캡스톤디자인 계획 요약서

제출일: 2024.12.03.

캡스톤디자인 제목	Unity Game Engine을 활용한 육성 시뮬레이션 게임
캡스톤디자인 내용	본 프로젝트는 Unity Game Engine으로 만들어진 텍스트 위주로 진행되는 로그라이크 속성의 육성 시뮬레이션 게임이다. 빠르게 성장하는 게임 시장에 맞춰 '무엇이 게임을 재밌게 만드는가'라는 질문에 대해 답하고자 진행된 프로젝트이다. 부수적인 요소들을 제외하고, 게임적 즐거움의 필요조건만을 떠올려 핵심 아이디어를 구성하여 개발이 진행되었으며, 결과적으로 유저가 새로운 것을 발견하는 과정, 주어진 상황을 활용하여 다양한 방식으로 성장하는 과정에서 즐거움을 느낄 수 있도록 하는 것이 본 프로젝트의 의도이다.

조편성 색인번호 202412_41003-03	구분	학 번	이 름	연락처
	조장	19101262	이은혁	010-3527-0656
	조원			
지도교수	권태수		강좌번호	41003

캡스톤디자인 결과 보고서
202412_41003-03

Unity Game Engine을 활용한 육성 시뮬레이션 게임

(Raising Simulation Game Using Unity Game Engine)

지도교수 : 권태수

제출일 : 2024년 12월 03일

조장 : 19101262 이은혁

캡스톤디자인 보고서 제출

윤리서약서

본인은 서울과학기술대학교 컴퓨터공학과 졸업종합시험 전공교과목(대체) 캡스톤디자인 작성에 다음과 같은 윤리의 기본원칙을 준수할 것을 서약합니다.

첫째, 지도교수의 지도를 받아 정직하고 엄정한 캡스톤디자인을 수행하여 학위 졸업 작품에 필요한 프로젝트를 작성함.

둘째, 캡스톤디자인 구현시 위조, 변조, 표절 등 학문적 진실성을 훼손하는 어떤 연구 부정행위도 하지 않았음.

개인정보 수집. 이용 및 활용 동의서

서울과학기술대학교 컴퓨터공학과 캡스톤디자인 구현과 결과보고서 제출을 위해 아래와 같이 개인정보 수집·이용과 관련한 관계법령에 따라 고지하오니 동의하여 주시기 바랍니다.

1. 개인정보 수집·이용·목적: 캡스톤디자인 구현과 결과보고서 수합, 관리 및 보관

2. 개인정보 수집·이용·항목 : 학번, 성명, 연락처

3. 개인정보 보유 및 이용 기간: 캡스톤디자인 보고서 제출 후 4년

4. 상기 개인정보 수집 및 활용 동의서에 관한 사항에 동의하십니까?

■ 동의함 **□ 동의하지 않음**

5. 개인정보「민감정보 처리」규정에 따라 수집하고 활용하고자 합니다.
상기 개인정보 수집 및 활용 동의서에 관한 사항에 동의하십니까?

■ 동의함 **□ 동의하지 않음**

■ 제공된 개인정보는 상기 제시된 목적으로만 수집 및 활용됩니다.

■ 제출된 캡스톤디자인의 내용은 보호되며 사유의 목적으로 이용하지 않고 작성자에게 모든 권한과 책임이 있습니다.

■ 개인정보 보호법에 따라 개인정보를 수집 및 이용에 관하여 거부할 수 있으며, 동의 거부시 캡스톤디자인 결과 등에 불이익이 있을 수 있습니다.

캡스톤디자인 구현과 결과보고서 제출을 위해 위 윤리서약 준수와 개인정보 활용에 아래의 조원(팀)은 동의합니다.

색인번호 202412_41003-03	구 분	성 명	서 명*(성명을 필기체로 넣으세요)*
	조 장	이은혁	*이은혁*
	조 원		
지도교수 권태수			

2024.12.03.

