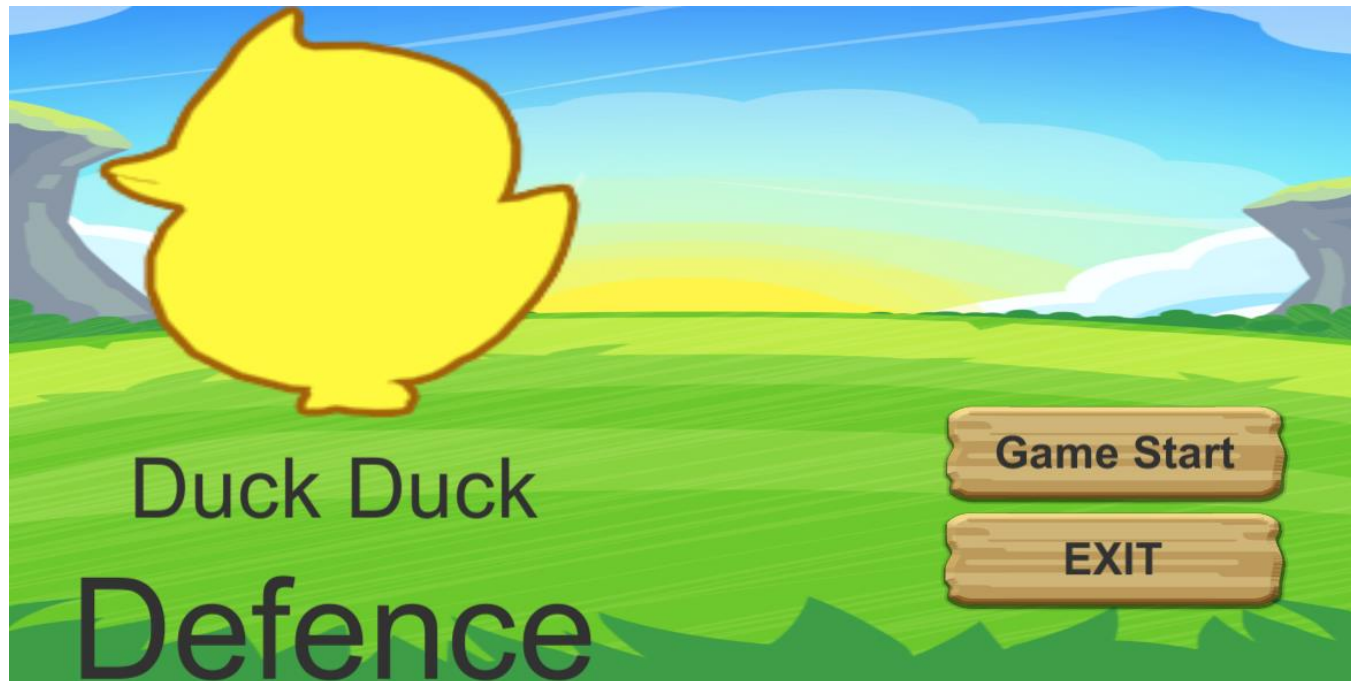


[포트폴리오]

