SW 설계기초 1 조 최종 보고서

<프로젝트명 : Card Of Defense>

전공	소프트웨어학과			
학번	21011776	21011758	21011804	21011778
이름	안승기	이제희	양남욱	염지환
제출일	2024.12.20			

세종대학교
SEJONG UNIVERSITY

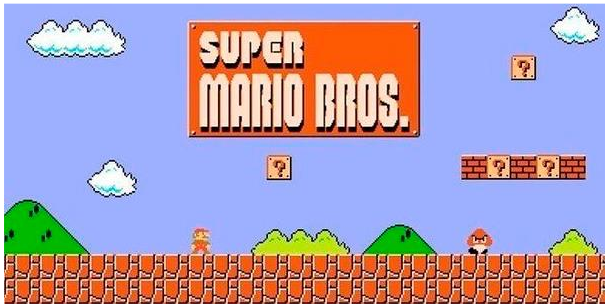
[목차]

내용

1. 기획 과정

I. 시장 조사 및 장르 선정

프로젝트를 시작하기 전 게임을 정하기 위해 가장 먼저 10 주 정도의 짧은 기간 안에 어떤 게임을 만들어야 평가 지표를 만족하는 게임을 만들 수 있는가를 고민하였다. 게임의 볼륨과 기능이 다양한 최근의 게임들을 목표로 삼으면 개발기간이 길어질 것으로 예상하여 고전 및 초창기 모바일 게임을 조사하였다.



각자 시중에 발매된 조사한 결과 각 장르에서 대표적인 게임 2 가지가 후보로 좁혀졌으며 마리오의 경우 사용자가 게임의 시나리오를 진행하며 사용자의 컨트롤과 게임의 흐름에 따라 재미를 느낄 수 있는 게임이며, 팔라독은 사용자가 직접 상호작용 하는 요소는 적더라도 여러 경우의 수를 활용해 사용자가 다양한 경험을 반복하며 재미를 느끼게 하는 게임이었다.

개발 기간을 고려해 보았을 때 마리오와 같은 게임은 다양한 맵을 구성하고 짜임새 있는 시나리오 구성이 필요하기 때문에 개발시간이 짧은 해당 프로젝트에는 부적합하다고 생각하였다.

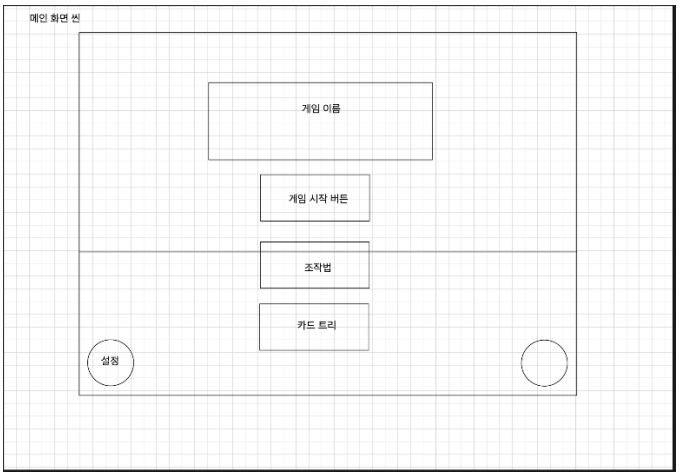
팔라독과 같은 게임은 사용자가 상호작용하는 부분은 적어도 비슷한 스테이지를 플레이 하더라도 사용자의 선택에 따라 게임의 흐름를 크게 변화시킬 수 있으므로 코드의 효율적인 구현 및 재사용을 통해 짧은 개발 기간 안에 재밌는

게임을 만들 수 있을 것이라고 생각하여 팔라독과 같은 형태의 디펜스 게임을 개발하기로 결정하였다

팔라독을 그대로 벤치마킹하여 만들기에는 기존에 존재하는 디펜스 게임들과 크게 다를 바가 없을 거라고 생각하여 추가적으로 접목할 게임 요소를 찾아본 결과 슬레이 더 스파이어라는 게임의 카드를 이용한 전투에서 아이디어를 얻었다. 최종적으로는 일반 디펜스 게임과는 달리 카드를 통한 실시간 상호작용을 통해 플레이어의 선택의 폭을 넓히는 방법을 채택하기로 하였다.

