# Lost Bible

1788007 게임공학과 김성륜

# 목차



Part 1,

# 게임 소개

Part 1,

# 게임 소개

| | |
|---|---|
| 제목 | Lost Bible |
| 장르 | 전략 시뮬레이션 (RTS) |
| 플랫폼 | PC / Windows |
| 개발도구 | Unity 3D |

Part 1,

# 게임 특징

## 1. 심플한 조작 - 편리성

마우스만으로도 할수있어 누구나 쉽게 입문할 수 있는 게임임.

## 2. 실시간 전략으로 게임의 승패가 결정 - 전략성

전략으로 승패가 결정되며, 힘든 상황에서도 위기를 모면하여 역전의 기회를 가질수가 있음.

Part 2,

# 게임 시스템

Part 2,

# 게임 규칙

1. 도전 과제로는 유닛을 적절하게 배치하여 상대 유닛을 격파하는 것이 목표

2. 승리 조건으로는 적 유닛을 30기 격파하는 것이고, 패배 조건은 아군 베이스가 파괴되는 것임.

Part 2,

# 주요 오브젝트(맵)

Part 2,

# 주요 오브젝트(유닛)

## 1. 아군 유닛



## 2. 아군 베이스

Part 2,

# 전투 시스템

1. 유닛을 소환한뒤 드래그를하여 유닛을 지정하고, 화면 중하단에 있는 이동과 대기, 공격을 명령하여 상대 유닛의 공격을 수비하고 전진한다.

2. 다양한 아군 유닛으로 기지 방어,공격, 회피를 할수있다. 전략적으로 게임을 읽고, 승리를 가져오게 할 수 있다.

3. 유닛을 소환할 때, 원하는 포지션에 소환이 가능하며(능선에서는 소환불가) 일정 자원이 차감한다.

Part 3

# 주요 스크립트 설정

Part 3,

# 주요 스크립트

## 저장관련 스크립트

```
참조 1개
public void OpenMenu()
{
    Time.timeScale = 0;
    SaveMenu.SetActive(false);
    LoadMenu.SetActive(false);
    PauseMenu.SetActive(true);
    Camera.main.GetComponent<CameraMovement>().CanMove = false;
}
```

**MenuController 스크립트를 작성하여 게임의 진행을 멈추고 Save창과 Load창을 불러오게 됩니다. 이때 카메라의 움직임도 멈추기때문에 전체 게임이 멈추도록 설정하였습니다.**

```
참조 0개
public void OnSaveGameClicked()
{
    SaveMenu.SetActive(true);
    LoadMenu.SetActive(false);
    PauseMenu.SetActive(false);
}

참조 0개
public void OnLoadGameClicked()
{
    LoadMenu.SetActive(true);
    SaveMenu.SetActive(false);
    PauseMenu.SetActive(false);
}
```