서울과학기술대학교 컴퓨터공학장 귀하

목차

1. 개요 및 목적

1.1 프로젝트 개요

본 프로젝트는 Unity Game Engine으로 만들어진 텍스트 위주로 진행되는 로그라이크 속성의 육성 시뮬레이션 게임이다. 유저에게 게임 내내 수많은 선택지가 던져지며, 선택에 따라 다양한 방식으로 캐릭터는 성장하게 되고 이를 통해 최종 목적인 보스 처치를 향해 나아가게 된다. 유저는 보스전에서 패배하는 등의 정해진 조건에 따라 게임 도중 패배할 수도 있는데, 이 과정을 겪을 때마다 새로운 아이템 해금 등의 성장 요소를 추가하여 각 플레이 경험 자체의 난이도를 조금씩 낮추어 클리어에 용이하게끔 유도한다. 유저가 새로운 것을 발견하는 과정, 주어진 상황을 활용하여 다양한 방식으로 성장하는 과정에서 즐거움을 느낄 수 있도록 하는 것이 본 프로젝트의 의도이다.

1.2 개발 동기

게임 시장은 과거 어느때보다도 빠르게 성장하고 있다. 이 과정에서 전례 없는 수의 게임이 끊임없이 출시되고 있는데, 개중에는 그래픽, 물리 엔진, 사운드 등의 품질적인 외부 요소가 정말 훌륭하지만 게임 플레이적 재미가 부족한 사례도 존재한다. 이런 사례들은 '무엇이 게임을 재미있게 만드는가'라는 질문으로 이어졌으며, 이것의 연장으로 '재미있는 게임'을 만들고자 하는 것이 동기가 되었다.

2. 배경

2.1 시장 조사

분석 과정에선 우선 FPS, RTS, 오픈 월드, 타워 디펜스 등의 게임 플레이를 결정하는 장르는 제외하게 되었다. 이들은 각각 게임 플레이 방식이 되어 플레이어의 기호에 따라 게임을 선택하는 요소가 될 뿐, 장르적 인기와 같은 시장의 흐름과 직접적인 연관은 적다고 생각했기 때문이다. 하나의 게임이 성공했을 때 그것과 비슷한 게임이 우후죽순 출시되는 것은 맞지만, 해당 게임들은 대부분 빠르게 잊히거나 관심을 받지 못하는 반면 오히려 별개로 출시된 새로운 방식의 게임들이

다시 한번 성공의 예시가 되는 경우가 잦기에 이 사례가 게임플레이 장르의 인기를 의미한다고는 하기 어렵다.

때문에 게임의 전개를 정의하는 장르에 집중하게 되었고, 이 과정에서 다양한 게임을 찾아본 결과 두가지의 키워드를 도출해낼 수 있었다. '로그라이크'란 1980년도 인기 게임 'Rogue'에서 파생된 장르로, 해당 게임의 특징인 각 회차의 랜덤성과 선택의 중요성, 영구적 죽음의 요소를 담고 있는 게임을 칭하는 장르이다. 매번 플레이 할 때마다 새로운 경험을 전달하고, 유저의 선택 하나하나가 게임 전개에 큰 영향을 끼치며, 임의의 세이브가 불가능하기에 게임 오버 시 모든 소지품을 잃게 되는 게임들이 해당 장르에 속한다. 그에 반해 '소울라이크'란 비교적 최근인 2009년작 'Demon's Souls'에서 파생된 장르로, 해당 게임의 진행 방식과 조작, 넓게는 세계관적 요소까지 담고 있는 게임을 칭하는 장르이다. 주된 특징으로는 플레이어의 높지 않은 체력으로 인해 보스가 아닌 기본적인 적에게도 패배하기 쉽다는 점과, 특유의 묵직한 조작감과 더불어 실패시엔 대가가 따르는 방식으로 함부로 공격을 난사할 수 없다는 점 등 높은 난이도의 전투 시스템을 갖추었다는 점이 있다.

때문에 본 프로젝트에선 '로그라이크' 시스템의 기본 틀을 따라가되 회차가 진행될 때마다 업그레이드가 일부 진행되어 게임의 난이도를 낮춰주는 '로그라이트' 시스템을 차용하기로 했다. '로그라이크' 그대로의 시스템은 유저에게 게임을 진행하며 얻는 성장감각을 주기 어렵고, 이는 시간이 지날수록 큰 피로감을 주기에 일부 코어 게이머들을 제외한 캐주얼 함을 지향하는 유저에게 어필하기 어렵다는 것이 이유이다. 또한 '소울라이크'를 차용하지 않은 이유는 게임 플레이에서 나오는 즐거움을 최소화하고 싶었기 때문이다. 개발 동기에서 지적한 것과 같이 퀄리티만 높고 재미있지 않은 게임이 되는 것을 원하지 않았는데, 게임 플레이 경험에서 나오는 즐거움에만 과하게 의존하는 것은 이런 예시와 공유하는 일면이 있다고 생각한다. 이런 이유로 '로그라이트' 시스템이 본 프로젝트의 핵심 시스템 중 하나가 되었다.

