클라이언트 개발자 포트폴리오

최윤호

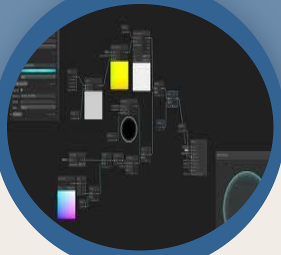
활용가능한툴및언어

유니티

- 여러 기능들을 알고 활용이 가능(Chinemachin, NavMesh등)
- 3D, 2D 프로젝트 제작 경험

C#,C++라이브러리

- 자료구조 및 알고리즘 지식
- C#으로 FrameWork 활용 경험



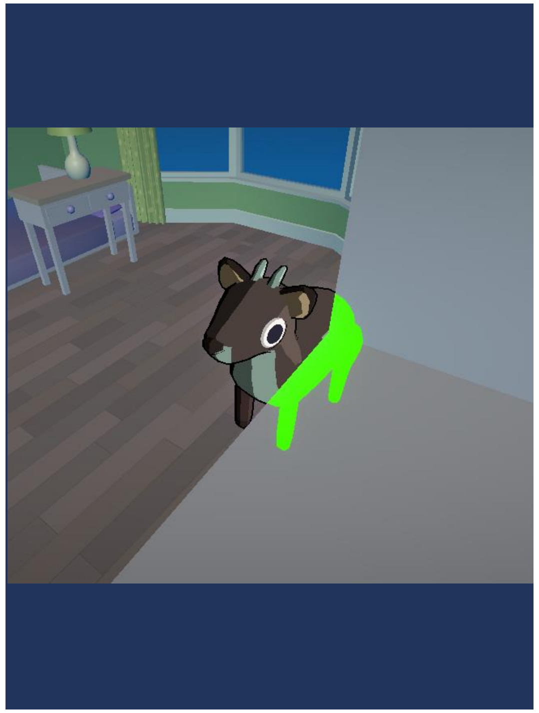
셰이더 그래프

- 다양한 노드로 Mesh 제작 경험

목차

1.1 만든 셰이더 그래프 소개

001



OffScene상태 시 투영

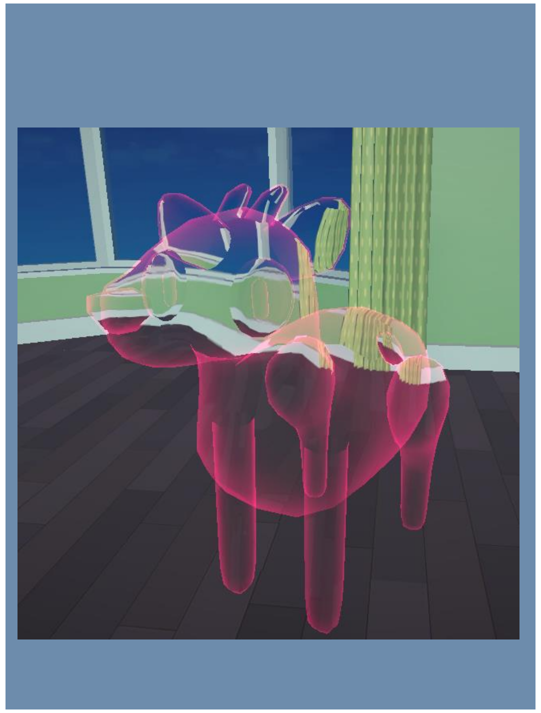
- 가려진 부분을 실루엣으로 표현

002

용암 베리어

- Time을 써서 움직이게 표현
- OutLine을 이용해 툰 처럼 표현

003



투명

- 왜곡과 투명하게 표현
- OutLine을 이용해 실루엣 표현

004

ToonShader

- 거리에 따라 OutLine의 크기 조절
- Gradient를 사용해 명암 표현

1.2 만든 셰이더 그래프 설명영상

설명영상

https://drive.google.com/file/d/12x8TZv3aCx1BDgA-fWzFV982HAplth4a/view?usp=sharing