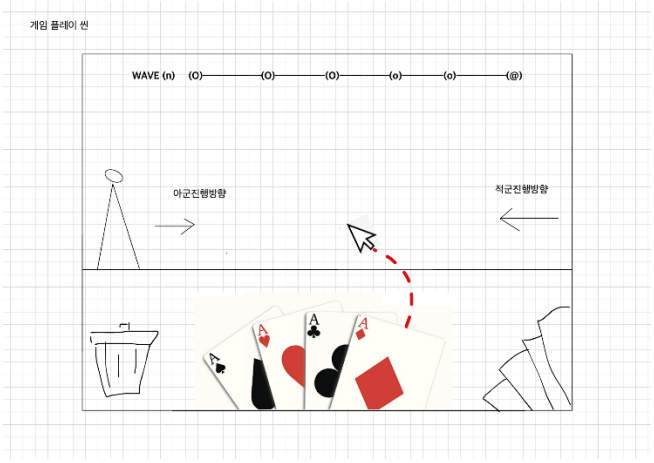
II. 게임 규칙 및 시나리오 구상

벤치마킹할 게임들을 정한 뒤 바로 게임 규칙과 시나리오를 구상하였다 초기에 구상한 게임 진행 시나리오는 다음과 같다

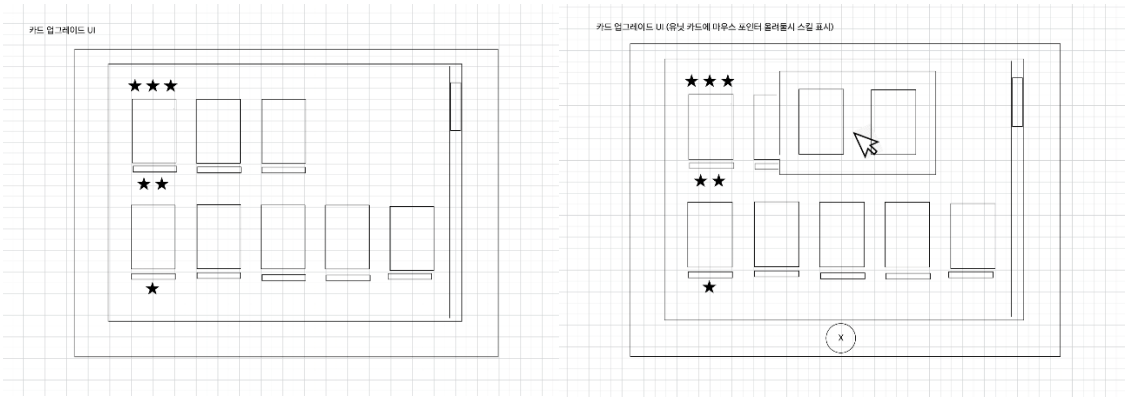


먼저 게임을 실행하게 되면 사용자는 메인 화면을 보게 되며 메인 화면에는 게임 이름과 게임 시작 버튼, 조작법 확인 버튼, 게임에 사용되는 카드 종류와

업그레이드 루트를 정리한 트리를 볼 수 있는 버튼이 있다. 또한 환경 설정 버튼으로 사용자가 게임 플레이 환경을 조정할 수 있도록 할 계획이었다.



메인화면에서 게임 시작 버튼을 누르면 게임 씬으로 이동하며 게임 씬의 맵 왼쪽에는 아군이 지켜야 할 메인 건물이 존재한다 적군 유닛의 경우 맵의 오른쪽에서 생성되어 아군 메인 건물을 공격하기 위해 왼쪽으로 진행한다. 사용자는 해당 적군 유닛들에게서 건물을 지키기 위해 카드를 이용해 유닛을 소환하고, 스킬을 사용하여 적군들을 처치해야한다. 카드 패는 일정 시간마다 전부 버려지고 다시 같은 장수를 덱에서 드로우 하여 게임에 랜덤성을 부여하였다.



게임 중 업그레이드 버튼을 누르면 업그레이드 창이 화면에 표시된다. 웨이브를 클리어하면 얻을 수 있는 재화를 통해 카드 업그레이드가 가능하며. 자신이 가지고 있는 카드를 클릭하여 업그레이드할 수 있는 선택지가 표시되면 해당 선택지를 클릭하여 카드를 업그레이드 할 수 있다.

게임이 시작할 때에는 덱에 6 장의 농부 카드만이 존재하며 사용자가 카드를 업그레이드 하면 덱 목록의 유닛이 업그레이드 되고 해당 유닛에 대응되는 스킬카드를 같이 덱에 추가한다.

III. 차별화 요소

해당 게임은 전통적인 2d 횡방향 디펜스게임의 틀을 가지고 있지만, 앞서 설명한 슬레이 더 스파이어의 카드를 이용한 RPG 전투의 시스템을 일부 사용하여, 사용자가 유닛의 소환 위치, 스킬 사용 타이밍 등을 능동적으로 판단하여 플레이 할 수 있도록 하였다.

또한 랜덤한 드로우를 통해 매번 사용자에게 다른 상황을 제시하여 게임이 단조로워지는 것을 방지하였다.

마지막으로 다양한 유닛과 스킬들을 구현하여 업그레이드 된 유닛과 스킬에 대한 플레이어의 호기심을 유발하고, 다양한 조합마다 다른 경험을 얻을 수 있도록 구성하여 여러 번 플레이를 할 수 있도록 하였다.

IV. 요구사항 명세서 작성

전반적인 게임 시나리오 구성을 하는 과정에서 각자 파트 별로 요구사항 명세서를 작성하였다 해당 명세서는 게임 개발 도중 다소 수정되었으나, 해당 명세서를 자주 참고하여 게임을 개발하는 과정에 있어서 프로젝트의 핵심 요소를 잘 유지할 수 있었다.

파트별 요구사항 명세서 내용을 요약하면 다음과 같다 시나리오 구성보다 먼저 작성된 내용도 존재하여 세부적인 내용은 앞서 설명한 시나리오 구성과 다소 차이가 있을 수 있다.

UI

게임 시작 화면

- Unity UI ToolKit 을 이용한 개발

- 게임 제목을 화면의 중앙 상단에 표시하고 아래에 버튼을 배치
- 각 버튼마다 Scene 이동 할당
- ~~아무 구조물이 배치되지 않은 게임 Scene 을 배경으로 설정~~

  > 별도의 삽화를 배경으로 설정하였다

업그레이드 창

- 현재 덱에 있는 카드들을 스크롤 뷰로 한눈에 볼 수 있게 구현
- 카드는 레벨 순서대로 정렬하여 순서대로 보여 줄 수 있어야 함
- ~~스킬 카드와 유닛 카드를 분리해서 창을 만들어야 하며 상단에 스킬카드와 유닛 카드를 선택할 수 있는 창이 있어야 함.~~

  > 스킬 카드는 카드 트리에서 확인 가능하며 따로 분리할 필요가 없다고 제작하여 창을 따로 제작하지 않았다
- 현재 창을 닫을 수 있는 버튼을 중앙 상단에 표시
- ~~카드에 마우스 커서를 가져다 댈 시 대화 창을 띄워 카드에 대한 설명 표시~~