2.2 관련 예시

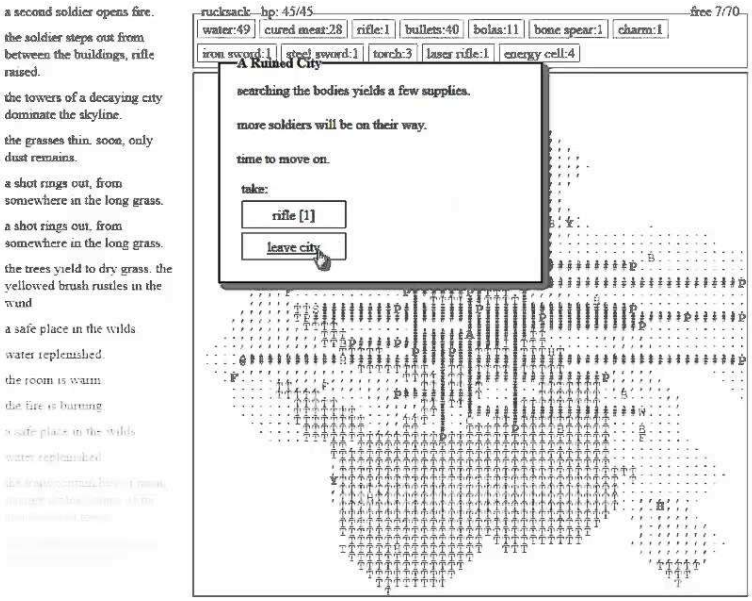
'Slay the Spire'의 유물 시스템은 전투에 직접적인 영향을 주며, 공격과 방어에 장비로써 매번 작동하여 전황을 바꾸어 준다. 이런 유물들은 다양한 종류가 존재하여 게임을 진행하며 랜덤하게 수집할 수 있다. 해당 시스템은 본 프로젝트의 장비 혹은 유물 시스템에 차용하여 적용할 예정이다.

'A Dark Room'의 UI 및 진행 방식은 대부분 텍스트 혹은 ASCII 코드를 이용한 그래픽 요소로 유저에게 게임 플레이 경험을 전달한다. 이런 방식은 시각적인 요소를 최소화하는 방법으로 플레이어의 상상력을 자극하는 소설책과 비슷한 효과를 갖는다. 이는 때론 영화나 동영상보다 뛰어난 흡입력을 갖는 전개 방식으로, 해당 게임에선 텍스트의 효과를 십분 발휘하여 높은 몰입도를 선

사하는 경험을 주었다. 이는 본 프로젝트의 '최소화된 퀄리티 자극 요소'라는 특징과 비슷하여 게임의 전개 방식에 차용하게 되었다.

'Princess Maker 2'의 육성 시스템은 턴제 시스템을 일부 차용하여, 유저에게 매달의 시작 마다 캐릭터의 육성 스케줄을 선택할 수 있게 하여 진행 차도에 따라 특정 이벤트의 결과값을 다르게 하는 형식으로 성장이 쉽게 체감되도록 하였다. 본 프로젝트에선 유저에게 매 턴 마다 전투, 육성 등의 선택지를 주어 원하는 방향으로 캐릭터의 성장이 가능하게 하고, 마주하는 이벤트에서 특정 방향으로의 성장을 돕거나 막는 형식의 진행을 통해 해당 시스템을 일부 차용하게 되었다.