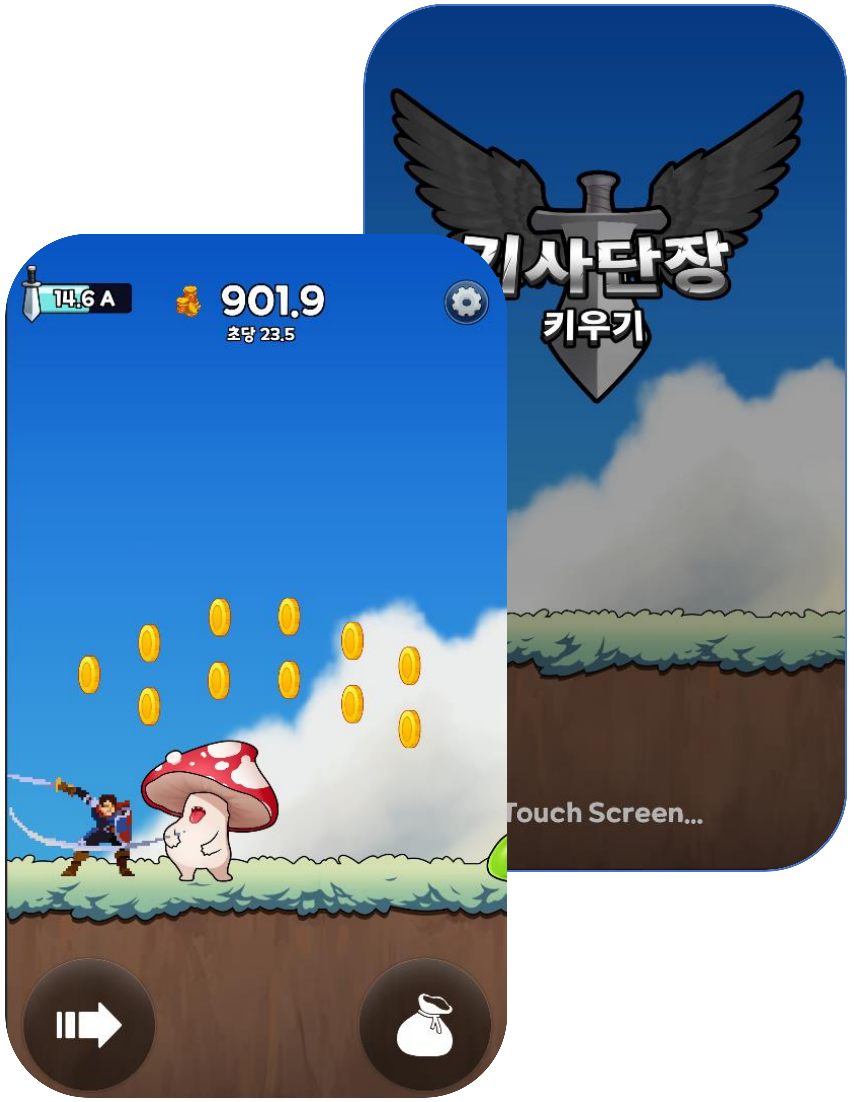
2.1 게임소개

기사단장키우기

- 게임장르 : 방치형 RPG
- 플랫폼 : Android
- 개발기간 : 3개월

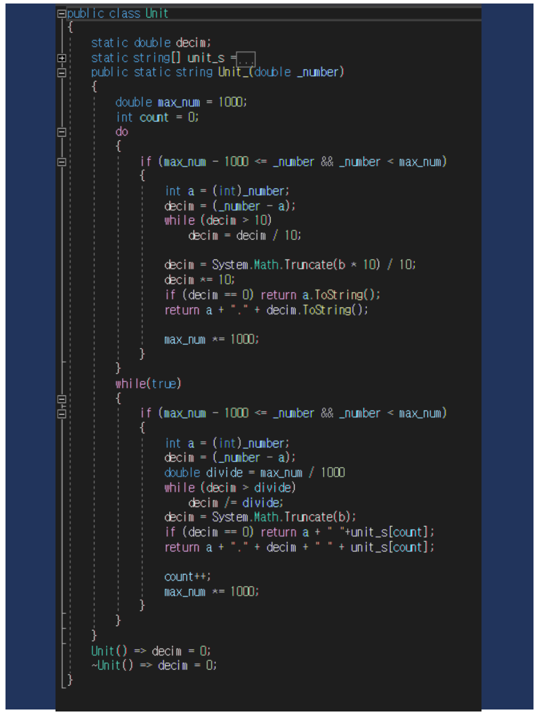
디자인 : 1명
프로그래밍 : 1명

졸업작품



2.2 스크립트소개

001



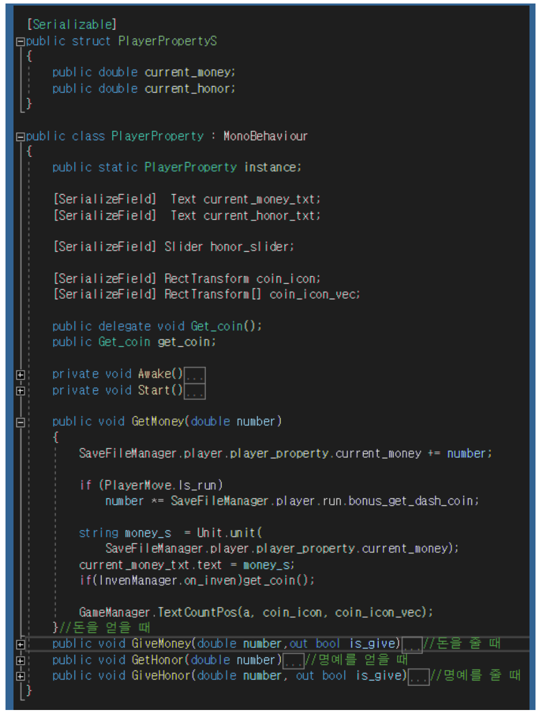
```
public class Unit
{
    static double decim;
    static string[] unit_s = ...
    public static string Unit_(double _number)
    {
        double max_num = 1000;
        int count = 0;
        do
        {
            if (max_num - 1000 <= _number && _number < max_num)
            {
                int a = (int)_number;
                decim = (_number - a);
                while (decim > 10)
                    decim = decim / 10;

                decim = System.Math.Truncate(b * 10) / 10;
                decim *= 10;
                if (decim == 0) return a.ToString();
                return a + "." + decim.ToString();

                max_num *= 1000;
            }
        }
        while(true)
        {
            if (max_num - 1000 <= _number && _number < max_num)
            {
                int a = (int)_number;
                decim = (_number - a);
                double divide = max_num / 1000
                while (decim > divide)
                    decim /= divide;
                decim = System.Math.Truncate(b);
                if (decim == 0) return a + " "+unit_s[count];
                return a + "." + decim + " " + unit_s[count];

                count++;
                max_num *= 1000;
            }
        }
    }
    Unit() => decim = 0;
    ~Unit() => decim = 0;
}
```

단위 변환

- 천 이상일 때 A, 백만 일 때 B

(6431 = 6.4A)