  > 개발 여건상 모두 삽입하기 어렵다고 판단되어 개발 과정에서 취소되었다.
- 좌상단에 현재 업그레이드 가능한 레벨 수 표시

업그레이드 버튼

- ~~카드 업그레이드 창 하단에 업그레이드 버튼 표시~~

  > 강화할 카드를 클릭하면 업그레이드 대상 카드를 띄워 해당 카드를 클릭시 즉시 업그레이드 되도록 변경.
- ~~업그레이드 가능 횟수가 1 이상인 경우 버튼 활성화 아닐 경우 비활성화~~

  > 업그레이드 버튼이 사라졌으므로 구현할 필요 없음

- ~~업그레이드 과정이 둘로 나뉠 경우 창을 두개로 나누어 표시~~

  > 업그레이드 과정이 나뉠 경우 카드 클릭시 여러 카드가 뜨는 방식으로 변경

카드

카드 기본 구현

- 시나리오 상 존재하는 유닛 카드를 제작
- 각 카드의 등급 및 특징에 따라 구분 가능하게 구현
- 특수 카드의 경우 확연히 구분되게 구현
- 카드는 덱, 패, 더미 뭉치(트래쉬) 중 한 곳에 위치
- 패에 있는 카드들은 카드 수에 따라 자연스러운 부채꼴 형태로 정렬
- ~~카드를 클릭하면 마우스를 따라 이동~~

  > 카드가 패에서 조금 튀어나오게 구현, 마우스를 따라 이동할 시 맵의 전투상황을 보는데 방해가 되어 이동하지않고 확대되도록 수정

- ~~카드가 사용 불가한 상태일 경우 카드 비활성화~~

  > 카드가 사용 불가한 상태일 경우가 거의 존재하지 않으며, 스킬 카드 사용 시에도 그냥 사용이 되지 않고 패에 남아있는 것으로만 처리.

카드 드로우

- 카드는 정해진 시간 주기마다 드로우
- ~~초기에는 3 장을 기본으로 드로우 하게 설정하고 게임 진행에 따라 드로우 개수 증가~~

  > 초기에 기본으로 4 장을 드로우하며 스테이지를 클리어 하면 업그레이드 횟수, 드로우 수, 한 드로우당 사용할 수 있는 비용 수를 획득할 수 있음.

- 정해진 주기가 왔을 때 사용되지 않은 카드들은 버리고 다시 드로우
- 덱에 더 이상 카드가 없을 경우 더미 뭉치의 카드를 다시 덱에 추가
- ~~카드에 효과에 따른 추가 드로우 기능~~

  > 카드를 추가로 드로우하는 특수 카드 구현이 안되었으므로 구현할 필요 없음

- ~~효과에 의해 버려지지 않는 경우 패에 남아있는 기능~~

  > 특수 카드 구현이 안되었으므로 구현 필요성 없음

카드 생성

- ~~초기에는 기본 카드 4 장으로 시작~~

  > 기본 카드는 6 장으로 시작하는 것으로 변경

- ~~플레이어의 레벨이 증가하는 경우 기본 카드 한 장 추가~~

  > 총 유닛 카드 수는 변화 없음, 대신 웨이브 클리어 시 업그레이드 횟수, 드로우 수, 드로우 당 사용 가능 비용 수를 증가를 선택 가능

- ~~플레이어의 레벨은 초기에 빠르게, 후반에 느리게 상승~~

  > 플레이어 레벨은 따로 구현하지 못함

- 한 웨이브를 클리어 했을 경우 업그레이드 카드를 획득
- 카드의 등급에 따라 더 많은 업그레이드 비용이 필요
- 스킬 카드는 기본 카드 강화시 변경된 카드에 해당하는 스킬 카드가 덱에 자동으로 추가
- ~~카드의 등급에 따라 1~2 장의 스킬카드 획득~~

  > 농부를 제외한 카드는 전부 스킬 카드를 하나씩만 가짐.

- ~~특수카드 생성 기능~~

- ~~재화를 통해 카드를 삭제하는 기능을 통해 덱에 불필요한 카드가 너무 많아지는 것을 방지~~

  > 카드 증가가 없으므로 구현할 필요 없음

카드 사용 방식

- 유닛 카드는 패에 있는 유닛 카드를 드래그하여 원하는 위치에 유닛이 소환되도록 구현
- ~~해당 위치가 소환 불가능한 위치라면 사용 불가하도록 구현~~

  > 소환이 불가능한 위치가 사실상 존재하지 않으므로 구현할 필요 없음
- 스킬 카드는 패에 있는 스킬 카드를 해당 직업의 유닛에게 태그하여 유닛이 스킬을 사용하는 방식으로 구현
- 유닛이 해당 스킬을 사용 가능한 경우에만 카드 사용이 가능하도록 구현
- 스킬 카드는 같은 직업군의 유닛만 사용 가능
- 태그된 유닛의 등급이 스킬 카드의 등급 이상일 경우에만 사용 가능하게 구현

유닛

아군 유닛 이동

- 아군 유닛의 정해진 시야 범위내에 적군이 들어올 경우 아군 유닛의 사정거리에 적이 들어올 때 까지 이동한 다음 기본 공격
- 시야에 적군이 있는 경우 전투 상황으로 간주
- 전투 상황이 아닌 경우 아군 유닛은 소환 초기 위치로 이동