[그림- 2.1] 'Slay the Spire'의 유물 시스템



[그림- 2.2] 'A Dark Room'의 UI 및 진행 방식

[그림- 2.3] 'Princess Maker 2'의 진행 구조

3. 요구명세

3.1 User Requirements

1. 유저 캐릭터
   A. 유저는 자신의 캐릭터의 특성을 뜻에 맞게 육성할 수 있어야 한다.
   B. 각 특성은 개별적으로 담당하는 효과가 존재해야 한다.
2. 이벤트
   A. 이벤트의 종류는 시작 시 이벤트, 전투 이벤트, 육성 이벤트의 세가지로, 목표 시점에 맞게 나와야 한다.
   B. 이벤트의 효과는 단순 수치 증가, 재화 추가 등 다양하게 존재해야 한다.
   C. 이벤트는 랜덤하게 표시되되, 유저의 수치에 기반해 표시되는 이벤트의 성향에 영향을 줄 수 있어야 한다.
3. 육성
   A. 유저는 자신이 원하는 방향으로 캐릭터를 육성할 수 있어야 한다.
   B. 6가지의 정해진 수치가 존재하며, 각 수치들은 게임에 있어 다른 방향으로 의미를 가져야 한다.
4. 전투
   A. 전투 과정은 텍스트로 이루어지며, 확률에 기반한 운 시스템으로 진행되어야 한다.
   B. 플레이어의 수치에 따라 해당 확률이 변동하여 전투의 승리를 유도할 수 있도록 설계되어야 한다.
   C. 전투의 결과에 맞는 보상 혹은 불이익을 가져야 한다.
   D. 전투의 텍스트는 전투 과정을 효과적으로 표현해야 한다.
5. 장비
   A. 장비는 캐릭터의 세부 수치에 따라 전투에 직접적인 영향을 주어야 한다.
   B. 각 장비는 적용하는 캐릭터의 수치의 계산법을 다르게 하여 전투의 방향에 영향을 주어야 한다.
6. 적
   A. 적은 매번 랜덤하게 선정되어 유저에게 새로운 경험을 주어야 한다.
   B. 유저의 캐릭터와 비슷한 난이도로 정해져 승리와 패배를 공평하게 전달해야 한다.
7. 컨셉
   A. 유저가 몰입할 수 있는 일관적이고 흥미로운 배경과 스토리를 가지고 있어야

한다.

3.2 System Requirements

1. System Structure
    A. Scene의 전환으로 게임의 진행이 이루어짐.
    B. 각 Scene들은 핵심 상황별로 분류됨.
    C. Scene의 기능을 효과적으로 실행하기 위한 Object 설계가 필요함.
        1. 각 Scene의 시각적 변화와 내부적 이벤트를 컨트롤하는 Object가 구분됨.
        2. 여러 Scene에 공통으로 활용되는 기능들은 컨셉에 따라 분류
2. MainMenu
    A. New Game, Load Game, Quit 옵션이 존재하고 각각 의도에 따라 작동함.
    B. New Game의 경우 기존 데이터를 삭제하고 새로운 게임을 처음부터 시작함.
    C. Load Game의 경우 기존 데이터를 불러와 게임을 종료한 시점부터 이어서 할 수 있음. 단, 미리 지정된 이벤트, 적 등의 옵션은 그대로 유지되어야 함.
    D. Quit의 경우 게임을 종료해야 함.
3. Event
    A. 각 상황(StartEvent, TrainingEvent, BattleEvent)별 다른 이벤트를 출력하기 위해 분류되어 있어야 한다.
    B. 이벤트는 랜덤하게 출력되지만, User의 medStat에 따라 Good, Normal, Bad 값을 출력하는 Karma 시스템을 통해 그 방향성이 일부 결정될 수 있음.
    C. User은 랜덤하게 결정된 Event를 '세이브-로드' 등의 방식으로 인위적으로 바꿀 수 없음.
    D. Event 내부의 Text, 선택지 들은 User의 Stat에 따라 추가로 등장하며 확인 혹은 선택 가능함.
4. Battle
    A. Battle Log는 화면에 표시되어 전투의 과정을 쉽게 알 수 있도록 표현되고, 공개 정도는 User의 Stat에 따라 바뀜.
    B. Battle은 특이한 경우를 제외, 100% 확률의 승리 혹은 패배가 이루어지지 않음.
    C. Battle의 결과는 출력 전에 저장되어 '세이브-로드'등의 방식으로 임의적으로 바꿀 수 없음.
    D. User은 랜덤 한 적을 만나되, 확인한 적은 '세이브-로드'등의 방식으로 인위적으로 바꿀 수 없음.
5. Training