002

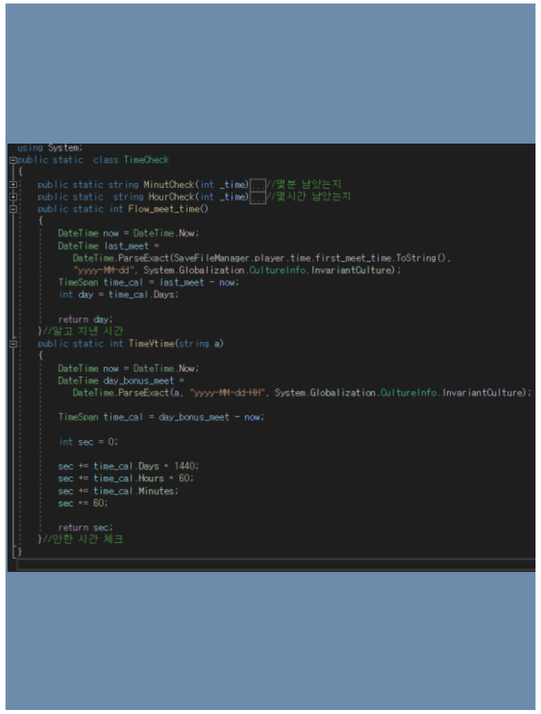


```
[Serializable]
public struct PlayerPropertyS
{
    public double current_money;
    public double current_honor;
}

public class PlayerProperty : MonoBehaviour
{
    public static PlayerProperty instance;

    [SerializeField]  Text current_money_txt;
    [SerializeField]  Text current_honor_txt;

    [SerializeField] Slider honor_slider;

    [SerializeField] RectTransform coin_icon;
    [SerializeField] RectTransform[] coin_icon_vec;

    public delegate void Get_coin();
    public Get_coin get_coin;

    private void Awake()...
    private void Start()...

    public void GetMoney(double number)
    {
        SaveFileManager.player.player_property.current_money += number;

        if (PlayerMove.Is_run)
            number *= SaveFileManager.player.run.bonus_get_dash_coin;

        string money_s  = Unit.unit(
            SaveFileManager.player.player_property.current_money);
        current_money_txt.text = money_s;
        if(InvenManager.on_inven)get_coin();

        GameManager.TextCountPos(a, coin_icon, coin_icon_vec);
    }//돈을 얻을 때
    public void GiveMoney(double number,out bool is_give)...//돈을 줄 때
    public void GetHonor(double number)...//명예를 얻을 때
    public void GiveHonor(double number, out bool is_give)...//명예를 줄 때
}
```

프로퍼티

- 플레이어 재화 및 명예 관련
- 재화 소비 및 습득 관리

003

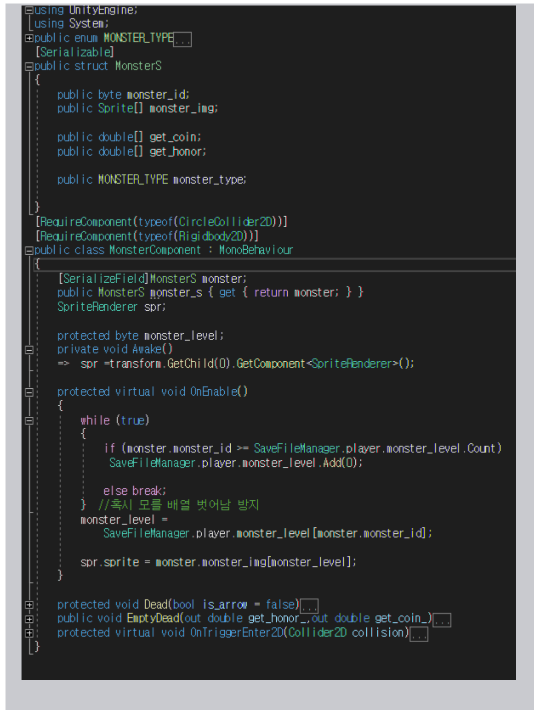


```
using System;
public static  class TimeCheck
{
    public static string MinutCheck(int _time)...//몇분 남았는지
    public static  string HourCheck(int _time)...//몇시간 남았는지
    public static int Flow_meet_time()
    {
        DateTime now = DateTime.Now;
        DateTime last_meet =
          DateTime.ParseExact(SaveFileManager.player.time.first_meet_time.ToString(),
          "yyyy-MM-dd", System.Globalization.CultureInfo.InvariantCulture);
        TimeSpan time_cal = last_meet - now;
        int day = time_cal.Days;

        return day;
    }//알고 지낸 시간
    public static int TimeVtime(string a)
    {
        DateTime now = DateTime.Now;
        DateTime day_bonus_meet =
          DateTime.ParseExact(a, "yyyy-MM-dd-HH", System.Globalization.CultureInfo.InvariantCulture);

        TimeSpan time_cal = day_bonus_meet - now;

        int sec = 0;

        sec += time_cal.Days * 1440;
        sec += time_cal.Hours * 60;
        sec += time_cal.Minutes;
        sec *= 60;

        return sec;
    }//안한 시간 체크
}
```

시간

- 비 접속시간 계산
- 플레이 시간 계산

004



```
using UnityEngine;
using System;
public enum MONSTER_TYPE...
[Serializable]
public struct MonsterS
{
    public byte monster_id;
    public Sprite[] monster_img;

    public double[] get_coin;
    public double[] get_honor;

    public MONSTER_TYPE monster_type;

}
[RequireComponent(typeof(CircleCollider2D))]
[RequireComponent(typeof(Rigidbody2D))]
public class MonsterComponent : MonoBehaviour
{
    [SerializeField]MonsterS monster;
    public MonsterS monster_s { get { return monster; } }
    SpriteRenderer spr;

    protected byte monster_level;
    private void Awake()
    =>  spr =transform.GetChild(0).GetComponent<SpriteRenderer>();

    protected virtual void OnEnable()
    {
        while (true)
        {
            if (monster.monster_id >= SaveFileManager.player.monster_level.Count)
             SaveFileManager.player.monster_level.Add(0);

            else break;
        }  //혹시 모를 배열 벗어남 방지
        monster_level =
            SaveFileManager.player.monster_level[monster.monster_id];

        spr.sprite = monster.monster_img[monster_level];
    }

    protected void Dead(bool is_arrow = false)...
    public void EmptyDead(out double get_honor_,out double get_coin_)...
    protected virtual void OnTriggerEnter2D(Collider2D collision)...
}
```