유닛 기본 공격

- 유닛은 각자 정해진 사정거리 안에 상대방 유닛이 있으면 이동을 멈추고 기본공격을 수행한다

유닛 일반

- ~~유닛은 공통적으로 최대 체력, 공격력, 마력, 방어력, 마법 저항, 공격 속도, 사야거리를 능력치로 가진다~~

  > 마력, 마법 저항은 밸런스 설정에 큰 어려움을 줄 것으로 예상하여 삭제

- 유닛이 처음 소환될 때의 기본 능력치는 종류마다 다르다
- 유닛이 소환된 후에도 아군이나 적의 스킬에 의해 능력치가 변할 수 있다

유닛 강화 시나리오

- 유닛 카드의 강화 단계는 총 4 단계로 구성
- ~~유닛 카드가 1 단계에서 강화하는 경우 총 근접, 원거리. 마법, 건축, 4 가지 계열 중 하나의 계열을 선택하여 강화 가능~~

  > 건축 유닛의 경우 구현상의 어려움으로 삭제됨

- 카드를 강화할수록 유닛은 특정한 역할에 특화되며 그에 따라 능력치가 변화하고 역할에 맞는 스킬을 소유

유닛 구성

- 유닛 카드는 기본 유닛 카드를 강화하여 얻는 일반 유닛 카드와 ~~보스 클리어 보상으로 획득할 수 있는 특수 유닛 카드로 구분~~

  > 특수 유닛 카드는 구현 취소되어 구현할 필요 없음

- 일반 유닛 카드는 총 25 종으로 구성

  > 건축 계열 유닛이 삭제되고 4 단계 유닛이 계열당 2 종류 씩 삭제되어 총 9 개의 유닛이 삭제, 16 개의 일반 유닛으로만 구성

유닛 스킬 구성

- ~~유닛 카드를 강화하는 경우 이미 덱에 존재하는 스킬 카드는 다음 단계 유닛 카드가 사용하는 스킬 카드로 대체되지 않고 덱에 유지됨~~

  > 덱의 존재하는 스킬 카드는 유닛 카드 업그레이드시 대체되는것으로 변경 -> 스킬 카드를 계속 가져갈 경우 덱의 매수가 설계보다 과도하게 증가

- ~~결과적으로 유닛 스킬 카드는 계속 늘어나기 때문에 덱의 매수가 계속 늘어남~~

  > 덱의 매수는 1 단계에서 2 단계 업그레이드를 제외하면 늘어나지 않음

- 유닛은 그 유닛 카드에 대응하는 스킬 뿐만 아니라 이 유닛이 거쳐온 하위 유닛의 스킬도 사용할 수 있음.

~~특수 카드 / 특수 유닛~~

특수 스킬 카드, 특수 유닛 카드는 보스를 처치할 경우 보상으로 주어지는 일반 카드 외의 카드로 계획하였으나 시간상의 한계로 구현하지 못함.

적

적군 유닛 이동 및 공격

- 적군 유닛의 이동 방향은 아군의 기지 방향
- 적군 유닛은 이동간에 사거리에 아군 유닛이 들어오면 기본공격
- 적군 유닛의 사정거리에 아군 기지가 들어오면 기지를 우선 공격

적군 일반

- 근거리 공격, 근거리 방어, 원거리 공격 등 다양한 유형의 적을 구현
- 웨이브에는 정해진 시간은 따로 존재하지 않음
- 웨이브마다 정해진 수의 적군을 지속적으로 소환

- 웨이브마다 정해진 양의 적군을 모두 처치하였을 경우 다음 웨이브로 넘어갈 수 있도록 구현
- 웨이브가 올라갈 때 마다 적군의 능력치를 향상하고 외관을 변경

보스

- 보스 스테이지를 클리어 할 때마다 적군의 능력치 향상 및 소환 패턴 변경
- 매 보스 스테이지마다 보스의 체력 또는 공격력, 스킬을 강력하게 구현
- 보스는 일정 시간마다 스킬을 사용
- ~~보스 처치 시 특수 스킬 또는 특수 유닛 카드를 리워드로 제공~~

  > 특수 스킬 또는 특수 유닛 카드는 구현되지 않음

2. 초기 개발 과정

I. 개발 및 협업 환경 설정

개발 환경 설정

개발 환경은 초기에 C++ 기반의 콘솔 환경과 유니티 환경 중 어떤 환경을 선택할지 회의를 통해 결정하였다. 조원들 중 유니티 에디터를 활용하여 개발을 해본 경험이 있는 조원은 없었지만, 유니티에서 제공하는 여러 내장함수 및 물리엔진과 에셋을 활용하면 더 완성도 있는 게임을 만들 수 있다고 생각하여 유니티 개발 환경을 선택하였다. 조원 모두 유니티가 처음이고 협업에 익숙하지 않은 점과, 유니티 에디터에서 같은 씬에서 작업 시 충돌 위험이 있다는 것을 파악하여 각자 맡은 파트 별로 씬을 나누어 작업을 진행하였다.(UI, 아군 유닛, 몬스터, 카드 등). 각자 씬에서 작업 후 대면 회의나 온라인 화상통화(디스코드) 등을 통하여 실시간으로 소통하며 main 씬에 개발 내용들을 합치는 방식으로 개발을 진행하였다. 이를 통하여 충돌이 일어나는 상황을 최소화할 수 있었다.

또한 카드를 사용하여 유닛 소환 및 스킬사용, UI 를 통한 업그레이드 및 상호작용, 몬스터와 아군 유닛과의 전투 등 기본적인 기능들이 구현된 후부터는 각자의 씬에서 추가적으로 필요한 객체들을 빠르게 작업하여 추가할 수 있어 작업의 속도를 향상키실 수 있었다.