A. User은 원하는 Stat을 원하는 타이밍에 훈련할 수 있음.
B. Training을 통한 Stat의 증감 폭은 staStat 값에 따라 내부 계산법을 반영하여 변동됨.

6. Enemy
   A. 적 개체들은 User과 동일한 Stat과 Weapon 시스템을 가진 채로 설계됨.
   B. User의 전투 방식과 동일하게 적 개체들의 Stat과 Weapon 시스템에 따라 Battle에서의 데미지를 계산함.
   C. 적 개체들은 각각 스킬들을 가져 Battle 스테이지에서 미리 지정된 확률에 따라 발동되어 추가 데미지를 계산함.
7. Weapon System
   A. 무기는 User과 Enemy가 공통으로 사용할 수 있어, Stat의 수치에 따라 데미지의 크기와 횟수 등의 세부 사항이 결정된다.
   B. 무기는 세부적인 수치 차이, 특징 점 차이에 따라 분류된다.

4. 설계 및 구현

4.1 프로젝트 일정

구분	세부 사항	프로젝트 기간				
		1학기 1 분기	1학기 2 분기	방학 기간	2학기 1 분기	2학기 2 분기
계획	프로젝트 계획 작성					
	게임 기획					
스터디	Unity 툴 스터디					
개발	기초 Structure					
	세부 Scene					
	이외 세부 요소					
추가 요소	추가적 스토리 및 Event					