몬스터

- 몬스터 정보 죽음 및 관리

2.3 전체설명영상

설명영상

https://drive.google.com/file/d/1hcBAe8yDWwwniC8dP7Esy_d9O6Ei8gpq/view?usp=sharing

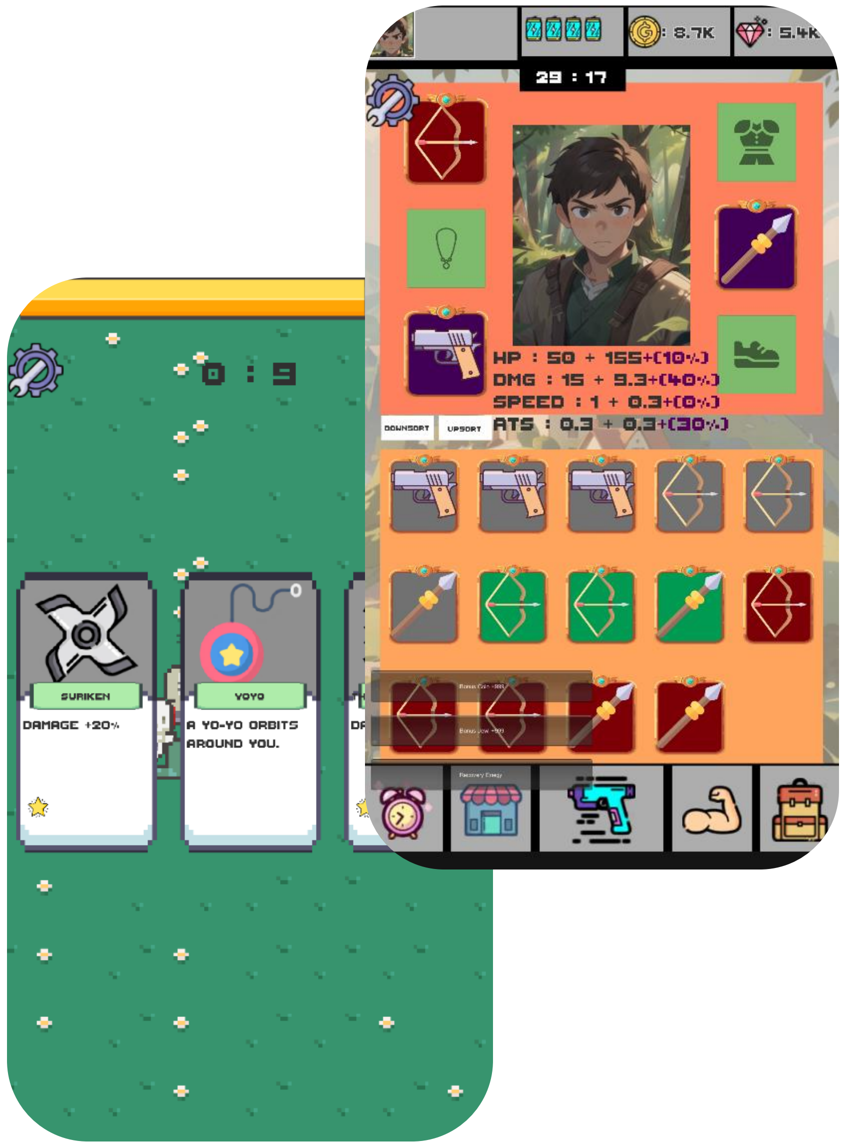
3.1 게임소개

빵야빵야친구들

- 참고 게임 : 탕탕 특공대
- 플 랫 폼 : Android
- 개발기간 : 1개월

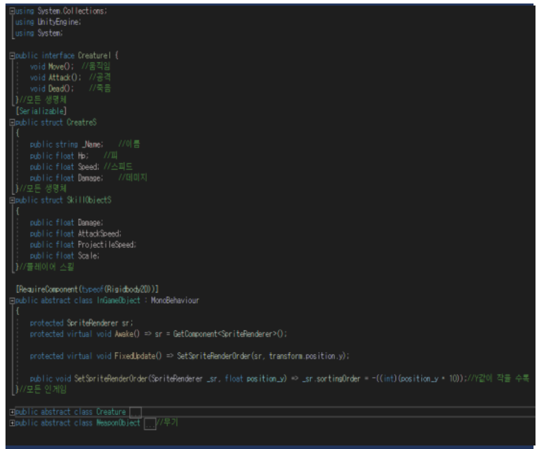
  프로그래밍 : 1명

개인작품



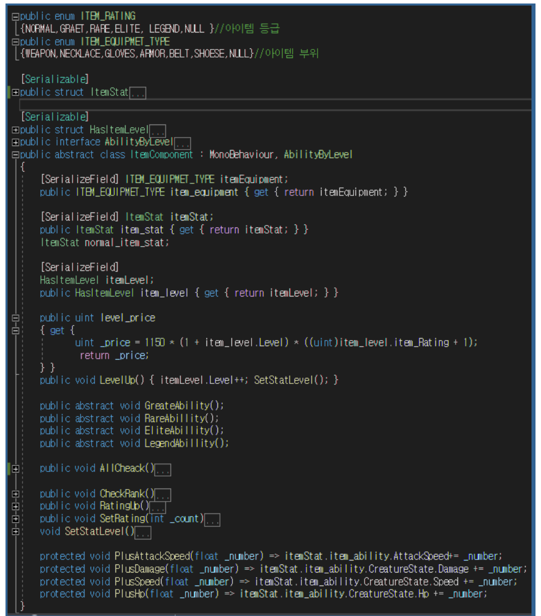
2.2 스크립트소개

001



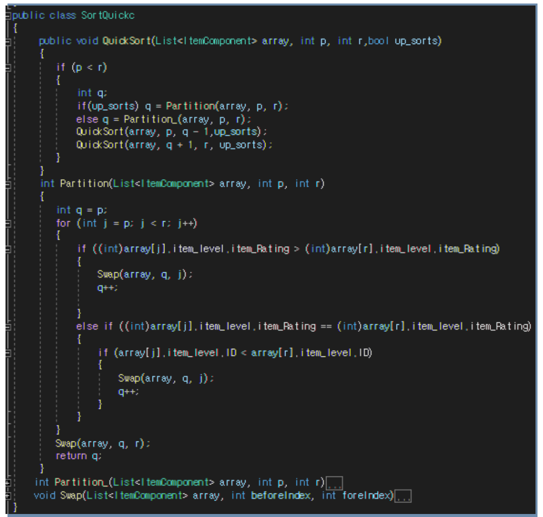
```
using System.Collections;
using UnityEngine;
using System;

public interface CreatureI {
    void Move();  //움직임
    void Attack();  //공격
    void Dead();    //죽음
}//모든 생명체
[Serializable]
public struct CreatreS
{
    public string _Name;    //이름
    public float Hp;    //피
    public float Speed; //스피드
    public float Damage;    //데미지
}//모든 생명체
public struct SkillObjectS
{
    public float Damage;
    public float AttackSpeed;
    public float ProjectileSpeed;
    public float Scale;
}//플레이어 스킬

[RequireComponent(typeof(Rigidbody2D))]
public abstract class InGameObject : MonoBehaviour
{
    protected SpriteRenderer sr;
    protected virtual void Awake() => sr = GetComponent<SpriteRenderer>();

    protected virtual void FixedUpdate() => SetSpriteRenderOrder(sr, transform.position.y);

    public void SetSpriteRenderOrder(SpriteRenderer _sr, float position_y) => _sr.sortingOrder = -((int)(position_y * 10));//Y값이 작을 수록
}//모든 인게임

public abstract class Creature ...
public abstract class WeaponObject ...//무기
```

002



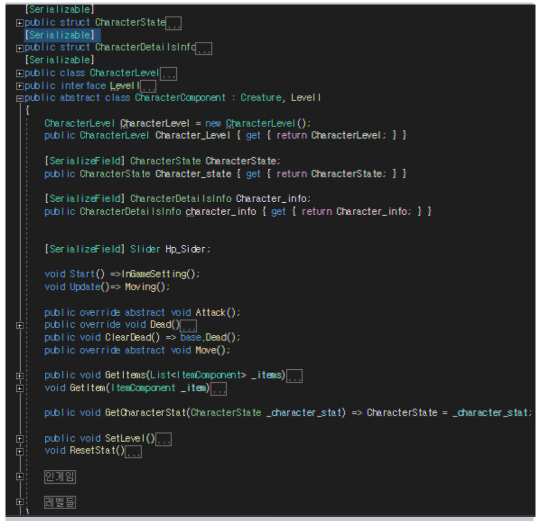
```
public enum ITEM_RATING
{NORMAL,GRAET,RARE,ELITE, LEGEND,NULL }//아이템 등급
public enum ITEM_EQUIPMET_TYPE