DUCK DUCK 디펜스 기술문서



동영상 링크

<https://youtu.be/Ns0-2FCMxQY>

경일 게임 아카데미

게임 프로그래밍 16기

이 도 규

0. 목차

1. 게임 내용 및 설명

1-1 개발 환경

1-2 일정

1-3 내용

2. 구현 내역

1. 게임

2. 동적 인터페이스

3. 구현 편의성

3. 기술 코드

1. 공용 DB

2. 동적 인터페이스

3. 기타 코드

부록

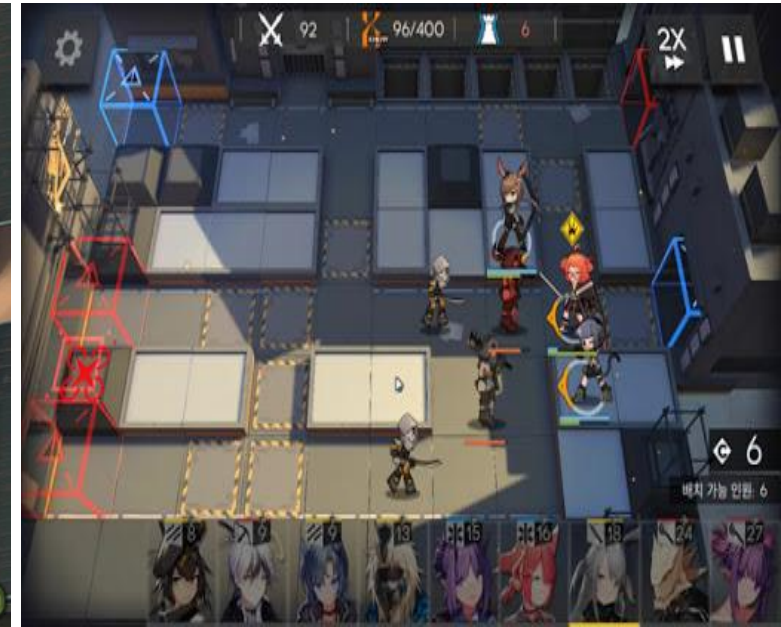
1. 게임 개요 및 설명 – 개발 환경

구분	환경
CPU	3세대 I5 3470 – 3.20GHZ
메모리	8GB
시스템	X64
하드	128GB
운영체제	WINDOW 10 PRO
해상도	1024 * 800
라이브러리	UNITY 2019.2.17f1

1. 게임 개요 및 설명 - 일정

1주차 일정	내용	2주차 일정	내용	3주차 일정	내용
월	게임 기획	월	오브젝트, 캐릭터 테이블 작성	월	적기 선정 및 작성
화	에셋 탐색	화	오브젝트 선정 및 캐릭터 테이블 작성	화	적기 테스트
수	기획 내역 조성 컨셉 기획	수	오브젝트 테스트	수	캐릭터 공격 조건 작성
목	필요 변수 및 데이터 테이블 작성	목	인터페이스 디버그	목	캐릭터 공격 테스트
금	타이틀 - 선택 - 게임 (3단계 구조 작성)	금	테스트 맵 작성 (마우스 배치)	금	타이틀 및 엔딩
토	3단계 호환 가능한 DB 클래스 작성	토	테스트 맵 작성 (아군 조건 테스트)	토	통상 모드 테스트
일	선택 화면을 이용한 플레이어 선택 작성	일	테스트 맵 작성 (적군 조건 테스트)	일	디버그

1. 게임 개요 및 설명 - 내용



건물 또는 유닛을 놓고 적들의 움직임을 막거나 처치하는 방어 게임

타일의 위치에 따라 속성이 변하거나, 건설 조건을 제어하는 등의 요소를 확인 할 수 있다.

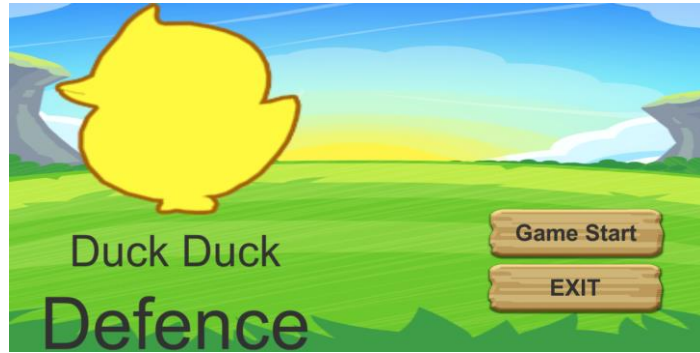
★ 데이터를 이용한 **캐릭터 오브젝트 관리**

1. 게임 개요 및 설명 - 내용

1. 플레이어 타워에 대한 정보 저장 테이블
2. 타일 구성 및 타일에 대한 조건 확인
3. 적기 등장 및 UI 등 전체 구성 및 mesh를 이용한 경로 탐색 적용
4. 타워를 놓아 적을 막는 시스템 테스트
5. 플레이어 타워 구성
6. UI 구성

단계	1단계	2단계	3단계	4단계	5단계	6단계	7단계
내용	맵 구성 및 맵 출력	적 구성 적의 등장 및 이동	플레이어 타워 구성 및 제작	맵과 오브젝트 간 상호작용	게임 승리 조건 및 패배 조건 적용	UI 및 게임 플레이	디버그