협업 환경 설정

협업은 기본적으로 깃허브를 통하여 진행하였다. 개발기간이 길지 않다는 점과 서로 다른 씬을 작업한 후 main 씬에 합치는 방식으로 개발환경을 정한 점 등을 고려하였을 때 Github Flow 방식을 선택하여 협업을 진행하였다. 유니티 에디터 특성 상, 한번에 매우 많은 양의 커밋이 올라가는 경우가 많은데 이를 효과적으로 관리하기 위하여 깃허브 데스크탑을 이용하여 commit, push 등 기본적인 작업을 진행하였다.

회의 정리 및 개발현황 등은 Notion 을 통하여 정리해여 회의 내용이나 개발해야 하는 내용들을 확인하기 편하게 하였다. 또한 Notion 에 있는 캘린더 기능을 활용하여 회의 일정 및 개발목표를 표시하여 더 효율적인 협업이 가능하도록 하였다. 평균적으로 주 1~2 회의 대면 회의를 진행하였는데, 작업을 하던 도중 충돌이 발생하거나 개발 내용이 다른 파트와 연관되어 즉각적인 소통이 필요한 경우 온라인 화상채팅(디스코드 등)을 사용하였다.

II. 파트 별 개발 내용

UI

게임 시작 화면

- 게임의 시작화면을 Unity UI ToolKit 을 이용한 개발
- 게임시작 버튼 클릭시 Map 씬으로 전환되도록 구현
- 삽화를 삽입하여 기본 화면을 구성
- 스킬트리 버튼 클릭시 스킬트리가 보여지도록 구현

- 스킬트리 창에서 각 직업군 별로 버튼을 할당하여 입력에 따라 해당 직업군의 트리를 보여지도록 구현

맵 및 업그레이드

- 스크롤를 사용하여 가로로 긴 형태의 지도를 구현
- 게임 데이터와 연동하여 현재 별의 수 및 드로우 수를 표시
- 카드 업그레이드 버튼 클릭시 업그레이드 창이 나오도록 구현

➔ 카드 업그레이드 창에서는 현재 업그레이드 포인트가 표시되고 카드에 마우스를 올려놓을 시 업그레이드 가능한 카드들이 나오도록 구현하였다. 업그레이드 성공 시 덱의 정보를 다시 불러와 즉각적으로 변경된 정보가 업그레이드 창에 보이도록 하였다.

- 지도 상의 스테이지들에 버튼을 할당하여 스테이지 버튼 클릭시 해당 스테이지로 이동되도록 구현

메인 전투 화면

- 전투 화면으로 전환 시 현재 스테이지 정보(1-2, 3-2 등)이 화면에 표시되는 기능 구현
- 화면 좌측 상단에 지도 및 직업트리 버튼을 통하여 전투 중에도 스킬 및 현재 스테이지를 확인 가능하도록 구현
- 덱 및 더미 버튼 클릭시 덱에 대한 데이터를 읽어서 현재 아직 드로우 하지 않은 카드들과 드로우된 후 버려진 카드들을 보여주는 기능
- 현재 턴의 남은 별(에너지)를 데이터를 읽어와 맵 하단에 표시
- 전투 맵의 좌우 끝에 카메라를 좌우 이동시킬 수 있는 버튼을 구현
- 스테이지 클리어 시, 보상을 선택하는 창을 뛰운 후 맵 씬으로 전환하는 기능
- 스테이지 클리어 실패 시 재시작 하는 기능 구현

카드

기본 카드 구현

- 카드 템플릿을 통하여 10 여 종의 유닛 및 스킬카드 제작
- 별(단계), 직업군 등에 대한 정보, 카드의 현재 위치 정보등을 할당
- 드로우 시 상황에 맞게 정렬되는 기능 구현
- 스크립터블 오브젝트를 사용하여 유닛 카드에 필요한 정보(이미지, 해당 유닛의 스킬 카드, 업그레이드 가능 유닛)를 할당
- 마우스 클릭시 크기를 확대하여 강조하는 효과 구현
- 마우스 입력을 통하여 유닛 및 스킬 카드가 사용되도록 구현
- 상황에 따라 카드 사용 가능 여부를 판단

➔ 현재 남아 있는 별의 개수가 카드의 별의 개수보다 높은 경우에만 카드 사용이 가능하도록 구현하였다. 스킬카드의 경우 해당 스킬을 사용할 수 있는 유닛인 경우에만 사용 가능하도록 설계하였다.

덱 및 드로우 관리

- 스크립터블 오브젝트 배열을 활용하여 덱 관리
- 덱에 더 이상 카드가 없어 드로우가 불가능 한 경우 덱 초기화

➔ 덱 SO 에서 정보를 읽어와 초기화. 유닛 카드가 추가될 때 해당 유닛카드의 스킬카드도 같이 추가한 후 랜덤하게 덱이 섞이도록 구현

- 게임 데이터에서 드로우 매수를 읽어와 일정 주기마다 해당 매수만큼 덱에서 드로우 하는 기능 구현

카드 업그레이드

- 카드 SO 에 해당 유닛 카드의 업그레이드 가능 카드를 할당
- 업그레이드 함수를 통하여 비용를 고려하여 업그레이드 가능 여부 판단
- UI 와의 상호작용을 통하여 UI 상에서 업그레이드 함수를 호출하여 사용
- 업그레이드 실행 후 덱에 정보를 업그레이드 된 카드로 변경되도록 구현

유닛

유닛의 기본적인 동작은 Unit 클래스에 스크립트로 구현하였으며 모든 유닛은 Unit 클래스를 상속하여 구현하였다.