{WEAPON,NECKLACE,GLOVES,ARMOR,BELT,SHOESE,NULL}//아이템 부위

[Serializable]
public struct ItemStat...

[Serializable]
public struct HasItemLevel...
public interface AbilityByLevel...
public abstract class ItemComponent : MonoBehaviour, AbilityByLevel
{
    [SerializeField] ITEM_EQUIPMET_TYPE itemEquipment;
    public ITEM_EQUIPMET_TYPE item_equipment { get { return itemEquipment; } }

    [SerializeField] ItemStat itemStat;
    public ItemStat item_stat { get { return itemStat; } }
    ItemStat normal_item_stat;

    [SerializeField]
    HasItemLevel itemLevel;
    public HasItemLevel item_level { get { return itemLevel; } }

    public uint level_price
    { get {
            uint _price = 1150 * (1 + item_level.Level) * ((uint)item_level.item_Rating + 1);
             return _price;
    } }
    public void LevelUp() { itemLevel.Level++; SetStatLevel(); }

    public abstract void GreateAbility();
    public abstract void RareAbillity();
    public abstract void EliteAbillity();
    public abstract void LegendAbillity();

    public void AllCheack()...

    public void CheckRank()...
    public void RatingUp()...
    public void SetRating(int _count)...
    void SetStatLevel()...

    protected void PlusAttackSpeed(float _number) => itemStat.item_ability.AttackSpeed+= _number;
    protected void PlusDamage(float _number) => itemStat.item_ability.CreatureState.Damage += _number;
    protected void PlusSpeed(float _number) => itemStat.item_ability.CreatureState.Speed += _number;
    protected void PlusHp(float _number) => itemStat.item_ability.CreatureState.Hp += _number;
}
```