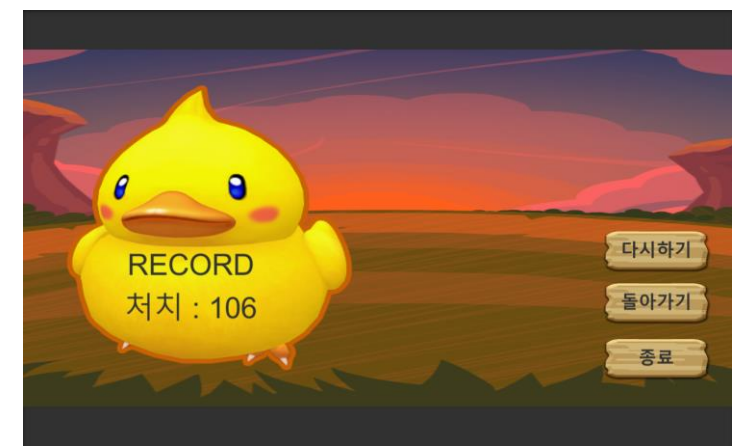
2. 구현 내역 - 게임



TITLE Scene



SELECT Scene

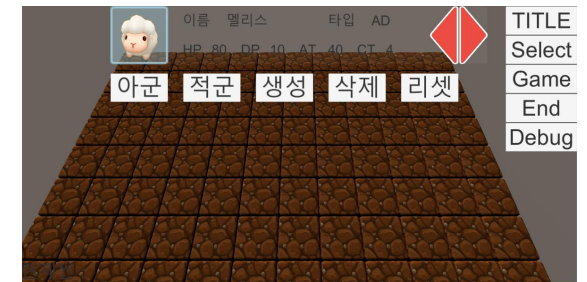


Ending Scene



Game Scene

Debug Scene



2. 구현 내역 - 게임

목숨

00

시간


0 : 39

적의 수

적 : 29

처치한 수

처치 : 17

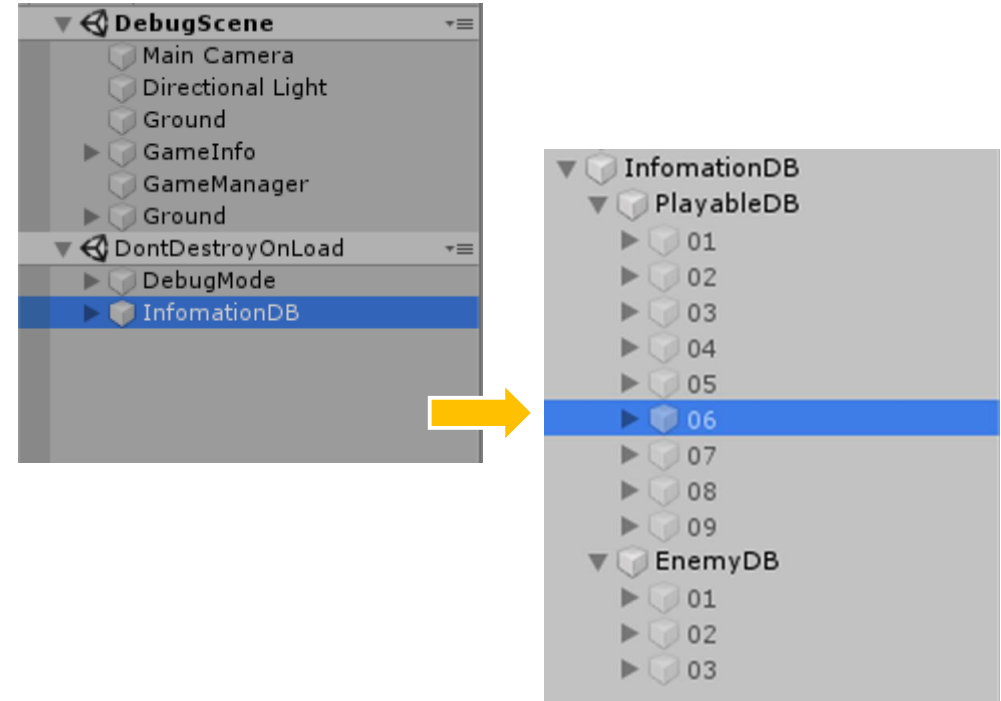
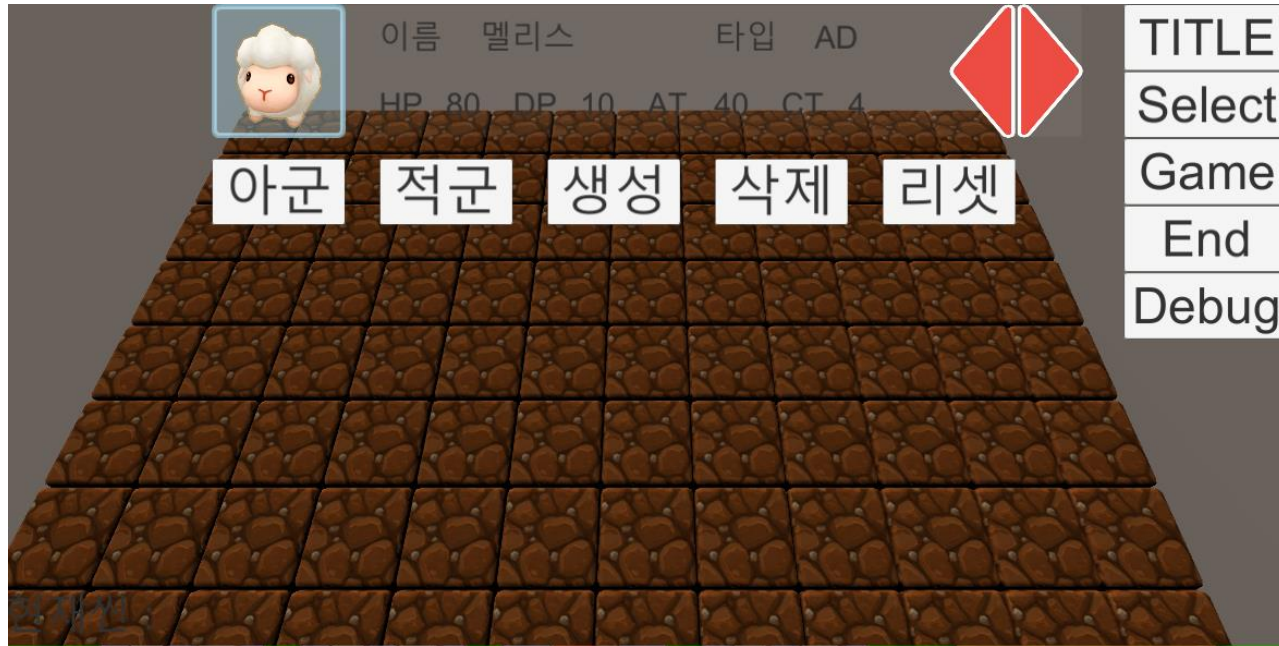