문서 작성	개발 과정 정리					
	보고서 작성					

[표- 4.1] 프로젝트 일정

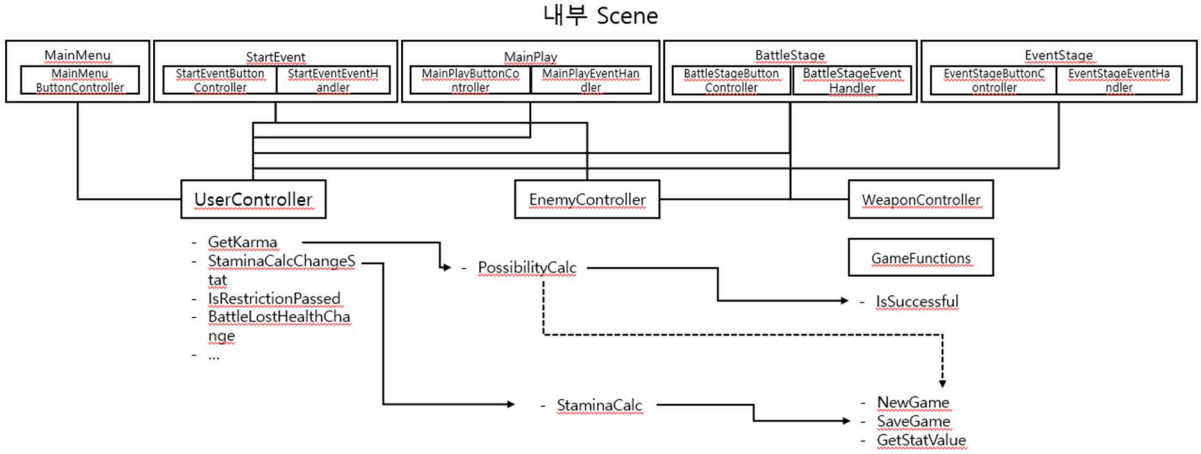
4.2 개발 환경

구분	세부 사항
OS	Windows 10
개발 언어	C#
게임 엔진	Unity Game Engine
아트(스프라이트)	Aseprite

[표- 4.2] 개발 환경

4.3 설계

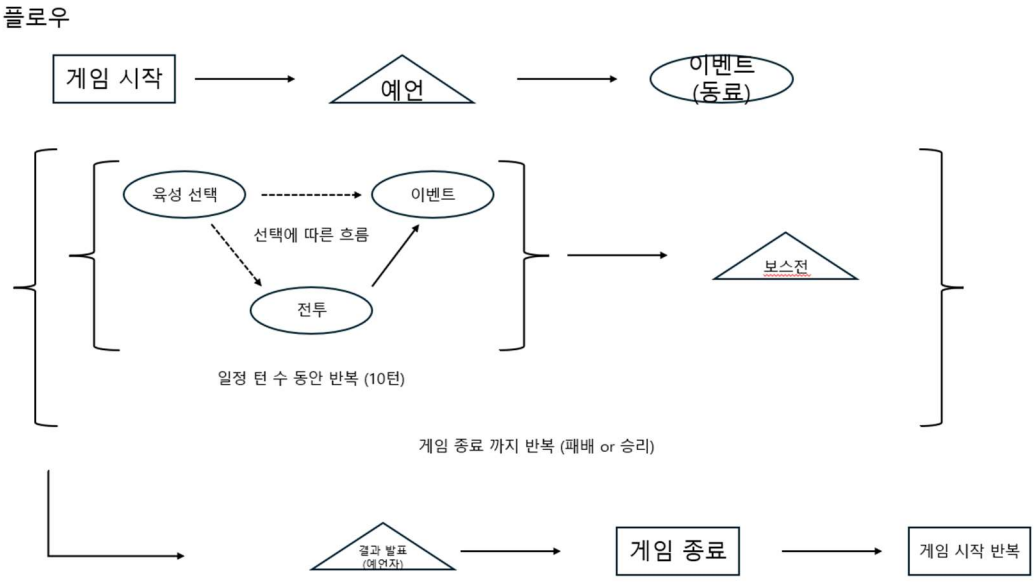
4.3.1 System Structure



[그림- 4.1]Scene 및 내부 Object 구조

다양한 Scene의 전환으로 게임의 진행이 이루어지며, 해당 Scene들은 각자 이벤트를 조절하는 EventHandler와 시각적 변화를 조절하는 ButtonController를 갖는다. 이외에 UserController, EnemyController 등의 공유되는 Object로 게임 전체에 저장되는 변화를 조절한다.

4.3.2 게임 동작 흐름

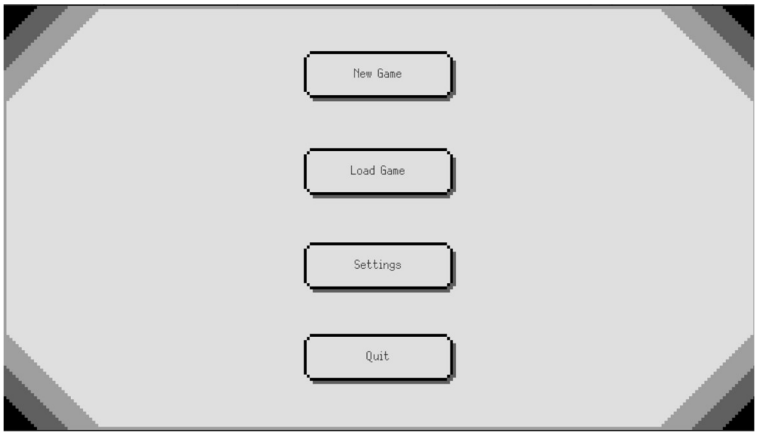


[그림- 4.2] 게임 동작 흐름도

게임을 시작하게 되면 첫 방향성을 정하는 StartEvent 단계를 거쳐 MainPlay에 이른다. MainPlay는 Training or Battle 이벤트 후에 Event를 실행 한 뒤 다시 MainPlay로 돌아가는 형식으로 일정 턴 수 동안 진행하게 되며, 특정 턴 수가 끝나면 보스전을 치룬다. 패배하게 되면 메인 화면으로 돌아가 다시 게임 시작이 가능하다.

5. 실행 결과

5.1 실행 화면



[그림- 5.1] MainMenu Scene

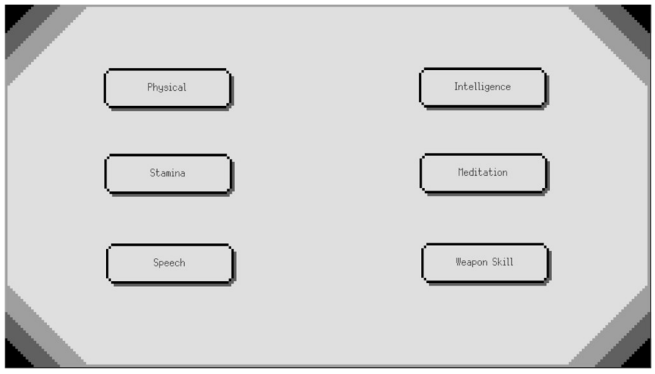
- MainMenu에는 New Game, Load Game, Settings, Quit등의 선택지가 있다.
- New Game등 파일에 변화를 주는 선택지는 실행 되기 전 팝업 창을 띄워 해당 사항이 유저의 결정이 맞는지를 묻는다.
- Load Game 선택 시 마지막으로 저장된 위치에서 게임을 이어 할 수 있다. Battle 승패 여부, 확정 Event 종류 등의 무작위 적인 아이템은 Load Game을 통해 재선정이 불가능하도록 설계.

