviour
{
    [SerializeField] GameObject gameOverText;
    public float maxTime = 60;
    private float currentTime;
    public Text textBox;
    Image timeBar;

    public static int score;


    // Use this for initialization
    void Start()
    {
        gameOverText.SetActive(false);
        timeBar = GetComponent<Image>();

        score = 0;
        textBox.text = maxTime.ToString();
        currentTime = maxTime;
    }

    // Update is called once per frame
    void Update()
    {
        textBox.text = Mathf.Round(maxTime).ToString();
        textBox.text = (int)currentTime + "초₩nSCORE : " + score;

        if (currentTime > 0)
        {

            currentTime -= Time.deltaTime;
            timeBar.fillAmount = currentTime / maxTime;


        }
        else
        {
            gameOverText.SetActive(true);
            Time.timeScale = 0;
        }
```

플레이어 스크립트는 이미 prefab에 작성되어 있어서 따로 작성하지 않았다. DragonMove와 Timer가 이 프로젝트의 핵심코드이다. 'DragonMove 스크립트'는 드래곤의 총을 맞는 횟수를 public으로 설정해서 inspector 창에서 설정 가능 하도록 만들었고 tag를 건 'Bullet'에 맞으면 사라지도록 만들었다. 총을 맞았을 때 'Bullet' tag를 가진 드래곤이 자기 자신을 파괴하도록 제작하기 위해서 OnCollisionEnter를 사용했다. 또한, 게임에 스릴을 주고 싶어서 Navmeshagent 컴포넌트와 SetDestination를 사용해주었다. 먼저 내비게이션 메시 관련된 기능을 사용하기 위해서는 "using UnityEngine.AI;"을 선언해주어야 한다. 인스펙터 창 옆에 Navigation 창을 만들고 Bake를 하면 Navmeshagent를 사용할 수 있다. 게임을 시작하면 드래곤에 부착된 Navmeshagent 컴포넌트를 찾아서 할당하도록 했다. Navmeshagent는 드래곤과 player(목표)까지 최단 거리를 계산해 추정해주고 장애물의 최대 속도를 정할 수 있다. 난이도 조절을 위해 각각의 장애물마다 다른 속도를 정해주었다. Agent는 내비게이션 메시 위에서 길을 찾아서 움직일 오브젝트(드래곤)를 의미한다. SetDestination는 player(목표지점)까지 움직이게 하는 함수로 Navmeshagent랑 함께 사용해야 한다.

Timer 스크립트는 14주 차에 배운 내용을 기반으로 작성했다. Timer 스크립트에 점수판 기능도 같이 작성했다. 수업 시간에 배우지 않았던 SerializeField는 private 변수를 인스펙터 창에서 접근할 수 있게 해주는 기능으로 원하는 변수 앞에 붙이는 것이다. Serialize는 직렬화라는 작업을 하는 것이다. 직렬화는 추상적 데이터를 전송할 수 있고 저장 가능한 형태로 바꾸는 것이다. Public은 인스펙터 창에서 접근할 수 있고 다른 스크립트에서 접근할 수 있다. 유니티에서 public 데이터만 직렬화하는데 Serialize를 선언함으로써 private도 직렬화할 수 있다. SerializeField로 gameOverText를 선언해주었고 인스펙터창에서 'TimeOver' UI 텍스트와 연결해 주었다. 최대 시간을 명시해주는 변수 maxtime을 설정하고 currentTime으로 남은 시간을 명시해주었다. Public으로 textbox 변수를 선언해서 inspector창의 UI 'Timer' 텍스트와 연결해주었다. 게임이 시작될 때 현재 시간과 최대시간이 같도록 작성해주었고 만약 현재 시간이 0보다 클 때 deltaTime으로 현재 시각을 매 1초씩 줄여주었다. 시간이 0일 때 gameOver Text를 true로 작성해 게임이 멈추도록 만들었다.