사용되는 플레이어블 캐릭터

삭제

코스트

2. 구현 내역 – InfomationDB

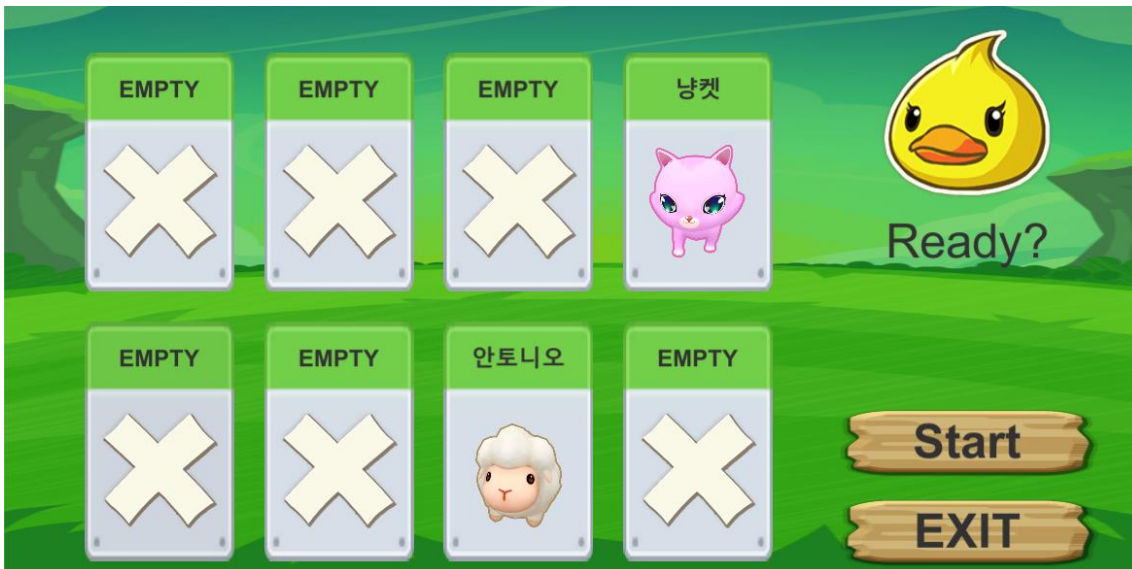


DEBUG 모드를 구현 캐릭터 오브젝트를 직접 구현하거나 설정할 수 있는 공간을 작성
InformationDB를 구성할 수 있다.

2. 구현 내역 – 동적 인터페이스



CSV 파일에 저장되어 있는
캐릭터 ID의 개수에 따라
캐릭터에 대한 정보가 출력된다.



DB에 저장된 캐릭터를
플레이어를 캐릭터로 작성

게임을 시작할 수 있다.