아군 유닛 이동

- 열거형 변수를 통해 대기, 접근, 전투, 복귀 4 개의 상태를 가질 수 있는 변수를 Unit 클래스에 추가하여 상태에 따라 이동하도록 구현
- 상태의 변화 같은 경우 적군을 감지하여 판단하므로 유니티의 RayCast 기능을 활용하여 사정거리, 시야거리 내의 유닛을 감지함
- 이동은 유닛의 이동속도와 순간 시간 변화량, 방향 벡터를 곱하여 유닛의 위치를 연속적으로 변화시키는 것으로 구현.
- 유닛이 이동중일 경우 에셋의 애니메이션을 삽입하여 유닛이 뛰는 모션을 표현

유닛 기본 공격

- 사정거리 Ray 에 적군이 감지되면 유닛은 attack()메소드를 지속적으로 호출하여 1 초에 공격속도만큼 공격을 진행
- 일정 시간마다 루틴을 진행시키는 방법으로는 유니티의 Coroutine() 메소드를 사용함

- 전사, 마법사, 궁수, 마법사의 전직인 사제 직군의 일반공격이 다르므로 다른 기본공격은 Unit 클래스의 attack()함수를 오버라이딩 하여 다시 구현
- 유닛마다 기본 공격 이펙트를 가지고 있으며 공격 타이밍에 맞게 스프라이트 애니메이션 객체를 생성하고 삭제하는 것으로 구현함

유닛 일반

- 유닛의 스텟은 Unit 클래스에 public 변수를 두어 추후 쉽게 유닛의 스텟을 변경할 수 있도록 하였다.
- 유닛에게는 damaged 함수가 있어 적에 의해 호출되면 유닛의 hp 가 줄어들고 hp 가 0 보다 낮을 경우 사망 애니메이션이 출력되고 유닛이 제거된다

유닛 구성

- 유닛은 기본 유닛 1 개, 각 계열별로 2 단계 1 종, 3 단계 2 종, 4 단계 2 종으로 계열별로 5 개의 유닛이 존재하므로 총 16 종으로 구성된다
- 유닛 스크립트는 전부 Unit 클래스를 상속하여 만들어졌고 특정 유닛만의 특정한 동작이 필요할 경우 메소드를 추가하거나 Unit 의 메소드를 오버라이딩하여 구현함

유닛 스킬 구성

- 유닛 스킬은 별도의 프리팹으로 만들어 스킬마다 하나의 스크립트를 가지고 있음
- 유닛 스킬의 경우 Skill 클래스를 상속받긴 하나 이는 유닛의 상태를 스킬에 전달할 뿐 큰 역할을 하지 않는다
- 유닛 스킬은 차별화를 위해 모든 스킬이 상이한 스크립트를 가지고 있음
- 유닛 스킬은 스킬 카드에 의해 유닛 스킬 프리팹이 유닛에게 전달되고 카드에 의해 유닛의 cast()함수가 호출되어 유닛의 정보가 스킬 프리팹에

전달되고 해당 프리팹 오브젝트를 유닛을 기준으로 일정한 위치에 생성하는 방식으로 동작함.

적

적군 유닛 이동 및 공격

- 적군 유닛의 이동 역시 유닛과 같은 방식으로 구현됨.
- 적군 유닛은 이동간에 사정거리에 아군 유닛이 들어오면 기본공격을 하며 적군은 시야에는 RayCast 를, 사정거리에는 오브젝트간 거리계산을 사용한다.
- 적군은 사정거리에 아군 기지가 들어오면 기지를 우선 공격하도록 설정되어있다.

적군 일반

- 적군은 스테이지 진행에 따라 달라지는 4 개의 종족과 종족마다 근접 단거리 공격, 근접 장거리 공격, 원거리 공격, 방어, 마법 공격 각각에 특화된 5 가지 종류의 유닛을 가지고 있다
- 웨이브는 waveSO Scriptable Object 를 통해 구현하였으며 해당 SO 에는 어떤 유닛이 어느 주기로 나올지 저장되어있으며 스테이지에 남아있는 적군 수를 카운트 하여 모든 적이 사망했을 경우 스테이지를 종료.
- 웨이브가 올라갈 때 마다 적군의 능력치를 향상하고 보스 스테이지를 클리어한 다음 스테이지마다 유닛의 외형이 변경됨.

보스

- 매 보스 스테이지마다 보스의 체력 또는 공격력, 스킬을 강력하게 구현
- 보스는 일정 시간마다 스킬을 사용하며 스킬 스크립트는 프리팹을 통해 따로 구현하거나 보스 스크립트 안에 내장하는 방식으로 구현함.