**옵션에 있는 Save와 Load를 불러올때 옵션창이 사라지며, 클릭한 창의 모습이 나옵니다.**

Part 3,

# 주요 스크립트

## 저장관련 스크립트

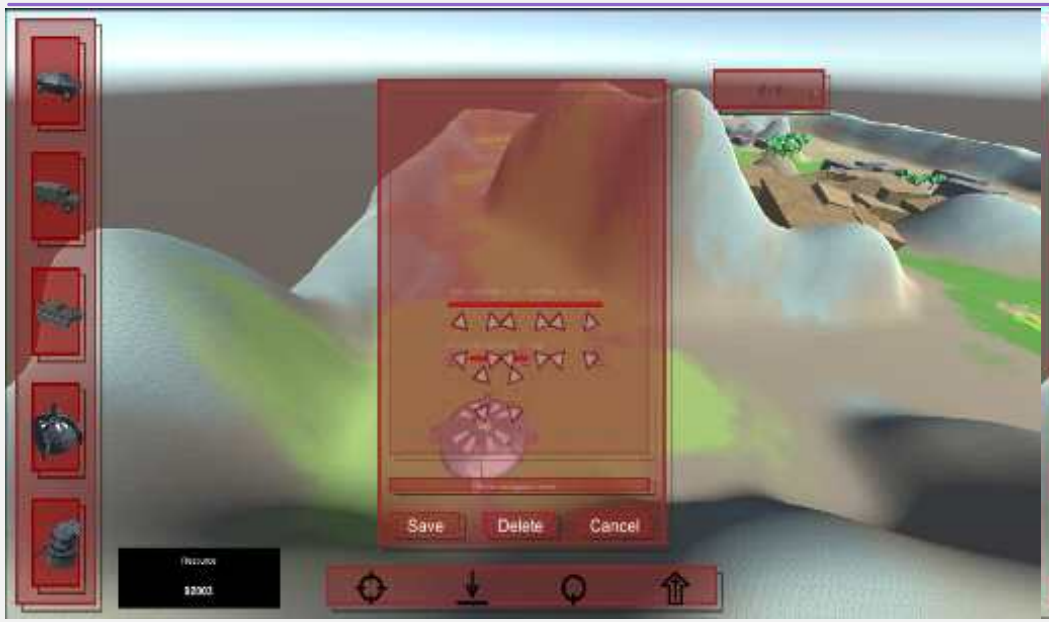
```
public class SaveMenuController : ListMenuController
{
    public InputField FilenameInput;
    public AICommander AIcommander;

    3
    참조 1개
    public override void OnConfirm()
    {
        string filename = FilenameInput.text == ""
            ? DateTime.Now.ToString("yyyy-dd-MM-HH-mm-ss")
            : FilenameInput.text;

        FileStream stream = new FileStream(SAVE_FOLDER + "/" + filename, FileMode.OpenOrCreate);
        BinaryFormatter formatter = new BinaryFormatter();
        formatter.Serialize(stream, GetSavegameData());
        stream.Close();

        Debug.Log("Game saved! - " + filename);
        OnEnable();

        OnCancel();
```

**Save로 게임을 저장할 시 저장공간내에 파일이 생기며, 파일이름은 연,월,일 순으로 기재되며 따로 세이브 파일이름을 지정할 수 있습니다. 저장이되면 옵션으로 돌아가며,Load로 들어가면 Save했던 게임파일이 있습니다. Load를 하면 해당 게임으로 돌아가며, 게임을 계속 이어갈 수 있습니다.**

Part 3,

# 주요 스크립트

**저장관련 스크립트**




Save 화면


Load 화면

Part 3,

# 주요 스크립트

## Enemy AI

```
void Start()
{
    _spawnLocations = MapData.Instance.SpawnAreas;
    StartCoroutine(SpawnUnit());
}

참조 1개
private IEnumerator SpawnUnit()
{
    while (true)
    {
        yield return new WaitForSeconds(Random.Range(MinSpawnInterval, MaxSpawnInterval));

        if (_spawnedUnits >= MaxUnits)
            continue;

        // Get randomized spawn position
        Vector3 spawnPos = GetSpawnPosition();

        // Raycast down to see spawn height (to place accurately on terrain)
        RaycastHit hit;
        if (Physics.Raycast(spawnPos, Vector3.down, out hit, Mathf.Infinity, 1 << LayerMask.NameToLayer("Terrain")))
        {
            spawnPos = hit.point + Vector3.up * 2;
        }

        GameObject unitToSpawn = UnitPrefabs[Random.Range(0, UnitPrefabs.Length)];
        UnitController spawnedUnit = Instantiate(unitToSpawn, spawnPos, Quaternion.identity, transform).GetComponent<UnitContro
        spawnedUnit.isPlayerUnit = false;
        spawnedUnit.tag = Constants.NONPLAYER_UNIT;
        spawnedUnit.GetComponentInChildren<MeshRenderer>().material.color = Color.blue;
        spawnedUnit.transform.SetParent(transform);
        EventManager.OnUnitSpawned(EventArgs.Empty, spawnedUnit.gameObject);

        _spawnedUnits++;
    }
}
```