2. 구현 내역 – 구현 편의성

1. CSV 작성

몬스터에 대한 기본적인 내역을 작성한다

Playable

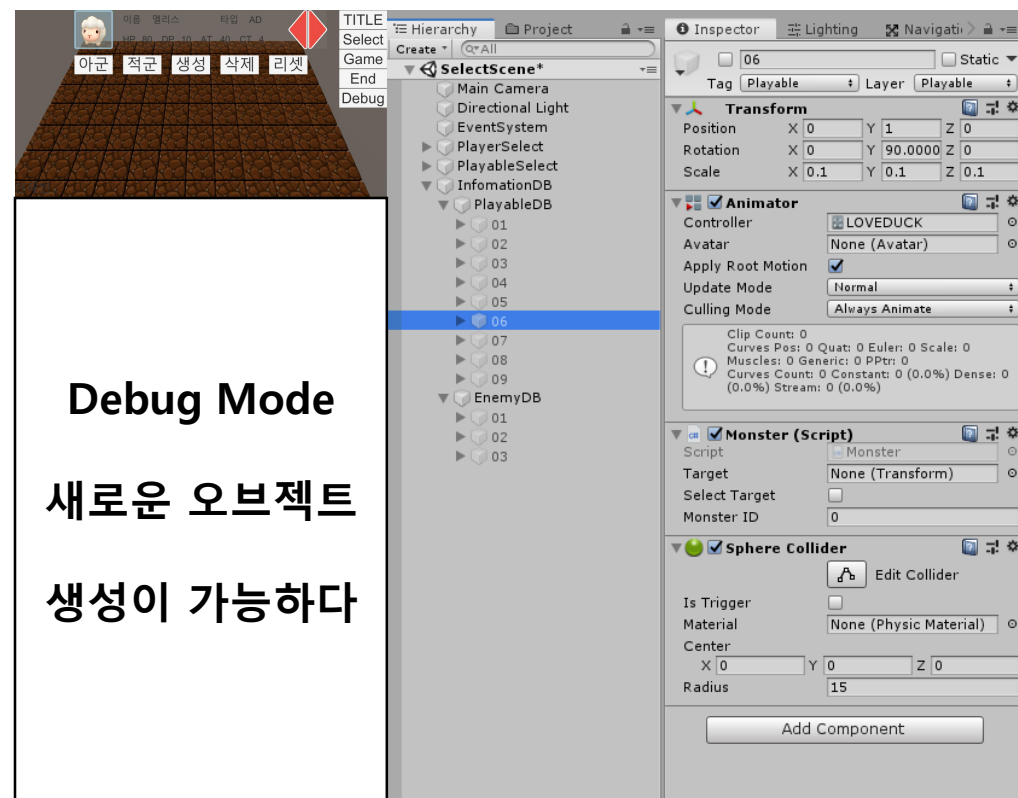
ID	TYPE	NAME	HP	ATTACK	DEFFENCE	COST	RANGE	DELAY	ImagePath	ImageLogoPath
0	EMPTY	EMPTY	0	0	0	0	0	0	empty	empty
1	AD	멜리스	80	40	10	4	1	1	01sheep	01sheep
2	AD	안토니오	100	50	2	5	1	1	01sheep	01sheep
3	AD	냥켓	80	10	12	1	2	1	08cat	08cat
4	AD	키라라	60	20	13	2	1	1	08cat	08cat
5	AD	팡	40	35	14	2	1	1	05peng	05peng
6	TD	오구리	20	80	15	10	3	2	09duck	09duck
7	TD	육구리	20	100	16	11	3	2	09duck	09duck
8	TD	칠구리	20	150	17	12	4	2	09duck	09duck
9	TD	팔구리	50	200	18	13	4	2	09duck	09duck