3. 피드백 반영 및 기능 추가

I. 유닛간 밸런스 조정

		Goblin(이동속도 빠름)				Skeleton(체력 낮고 공속이빠름)				Ice(한방 한방이 쎔)				Devil(밸런스적임)			
	wave	1	2	3	boss1	5	6	7	boss2	9	10	11	boss3	13	14	15	boss4
short	hp	70	90	100	1000	100	120	130	1500	170	200	220	2500	300	400	450	4000
	maxhp	70	90	100	1000	100	120	130	1500	170	200	220	2500	300	400	450	4000
	def	10	10	15	20	15	15	20	25	25	25	30	30	40	40	50	50
	atk	20	20	25	25	20	20	25	30	40	40	50	40	50	50	60	60
	movespeed	5	5	5	2	3	3	3	2	2	2	2	2	3	3	3	2
	attackspeed	3	3	3	3	6	6	6	3	1	1	1	3	3	3	3	3
long	hp	70	80	80		80	100	110		150	170	170		200	250	300	
	maxhp	70	80	80		80	100	110		150	170	170		200	250	300	
	def	10	10	10		10	10	15		20	20	25		30	30	40	
	atk	20	20	25		20	25	30		40	40	50		50	50	60	
	movespeed	3	5	5		2	2	2		1.5	1.5	1.5		3	3	3	
	attackspeed	2	2	2		5	5	5		1	1	1		3	3	3	
shield	hp	120	150	150		130	160	200		250	350	500		500	650	800	
	maxhp	120	150	150		130	160	200		250	350	500		500	650	800	
	def	20	20	30		20	20	25		35	35	40		60	60	60	
	atk	15	15	15		15	15	15		20	20	20		30	30	30	
	movespeed	5	5	5		2	2	2		1.5	1.5	1.5		2	2	2	
	attackspeed	2	2	2		3	3	3		1	1	1		3	3	3	
spear	hp	70	90	100		100	120	130		170	200	220		300	400	450	
	maxhp	70	90	100		100	120	130		170	200	220		300	400	450	
	def	10	10	15		15	15	20		25	25	30		40	40	50	
	atk	20	20	25		20	20	25		40	40	50		50	50	60	
	movespeed	5	5	5		3	3	3		2	2	2		3	3	3	
	attackspeed	3	3	3		6	6	6		1	1	1		3	3	3	
mage	hp	70	80	80		80	100	110		150	170	170		200	250	300	
	maxhp	70	80	80		80	100	110		150	170	170		200	250	300	
	def	10	10	10		10	10	15		20	20	25		30	30	40	
	atk	20	20	25		20	25	30		40	40	50		50	50	60	
	movespeed	3	5	5		2	2	2		1.5	1.5	1.5		3	3	3	
	attackspeed	2	2	2		5	5	5		1	1	1		3	3	3	

Level1	Level2		Level3			Level4	
	Warrior		Gladiator		Knight	Berserker	Warlord
	hp	150	hp	200	hp 250	hp 300	hp 400
	maxhp	150	maxhp	200	maxhp 250	maxhp 300	maxhp 400
	def	10	def	20	def 30	def 30	def 40
	atk	80	atk	120	atk 100	atk 140	atk 120
	movespeed	2	movespeed	2	movespeed 1	movespeed 2	movespeed 3
	attackspeed	1	attackspeed	2	attackspeed 1	attackspeed 2	attackspeed 1
Farmer	Mage		Sorcerer		Priest	ArchMage	Bishop
hp 100	hp	100	hp	150	hp 180	hp 200	hp 250
maxhp 100	maxhp	100	maxhp	150	maxhp 180	maxhp 200	maxhp 250
def 0	def	0	def	5	def 10	def 10	def 15
atk 50	atk	120	atk	160	atk 100	atk 200	atk 200
movespeed 2	movespeed	2	movespeed	2	movespeed 2	movespeed 2	movespeed 2
attackspeed 1	attackspeed	1	attackspeed	1.5	attackspeed 3	attackspeed 1.5	attackspeed 3
	Archer		SharpShooter		Hunter	BowMaster	Explorer

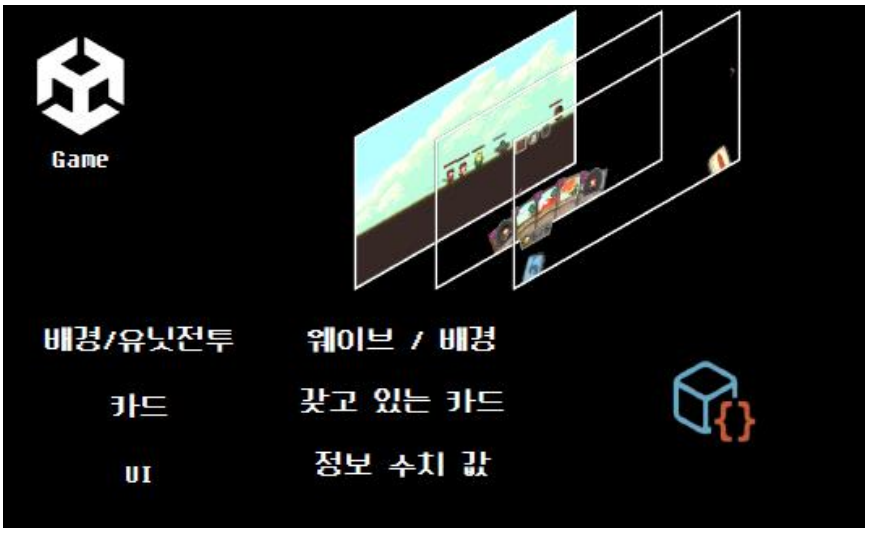
게임의 대략적인 틀을 확립한 뒤에는 적군과 아군을 맡은 조원 둘이 구글 스프레드 시트를 이용하여 밸런스를 맞추었다. 밸런스는 초반에는 게임이 쉬워보이다가 후반으로 갈수록 급격하게 어려워지는 구조를 채택하여, 유닛의 구성이 다양해지는 후반에 더 많은 고민을 할 수 있도록 설계하였다.

실제 엑셀 표는 아래 주소에서 확인 가능하다

https://docs.google.com/spreadsheets/d/1pwPHNmjPPq65XPlWuvP3g8SyRj_lwLWmJcUOGJP51zk/edit?gid=0#gid=0

II. 스테이지 시스템 채택

초기에는 하나에 맵에서 점수를 높여가는 점수제 게임을 구상했으나, 피드백을 통해 스테이지가 필요할 것 같다는 판단을 하게 되었고 사용자에게 더 많은 경험을 주기 위해 다양한 스테이지를 빠르게 만들어야 했다.



따라서 Scriptable 오브젝트를 이용하여 씬 내부에서 스테이지를 구성하기 위한 정보들을 저장 및 관리하였고, SO 의 배열을 만들어 여러 스테이지를 구현하였다.