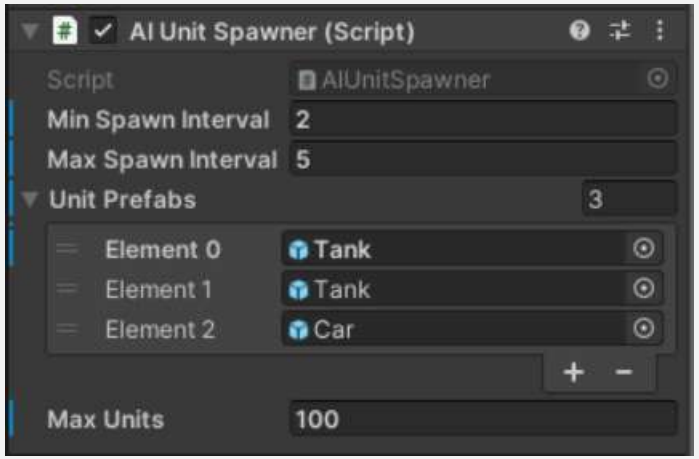
해당 스크립트는 AI유닛의 개체 수와 생성 최소시간, 최대시간을 관리하는 스크립트입니다. 스폰이 될 때 플레이어 유닛으로 간주하지않지만 유닛과 똑같은 사거리, 크기를 갖습니다.

Part 3,

# 주요 스크립트

## Enemy AI

```
void Start()
{
    _spawnLocations = MapData.Instance.SpawnAreas;
    StartCoroutine(SpawnUnit());
}

참조 1개
private IEnumerator SpawnUnit()
{
    while (true)
    {
        yield return new WaitForSeconds(Random.Range(MinSpawnInterval, MaxSpawnInterval));

        if (_spawnedUnits >= MaxUnits)
            continue;

        // Get randomized spawn position
        Vector3 spawnPos = GetSpawnPosition();

        // Raycast down to see spawn height (to place accurately on terrain)
        RaycastHit hit;
        if (Physics.Raycast(spawnPos, Vector3.down, out hit, Mathf.Infinity, 1 << LayerMask.NameToLayer("Terrain")))
        {
            spawnPos = hit.point + Vector3.up * 2;
        }

        GameObject unitToSpawn = UnitPrefabs[Random.Range(0, UnitPrefabs.Length)];
        UnitController spawnedUnit = Instantiate(unitToSpawn, spawnPos, Quaternion.identity, transform).GetComponent<UnitContro
        spawnedUnit.IsPlayerUnit = false;
        spawnedUnit.tag = Constants.NONPLAYER_UNIT;
        spawnedUnit.GetComponentInChildren<MeshRenderer>().material.color = Color.blue;
        spawnedUnit.transform.SetParent(transform);
        EventManager.OnUnitSpawned(EventArgs.Empty, spawnedUnit.gameObject);

        _spawnedUnits++;
    }
}
```