ENEMY

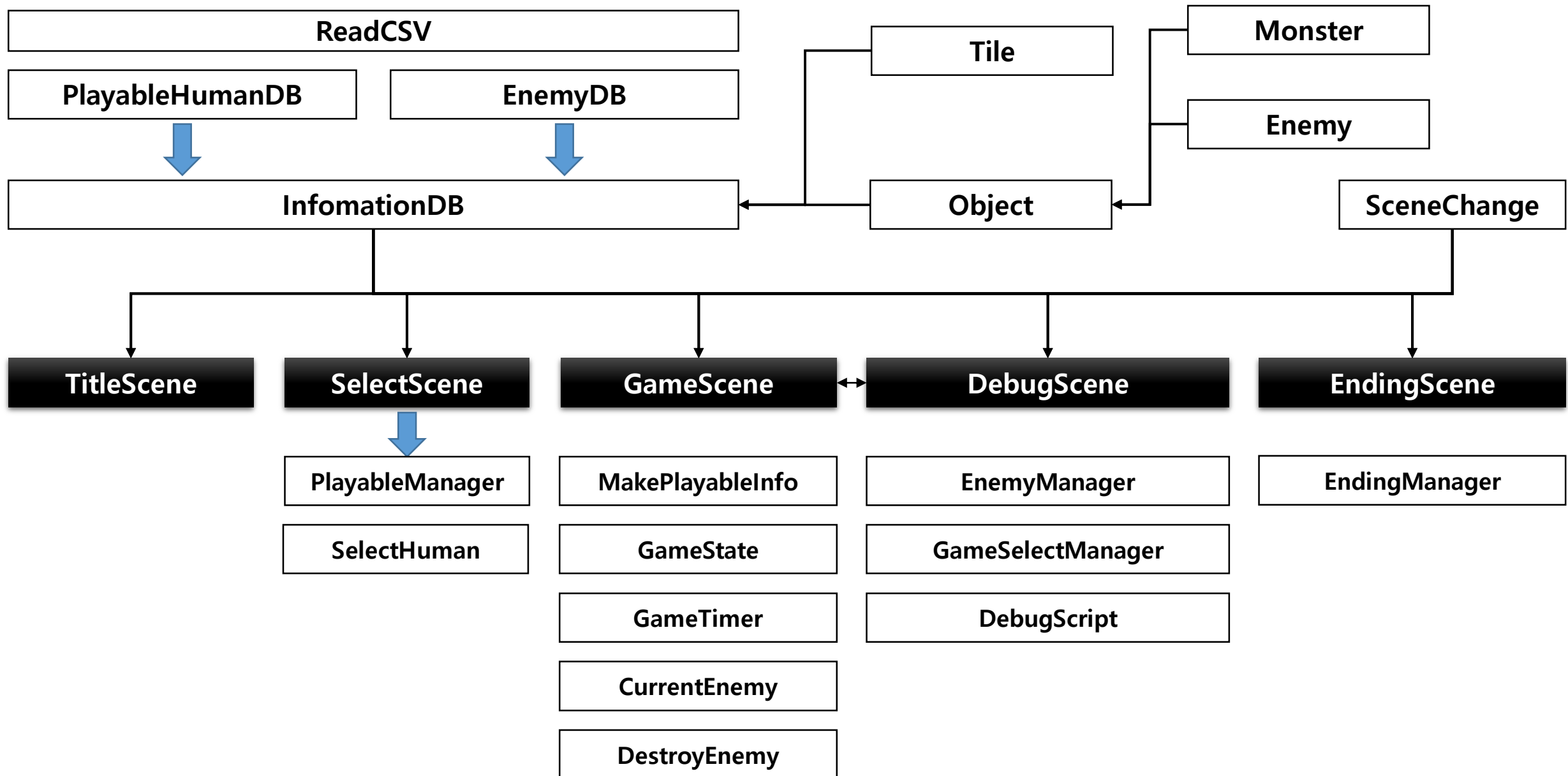
ID	TYPE	NAME	HP	ATTACK	DEFFENCE	SPEED	COST	RANGE	ATTACKDI	APPEARDI	ROUNDHP
0	EMPTY	EMPTY	0	0	0	0	0	0	0	0	0
1	EN	호남이	1000	20	10	3	0	2	10	10	50
2	EN	댕댕이	3000	10	20	2	0	2	15	30	100
3	EN	아서스	5000	10	30	1	0	2	20	60	200

2. 오브젝트 작성

컴포넌트를 추가하여 **개별적인 관리**가 가능
하며 이에 따른 **추가**가 용이하다



3. 기술 코드 - CLASS



3. 기술 코드 – 공용 DB 변수

```
public struct PLAYABLE_HUMAN_DB
{
    // 사용자 정보
    public int id;           // 식별 번호
    public string type;      // 유형
    public string name;      // 이름
    //
    public int hp;           // 체력
    public int attack;       // 공격력
    public int deffence;     // 방어력
    public int cost;         // 비용
    public int range;

    public float attackDelay;
    //
    public string mainImagePath; // 메인 이미지 Path
    public string logoImagePath; // 로고 이미지 Path

    public Sprite mainImage;
    public Sprite logoImage;

    // 판정 및 유무
    public bool exist;        // DB에서 플레이어가 정보를 가지고 있는지 확인하는 변수
    public bool isSelect;    // CURRENT에 선택되었는지를 확인하는 변수
    public bool isAttack;    // 공격 가능 상태에 대한 여부
}
```