003



```
public class SortQuickc
{
    public void QuickSort(List<ItemComponent> array, int p, int r,bool up_sorts)
    {
        if (p < r)
        {
            int q;
            if(up_sorts) q = Partition(array, p, r);
            else q = Partition_(array, p, r);
            QuickSort(array, p, q - 1,up_sorts);
            QuickSort(array, q + 1, r, up_sorts);
        }
    }
    int Partition(List<ItemComponent> array, int p, int r)
    {
        int q = p;
        for (int j = p; j < r; j++)
        {
            if ((int)array[j].item_level.item_Rating > (int)array[r].item_level.item_Rating)
            {
                Swap(array, q, j);
                q++;

            }
            else if ((int)array[j].item_level.item_Rating == (int)array[r].item_level.item_Rating)
            {
                if (array[j].item_level.ID < array[r].item_level.ID)
                {
                    Swap(array, q, j);
                    q++;
                }
            }
        }
        Swap(array, q, r);
        return q;
    }
    int Partition_(List<ItemComponent> array, int p, int r)...
    void Swap(List<ItemComponent> array, int beforeIndex, int foreIndex)...
}
```

004



```
 [Serializable]
public struct CharacterState...
 [Serializable]
public struct CharacterDetailsInfo...
 [Serializable]
public class CharacterLevel...
public interface LevelI...
public abstract class CharacterComponent : Creature, LevelI
{
    CharacterLevel CharacterLevel = new CharacterLevel();
    public CharacterLevel Character_Level { get { return CharacterLevel; } }

    [SerializeField] CharacterState CharacterState;
    public CharacterState Character_state { get { return CharacterState; } }

    [SerializeField] CharacterDetailsInfo Character_info;
    public CharacterDetailsInfo character_info { get { return Character_info; } }


    [SerializeField] Slider Hp_Sider;

    void Start() =>InGameSetting();
    void Update()=> Moving();

    public override abstract void Attack();
    public override void Dead()...
    public void ClearDead() => base.Dead();
    public override abstract void Move();

    public void GetItems(List<ItemComponent> _items)...
    void GetItem(ItemComponent _item)...

    public void GetCharacterStat(CharacterState _character_stat) => CharacterState = _character_stat;

    public void SetLevel()...
    void ResetStat()...

    인게임

    레벨들
}
```

모든 생명체 및 오브젝트
- 생명체의 죽음, 공격 관리

장비
- 장비 정보(등급 및 스탯 등)
- 장비 강화

Quick Sort
- 인벤토리 등급 별로 정렬
- 인벤토리 아이템 별로 정렬

플레이어
- 플레이어어 정보
- 플레이어 인 게임 관련

2.3 전체설명영상

설명영상

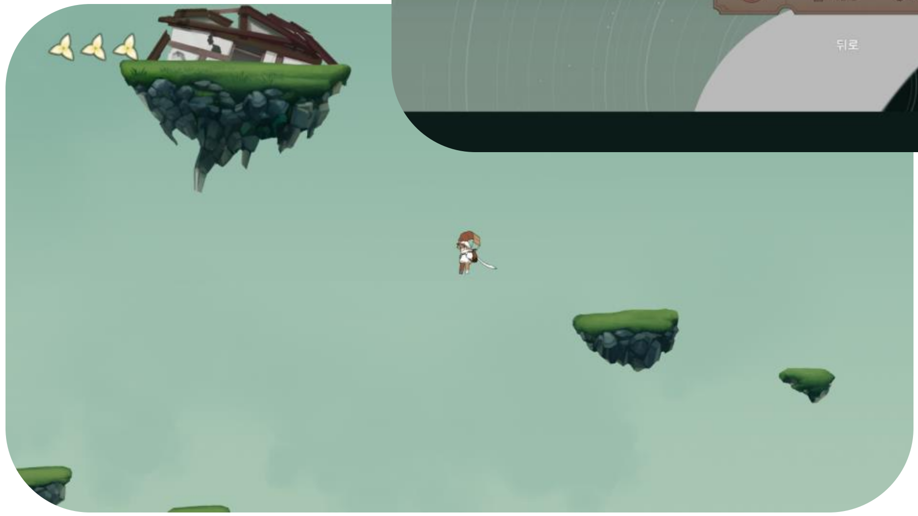
https://drive.google.com/file/d/1iuBCr0k_TR4sSVPb_GAfHBZnCQkLQqh_/view?usp=sharing