그리고 맵 Scene 을 따로 제작한 다음 맵 씬의 각 버튼들 마다 SO 배열의 각 스테이지를 호출하여 사용자가 버튼을 클릭하면 전투 Scene 과 그에 해당하는 스테이지 정보를 함께 불러와 스테이지를 구성하는 방식으로 구현하였다.

III. 사운드 출력

게임의 사운드가 없어 비교적 게임의 색채가 부족했기에 마지막으로 게임에 사운드를 추가하기로 하였다. 어떻게 해야 빠른 시간 내에 여러 사운드를 출력할 수 있을 지 고민하였고, 싱글톤 패턴을 이용한 Audio Manager 오브젝트와 스크립트를 이용하여 빠른 구현을 할 수 있었다.

싱글톤 패턴을 사용하면 프로젝트 내의 어떤 클래스에서도 해당 스크립트의 함수를 사용할 수 있으며 이를 통해 각 조원이 자신이 사운드를 넣어야 할 위치에 AudioManager 의 사운드 출력 함수를 호출하여 사운드를 출력할 수 있었다.

따라서 AudioManager 는 한 사람이 구현하고 모두가 같은 함수를 통해 오디오를 출력할 수 있게되어 개발 시간을 단축할 수 있었다.

4. 결과

게임은 성공적으로 빌드되었고 플레이 해본 결과 대부분 의도했던 방식으로 동작하는 것을 확인하였다 하지만 몇가지 아쉬운 점이 남아있었다.

I. 개선점

개발 초기에 계획했던 것보다 많은 부분을 구현하지 못해 아쉬운 점들이나 추가로 더 구현했으면 게임의 완성도가 올라갈 만한 요소들을 정리해 보았다

- 유닛과 유닛의 스킬 설명 등이 좀 더 들어가 있으면 좋았을 것이라 생각한다
- 몇몇 스킬의 경우 스킬이 한 방향으로만 구현되어 있는데 이를 수정하여 양방향 사용이 가능하도록 수정하면 좋을 것 같다
- 유닛의 이동이 물리 기반이 아닌 위치 기반으로 동작하는데 이를 물리기반으로 바꾸면 더 역동적인 게임 개발이 가능할 것 같다

- 메인 화면이나 맵 화면에 사용자가 설정을 변경할 수 있는 설정창이 있으면 좋을 것 같다
- 카드 사용 및 업그레이드 시에 애니메이션이나 VFX 를 추가했으면 좋았을 것 같다

II. 조원 별 프로젝트 후기

안승기

적은 시간안에 결과물을 내야하는 프로젝트를 설계하고 개발해야했기 때문에 팀원 모두 가장 잘하는 부분을 맡아서 할 수 밖에 없어, 접해보지 않은 분야들을 마주해보지 않았던 점이 아쉽지만 각자 아이디어들을 공유하며 회의하는 과정을 많이 거치며 약간의 개념은 머리속으로 잡을 수 있는 경험이었다. 게임을 설계하고 개발하는 과정을 거치며 프로그램을 바라보는 안목을 얻은 것 같다. 협업 또한 처음 진행하여 낯선 부분이 많았지만 머리속으로 개념을 많이 잡아서 앞으로 프로젝트를 설계하는 과정에서 어떤 부분을 신경써야하는지 많이 알아갈 수 있었다. 앞으로의 개발에 가장 좋은 영향을 끼칠 프로젝트를 경험했다고 생각한다.

이제희

이번 SW 설계기초를 통해서 처음으로 게임을 만들어 봤다. Unity 게임엔진을 처음 이용하여 만든 게임이라 처음 개발을 시작할 때 어려움이 있었지만 프로젝트를 완성하는 과정에서 많은 것을 배웠다. 처음 구상할 때 쉽게 할 수 있다고 생각했던 부분도 많이 오래 걸리고 많은 risk 가 있었다. 하지만 개발하는 것이 아닌 그러한 오류를 수정하는 과정에서 더 많은 것을 배웠다. 또한 우리만의 게임을 만들어서 더더욱 뿌듯함을 느꼈다. 어느정도 완성을 시켰으나 좀 더 보안을 하여 이 게임을 유포하는 것까지 이루고자 하는 바람이 생겼다. 이번 프로젝트를 통해 다음 프로젝트를 더 하고 싶다는 의욕도 생겼다.

양남욱

협업을 해본 경험이 많지 않았는데 이번 프로젝트를 통해 협업에 대한 이해를 높일 수 있었다. 개발 기간이 길지 않았고, 처음 사용하는 게임엔진과 언어를 사용하여 개발을 진행하다보니 처음 예상했던 수준의 완성도까지는 도달하지 못해 아쉬움이 남는다. 또한 초기에는 개발에 대한 비중을 크게 고려하였는데, 프로젝트가 진행될수록 초기 기획의 중요성에 대해 깨달았다. 여러 시행착오 끝에 조원들과 협력하여 하나의 우리만의 게임을 완성시켜서 의미가 큰 경험이었다고 생각한다.

염지환

짧은 시간 내에 써보지 않은 언어와 엔진을 사용하여 게임을 개발하는 것이 처음에는 매우 막막하게 느껴졌으나, 개발에 필요한 지식을 바로 검색하여 적용하는 것이 점점 적응이 되어 하나의 프로젝트를 진행 완료했다는 것이 매우 뿌듯했다. 그러나 단편적인 지식만 가지고 개발을 진행했기에 덜 효율적이고 게임이 무거워지도록 개발한 측면도 있어 이를 더 공부해보고 싶다. 또한 협업을 통해 깃허브를 통한 워크플로우를 익힐 수 있었고, 협업을 할 때 시스템을 잘 구축하는 것이 빠른 개발에 큰 도움이 된다는 것을 깨달을 수 있었다.

GitHub Repository 주소

https://github.com/jehhuilee/sejongSW