ID	TYPE	NAME	HP	ATTACK	DEFFENCE	COST	RANGE	DELAY	ImagePath	ImageLogoPath
0	EMPTY	EMPTY	0	0	0	0	0	0	empty	empty
1	AD	멜리스	80	40	10	4	1	1	01sheep	01sheep
2	AD	안토니오	100	50	2	5	1	1	01sheep	01sheep
3	AD	냥캣	80	10	12	1	2	1	08cat	08cat
4	AD	키라라	60	20	13	2	1	1	08cat	08cat
5	AD	펑	40	35	14	2	1	1	05peng	05peng
6	TD	오구리	20	80	15	10	3	2	09duck	09duck
7	TD	육구리	20	100	16	11	3	2	09duck	09duck
8	TD	칠구리	20	150	17	12	4	2	09duck	09duck
9	TD	팔구리	50	200	18	13	4	2	09duck	09duck

```
public struct ENEMY_DB
{
    // 사용자 정보
    public int id;           // 식별 번호
    public string type;      // 유형
    public string name;      // 이름
    //
    public int hp;           // 체력
    public int attack;       // 공격력
    public int deffence;     // 방어력
    public int speed;        // 속도
    public int cost;         // 비용
    public int range;        // 사거리

    public float attackDelay; // 공격 딜레이
    public int appearDelay;   // 등장 딜레이

    public int roundHP;       // 라운드 별 증가량
}
```

ID	TYPE	NAME	HP	ATTACK	DEFFENCE	SPEED	COST	RANGE	ATTACKDELAY	APPEARDELAY	ROUNDHP
0	EMPTY	EMPTY	0	0	0	0	0	0	0	0	0
1	EN	호랑이	1000	20	10	3	0	2	10	10	50
2	EN	땡땡이	3000	10	20	2	0	2	15	30	100
3	EN	아서스	5000	10	30	1	0	2	20	60	200

DB 구성 및 개발 내역

3. 기술 코드 - 공용 오브젝트 관리

```
// 데이터 베이스 공간
public PlayableInfo.PLAYABLE_HUMAN_DB[] playableDB = new PlayableInfo.PLAYABLE_HUMAN_DB[30];
public PlayableInfo.ENEMY_DB[] enemyDB = new PlayableInfo.ENEMY_DB[30];

public GameObject[] playableGameObjectDB = new GameObject[30];
public GameObject[] enemyGameObjectDB = new GameObject[30];

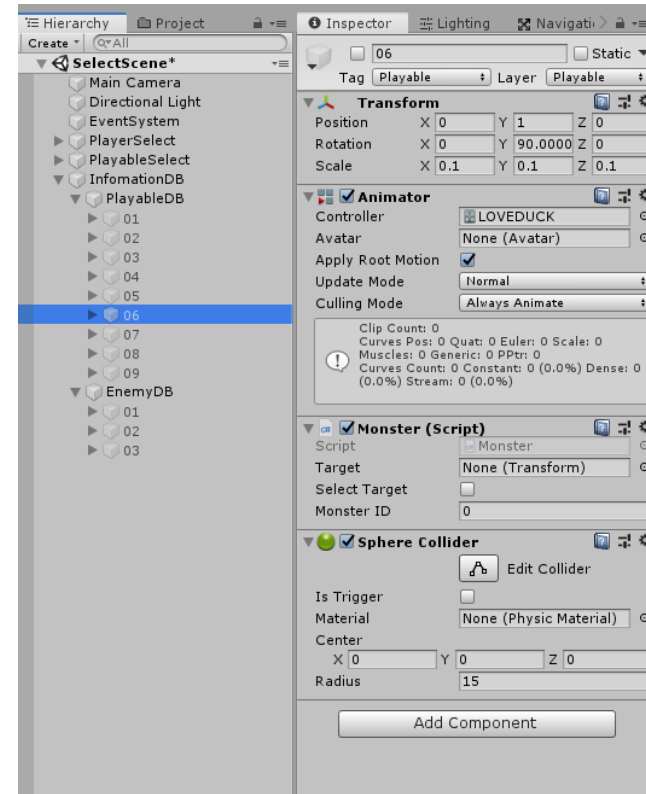
public int playableDBSize;
public int currentDBID;

private void ReadPlayableHumanDB()...

private void ReadEnemyDB()...

private void ReadDB()
{
    ReadPlayableHumanDB();
    ReadEnemyDB();
}
```