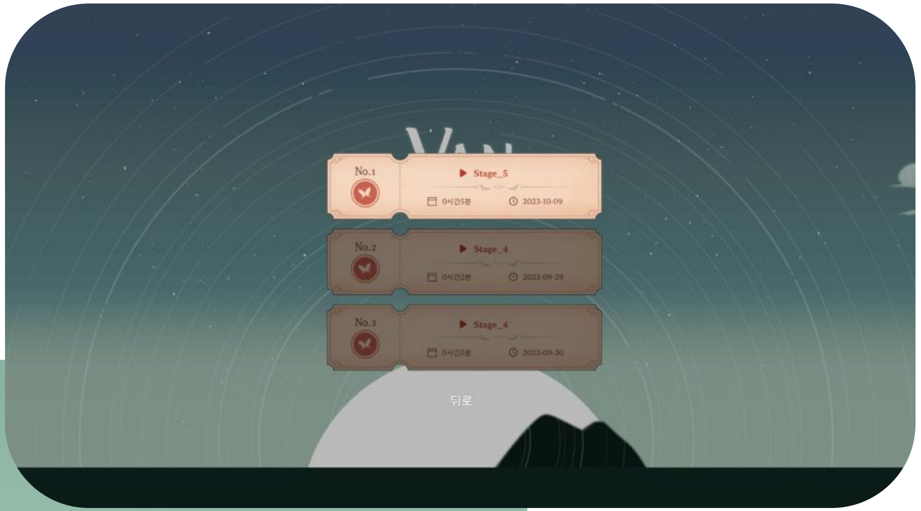
4.1 게임소개

Van

- 게임 장르 : 플랫포머
- 플랫폼 : PC
- 개발기간 : 3개월

프로그래밍 : 1명
기획 : 1명
디자인 : 4명

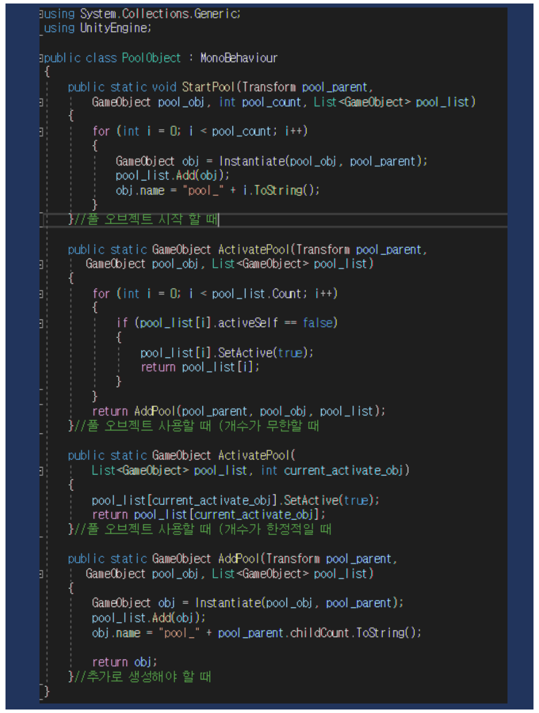




졸업 작품

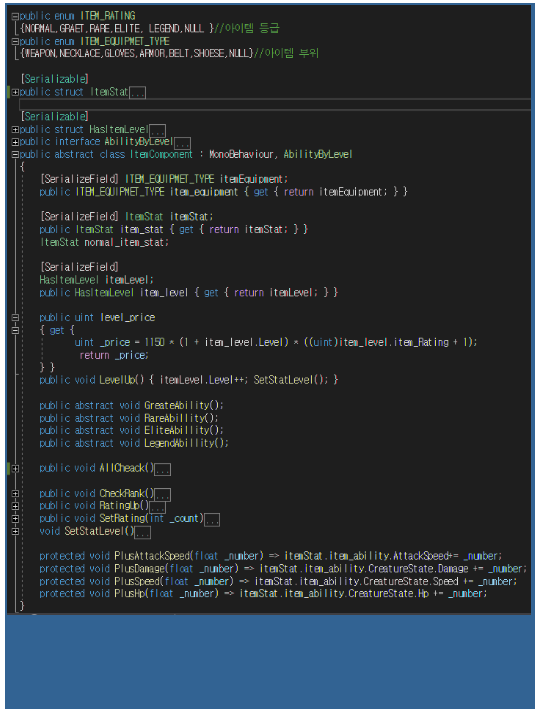
2.2 스크립트소개

001



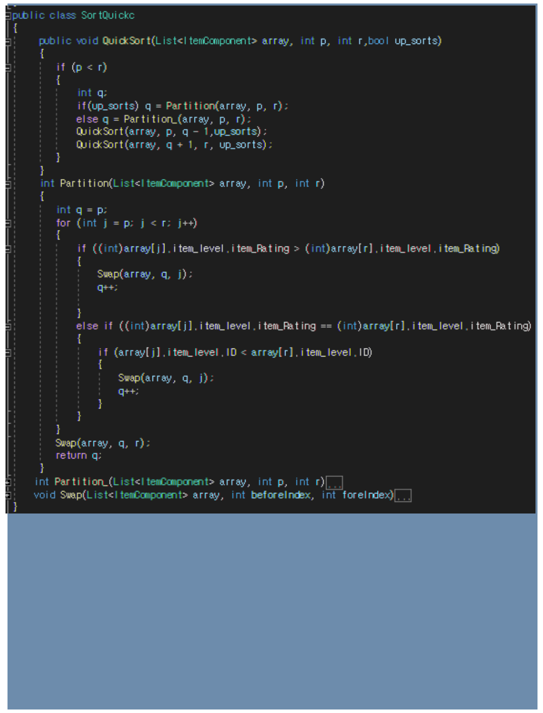
```
using System.Collections.Generic;
using UnityEngine;

public class PoolObject : MonoBehaviour
{
    public static void StartPool(Transform pool_parent,
        GameObject pool_obj, int pool_count, List<GameObject> pool_list)
    {
        for (int i = 0; i < pool_count; i++)
        {
            GameObject obj = Instantiate(pool_obj, pool_parent);
            pool_list.Add(obj);
            obj.name = "pool_" + i.ToString();
        }
    }//풀 오브젝트 시작 할 때

    public static GameObject ActivatePool(Transform pool_parent,
       GameObject pool_obj, List<GameObject> pool_list)
    {
        for (int i = 0; i < pool_list.Count; i++)
        {
            if (pool_list[i].activeSelf == false)
            {
                pool_list[i].SetActive(true);
                return pool_list[i];
            }
        }
        return AddPool(pool_parent, pool_obj, pool_list);
    }//풀 오브젝트 사용할 때 (개수가 무한할 때

    public static GameObject ActivatePool(
        List<GameObject> pool_list, int current_activate_obj)
    {
        pool_list[current_activate_obj].SetActive(true);
        return pool_list[current_activate_obj];
    }//풀 오브젝트 사용할 때 (개수가 한정적일 때

    public static GameObject AddPool(Transform pool_parent,
       GameObject pool_obj, List<GameObject> pool_list)
    {
        GameObject obj = Instantiate(pool_obj, pool_parent);
        pool_list.Add(obj);
        obj.name = "pool_" + pool_parent.childCount.ToString();

        return obj;
    }//추가로 생성해야 할 때
}
```