점수에 할당되는 변수 score를 public으로 선언하고 score 값은 0으로 초기화했다. UI의 timer 바로 밑에 score 값도 같이 나오도록 데이터를 변경해주었다. 'DragonMove' 스크립트에서 Score는 player가 드래곤을 맞출 때마다 10점을 얻고 총에 맞는 순간 드래곤이 Destroy 되도록 작성해주었다.

✓ 자체평가

이 게임은 간단하고 쉽게 만들었기 때문에 모든 연령이 할 수 있다. 간단한 총 게임으로 스트레스도 풀고 무서운 게임이 아니므로 부담 없이 즐길 수 있다. 이 프로젝트는 4가지이 한계점을 가지고 있다. 첫 번째, timer보다 빠르게 드래곤을 모두 처리했다면 남은 시간에 할 수 있는 것이 없다. 두 번째, 드래곤의 움직임이 자연스럽지 않다. 드래곤이 지정된 위치에서 플레이어로 다가올 때 나무, 바위 등 장애물의 collider 때문에 앞으로 나아가지 못하는 경우가 발생한다. 세 번째, 드래곤이 12

마리로 한정되어 있고 위치도 일정해서 게임을 반복하면 드래곤의 위치가 익숙해져 게임이 지루하게 느껴질 수 있다. 마지막으로, '시작, 게임 멈추기, 나가기'와 같은 메뉴가 없기 때문에 바로 게임이 시작되고 무조건 시간이 0이 돼야만 게임에서 나갈 수 있다. 이를 해결하기 위해서는 드래곤을 여러 마리 랜덤으로 생성시키도록 스크립트를 수정하고 움직임이 자연스럽도록 보완해야 한다. 또한, 'Start 버튼, 옵션설정, 레벨'을 선택할 수 있는 메뉴를 만들어 보완해야 한다.

더 좋은 퀄리티의 게임을 만들기 위해서 프로젝트에 사운드를 추가해야 하고 싶다. 프로젝트의 사운드가 없어 게임을 할 때 긴장감이 느껴지지 않았다. 시간이 10초 정도 남았을 때 사이렌 같은 긴장감 있는 사운드를 삽입해준다면 게임의 완성도가 높아질 것이라고 예상한다.

✓ 목표 대비 완성도

위 자체평가를 통한 나의 프로젝트의 목표 대비 완성도는 91%이라고 생각한다. 9%가 낮게 나온 이유는 기존에 목표한 것과 다르게 변경한 부분이 있기 때문이다. 점수로 순위를 정하는 게임으로 기획했지만 프로젝트를 제작할 때 시간제한 게임으로 변경하였다. 또한, 드래곤의 색이 아닌 움직임에 따라 총을 맞는 횟수를 다르게 하는 것이 난이도를 조절하는데 좋을 것 같아 변경해주었다. 더 좋은 방향으로 변경해주면서 점차 게임의 완성도가 높아졌지만 기존에 목표한 Start 버튼과 player를 직접 만들지 못한 것이 이 게임의 완성도를 떨어트린 것 같다.

퀴 즈

퀴즈01. 수업시간에 사용했던 “GvrEditorEmulator”의 기능을 설명하시오.

답안: 헤드 트래킹 시뮬레이션을 하는 기능이다. Alt+마우스 왼쪽 드래그는 좌우로 시야변경이 가능하고 Ctrl+마우스 왼쪽 드래그는 고개를 좌우로 회전하는 기능이다.

퀴즈02. Hierachy창의 UI메뉴의 Image, Text 등을 생성하면 Hierachy창에 자동으로 2개의 gameobject가 더 생성된다. 2개의 gameobject는 무엇인가.

답안: Canvas, Text

퀴즈03. 360도 이미지를 배경으로 선택하기 위해서는 OOOO을 만들고 인스 펙터 창에서 설정을 바꿔줘야한다. OOOO은 무엇이고, 설정은 어떻게 바꿔야 하는가.

답안: 매터리얼 / 인스펙터창에서 파노라마 이미지를 cube로 바꿔주고 skybox/cubemap 셰이더가 선택된 매터리얼에 넣어야 한다.

퀴즈04. 다음 세 개의 함수에 대해서 설명하시오. [OnTriggerEnter (Collider col), OnTriggerStay (Collider col), OnTriggerExit (Collider col)]

답안: OnTriggerEnter (Collider col) – 충돌한 순간에 호출, OnTriggerStay (Collider col)- 충돌하고 있는 동안 호출, OnTriggerExit (Collider col)- 충돌 후 분리되는 순간에 호출하는 함수이다.

퀴즈05. ARFoundation에 대해서 설명하고, Vuforia와의 차이점을 말하시오.

답안: ARFoundation은 유니티에서 AR 기능을 사용할 수 있게 한다. 마커인식은 asset에서 DB이미지를 직접 만들고 직접 유니티에 등록해야 한다. Vuforia는 마커방식 위주의 AR로 마커이미지를 등록해야한다. Vuforia는 license가 필요하며 뷰포리아 홈페이지에서 만들고 import해주어야 한다.

퀴즈06. UI에 관련된 코딩을 하기 위해서는 다음의 패키지를 추가해야 한다. 하늘색 빈칸에 들어갈 내용을 완성하시오.

답안: using UnityEngine.UI;

퀴즈07. Player이동시 AddForce()를 이용한 이동방법과 velocity를 이용한 이 동 방법 간의 차이를 설명하시오.

답안: AddForce()는 같은 힘을 가하면 속도가 점점 빨라지는 가속화하는 것이고 Velocity는 같은 힘을 가해도 동일한 속도로 이동할 수 있도록 하는 것이다.