변수에 대한 내역은 InfomationDB 스크립트

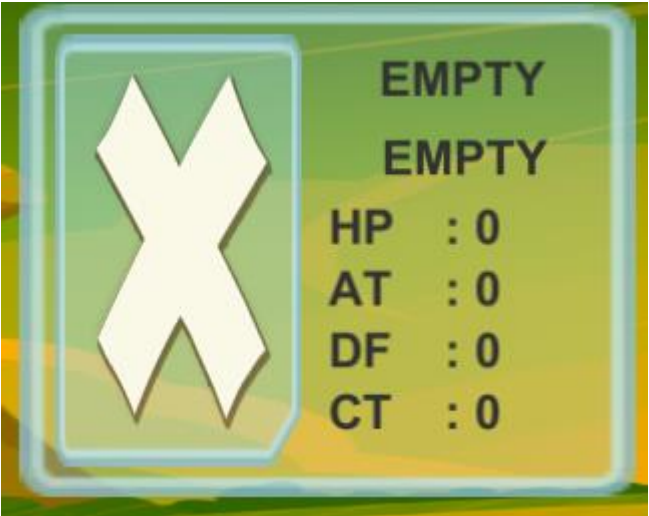


InfomationDB 에 오브젝트 구성

3. 기술 코드 – 동적인터페이스



ID	TYPE	NAME
0	EMPTY	EMPTY
1	AD	멜리스
2	AD	안토니오
3	AD	냥켓
4	AD	키라라
5	AD	팡
6	TD	오구리
7	TD	육구리
8	TD	칠구리
9	TD	팔구리



```
size = InfomationDB.Instance.playableDBSize;
currentID = 0;
for (int i = 0; i < size; i++)
{
    newUsePlayable[i] = Instantiate<Button>(usePlayable) as Button;

    newUsePlayable[i].transform.parent = transform;
    newUsePlayable[i].transform.localScale = new Vector3(1, 1, 1);

    newUsePlayable[i].transform.position = new Vector2(
        Camera.main.pixelWidth / 96 + Camera.main.pixelWidth / 5 +
        ((Camera.main.pixelWidth/3) - Camera.main.pixelWidth / 24) * (i%3),
        (Camera.main.pixelHeight - Camera.main.pixelHeight/4) - (Camera.main.pixelHeight / 2) * (int)(i/3));

    newUseScript[i] = newUsePlayable[i].GetComponent<ChoiceHuman>();
    newUseScript[i].buttonId = i;
}
```

카메라 해상도 / DB에 저장된 플레이어 수에 맞게 유저 정보를 생성하는 코드

3. 기술 코드 – 기타 게임 상호작용 소스코드

```
if (Input.GetMouseButtonDown(0))
{
    if (Physics.Raycast(ray, out hit, 100, 1 << LayerMask.NameToLayer("Playable")))
    {
        Debug.Log(hit.collider.gameObject.layer);
        // Debug.Log(hit.collider.gameObject.layer);
        Destroy(hit.collider.gameObject);
        deleteMode = false;
    }

    if (Physics.Raycast(ray, out hit, 100, 1 << LayerMask.NameToLayer("GROUND")))
    {
        Debug.Log(hit.collider.gameObject.layer);
        hit.collider.GetComponent<Tile>().playerCheck = false;
        deleteMode = false;
    }

    if (Physics.Raycast(ray, out hit, 100, 1 << LayerMask.NameToLayer("HILL")))
    {
        Debug.Log(hit.collider.gameObject.layer);
        hit.collider.GetComponent<Tile>().playerCheck = false;
        deleteMode = false;
    }
}
```

타일과 마우스간의 상호작용 코드
(오브젝트 삭제 소스코드)

```
for (int i = 0; i < size ; i++)
{
    waypoint.Add(transform.parent.GetChild(enemyID).GetChild(i).transform);
}
waypoint.Add(endPosition);
waypoint.Add(endPosition);

agent = GetComponent<NavMeshAgent>();
agent.autoBraking = false;

waypoint[waypoint.Count - 1] = GameObject.Find("DestPosition").transform;
waypoint[waypoint.Count - 2] = GameObject.Find("DestPosition").transform;

GotoNext();
```

Mesh를 이용한 적기의 움직임을
관리하는 소스코드

```
if(hit.collider.GetComponent<Tile>().playerCheck == false)
{
    if (costInfo.cost > InfomationDB.Instance.playable[checkID].cost - 1)
    {
        costInfo.cost -= InfomationDB.Instance.playable[checkID].cost;

        GameObject playableObject =
            Instantiate<GameObject>(InfomationDB.Instance.playableObject[checkID].gameObject);
        playableObject.transform.position = hit.collider.gameObject.transform.position + new Vector3(0, 1, 0);
        playableObject.SetActive(true);
        playableObject.GetComponent<Monster>().monsterID = checkID;

        hit.collider.GetComponent<Tile>().playerCheck = true;

        costInfo.CostInfo();
    }
}
```

플레이어 타워 오브젝트 생성 코드

4. 부록

포트폴리오 개발에 대한 의의

1. 타워 디펜스에 필요한 다양한 오브젝트를 관리하기 위한 방안 모색
2. DB관리를 통해 타워 디펜스 개발을 위한 일종의 툴 개발
3. 기획 단계부터 DB 관리에 필요한 자료가 무엇인지에 대한 의식 함양
4. 툴 개발 및 외부 데이터 관리 방안 탐색

읽어 주셔서 감사합니다.