002



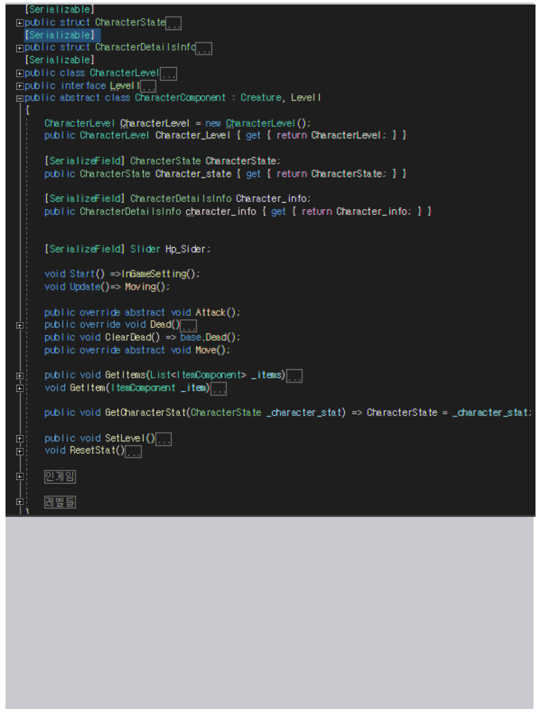
```
public enum ITEM_RATING
{NORMAL,GRAET,RARE,ELITE, LEGEND,NULL }//아이템 등급
public enum ITEM_EQUIPMET_TYPE
{WEAPON,NECKLACE,GLOVES,ARMOR,BELT,SHOESE,NULL}//아이템 부위

[Serializable]
public struct ItemStat...

[Serializable]
public struct HasItemLevel...
public interface AbilityByLevel...
public abstract class ItemComponent : MonoBehaviour, AbilityByLevel
{
    [SerializeField] ITEM_EQUIPMET_TYPE itemEquipment;
    public ITEM_EQUIPMET_TYPE item_equipment { get { return itemEquipment; } }

    [SerializeField] ItemStat itemStat;
    public ItemStat item_stat { get { return itemStat; } }
    ItemStat normal_item_stat;

    [SerializeField]
    HasItemLevel itemLevel;
    public HasItemLevel item_level { get { return itemLevel; } }

    public uint level_price
    { get {
            uint _price = 1150 * (1 + item_level.Level) * ((uint)item_level.item_Rating + 1);
             return _price;
    } }
    public void LevelUp() { itemLevel.Level++; SetStatLevel(); }

    public abstract void GreateAbility();
    public abstract void RareAbillity();
    public abstract void EliteAbillity();
    public abstract void LegendAbillity();

    public void AllCheack()...

    public void CheckRank()...
    public void RatingUp()...
    public void SetRating(int _count)...
    void SetStatLevel()...

    protected void PlusAttackSpeed(float _number) => itemStat.item_ability.AttackSpeed+= _number;
    protected void PlusDamage(float _number) => itemStat.item_ability.CreatureState.Damage += _number;
    protected void PlusSpeed(float _number) => itemStat.item_ability.CreatureState.Speed += _number;
    protected void PlusHp(float _number) => itemStat.item_ability.CreatureState.Hp += _number;
}
```

003



```
public class SortQuickc
{
    public void QuickSort(List<ItemComponent> array, int p, int r,bool up_sorts)
    {
        if (p < r)
        {
            int q;
            if(up_sorts) q = Partition(array, p, r);
            else q = Partition_(array, p, r);
            QuickSort(array, p, q - 1,up_sorts);
            QuickSort(array, q + 1, r, up_sorts);
        }
    }
    int Partition(List<ItemComponent> array, int p, int r)
    {
        int q = p;
        for (int j = p; j < r; j++)
        {
            if ((int)array[j].item_level.item_Rating > (int)array[r].item_level.item_Rating)
            {
                Swap(array, q, j);
                q++;

            }
            else if ((int)array[j].item_level.item_Rating == (int)array[r].item_level.item_Rating)
            {
                if (array[j].item_level.ID < array[r].item_level.ID)
                {
                    Swap(array, q, j);
                    q++;
                }
            }
        }
        Swap(array, q, r);
        return q;
    }
    int Partition_(List<ItemComponent> array, int p, int r)...
    void Swap(List<ItemComponent> array, int beforeIndex, int foreIndex)...
}
```

004



```
[Serializable]
public struct CharacterState...
[Serializable]
public struct CharacterDetailsInfo...
[Serializable]
public class CharacterLevel...
public interface LevelI...
public abstract class CharacterComponent : Creature, LevelI
{
    CharacterLevel CharacterLevel = new CharacterLevel();
    public CharacterLevel Character_Level { get { return CharacterLevel; } }

    [SerializeField] CharacterState CharacterState;
    public CharacterState Character_state { get { return CharacterState; } }

    [SerializeField] CharacterDetailsInfo Character_info;
    public CharacterDetailsInfo character_info { get { return Character_info; } }


    [SerializeField] Slider Hp_Sider;

    void Start() =>InGameSetting();
    void Update()=> Moving();

    public override abstract void Attack();
    public override void Dead()...
    public void ClearDead() => base.Dead();
    public override abstract void Move();

    public void GetItems(List<ItemComponent> _items)...
    void GetItem(ItemComponent _item)...

    public void GetCharacterStat(CharacterState _character_stat) => CharacterState = _character_stat;

    public void SetLevel()...
    void ResetStat()...

    인게임

    레벨들
}
```

Pool Object

- 생명체의 죽음, 공격 관리

장비

- 장비 정보(등급 및 스탯 등)
- 장비 강화

퀵 소트

- 인벤토리 등급 별로 정렬
- 인벤토리 아이템 별로 정렬

플레이어

- 플레이어어 정보
- 플레이어 인 게임 관련

2.3 전체설명영상

설명영상

https://drive.google.com/file/d/1TZTPRABmQBwgP-DI96sL537LpbHcOShg/view?usp=sharing

감사합니다.

전화번호 : 010-5293-6088
이메일 : yoyoyoyo0014@naver.com