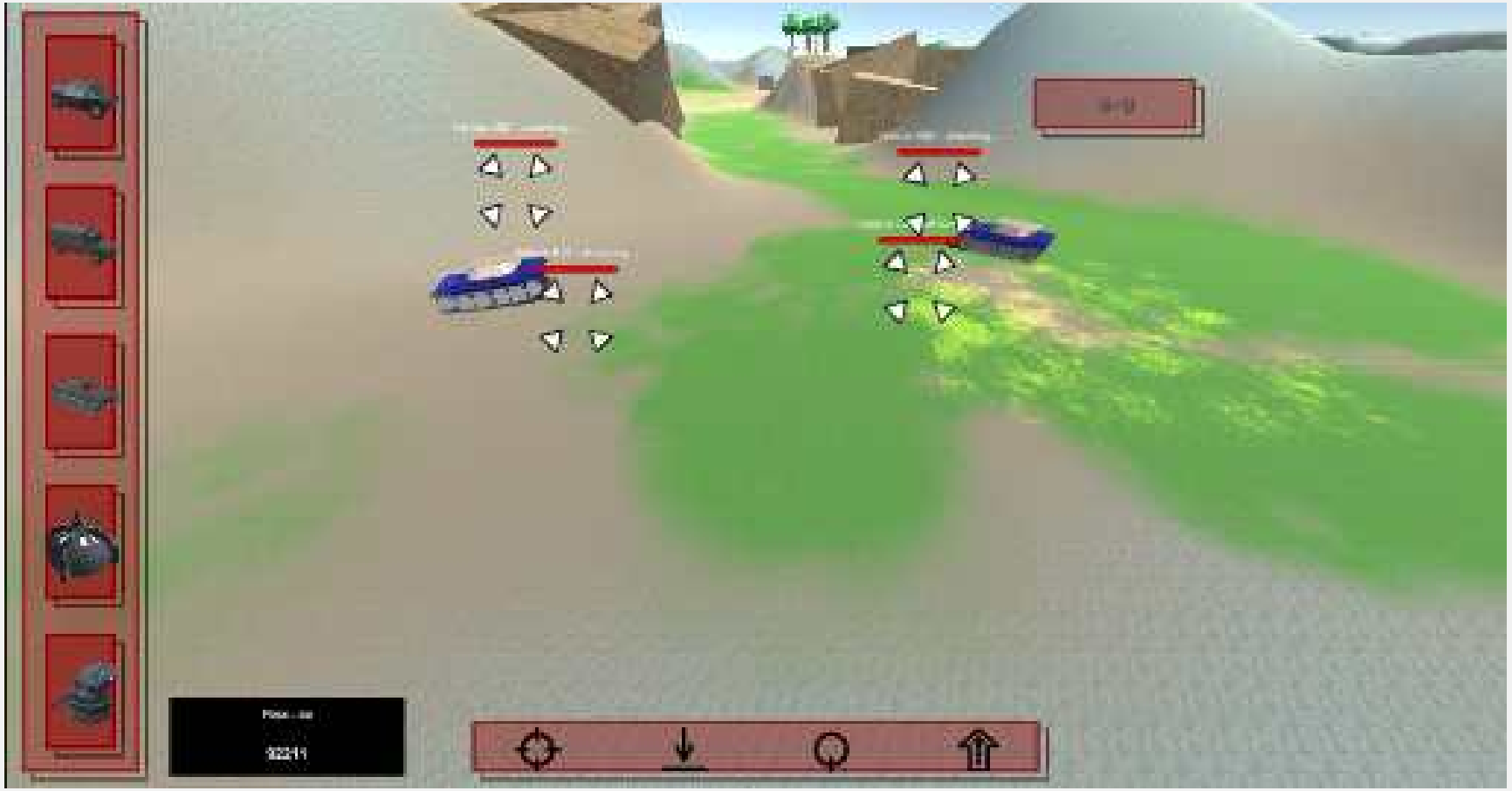



Unity 인스펙터창에서 해당 스크립트의 변수 값을 할당할 수 있습니다.

Part 3,

# 주요 스크립트

**Enemy AI**



Part 4,

# 영상 소개

Part 5,

# 추가 개발 계획

Part 5,

# 개발 추가 계획

1. 스코어 추가
2. 플레이 타임 시간 추가
3. 다양한 유닛 추가
4. 모델링을 추가하여 기지 추가
5. 승패 결과 기록 시스템, 유닛 추가후 유닛 종류 선택을 하여 전투 참여

Part 6,

# 소감 및 Q&A

Part 6,

# 소감 및 아쉬운점

## 아쉬운점

2022년 2학기부터 기획했던 전략 시뮬레이션 게임이지만, 약 4번의 실패를 경험하였고, 실패한 만큼 상대적 시간이 부족하여 다양한 컨텐츠 및 개인적으로 추가하고싶은 것을 넣지 못해 많이 아쉽습니다.

## 소감

부족한 퀄리티에 다양한 오류까지 생기면서 도저히 혼자 못하겠다 생각까지 들었을 때, 면담으로 지금까지 했던거 잘 다듬고 더 해보자라고 말씀하신 교수님, 버그가 나왔을 때 같이 봐주던 후배들, 구현이 어려울 때 없으신 시간 쪼개서 봐주신 지인분들 덕분에 실패하지 않고 완성할 수 있었던것 같습니다.


감사합니다
Q&A