[그림- 5.2] StartEvent Scene

- 랜덤으로 선택되는 Dialogue 와 Option을 통해 새로운 선택지를 제공하여 게임의 초반 방향성을 능동적으로 결정할 수 있도록 한다.
- 새로운 이벤트가 추가되기 쉽도록 Json형식을 활용해 이벤트를 제작한다.

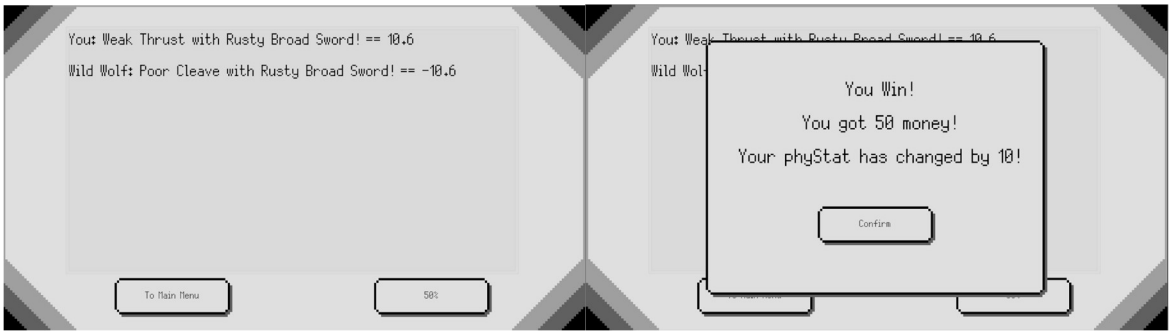
[그림- 5.3] MainPlay First Scene

- StartEvent 이후에 전개되는 MainPlay Scene.
- 플레이어는 Training과 Battle 중 다음 행동을 결정 할 수 있다.



[그림- 5.4] Mainplay Training Scene

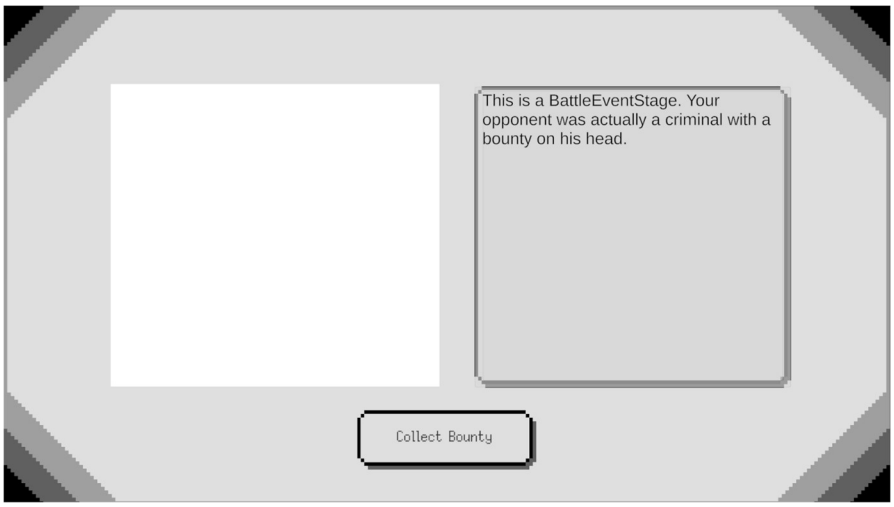
- Training 선택 시 등장하는 선택화면.
- PhyStat, StaStat, SpeStat, IntStat, MedStat, WeaStat 중 성장시킬 특성을 결정한다.
- 선택된 Stat은 플레이어의 StaStat 값에 따라 일정 수준 성장한다.
- 이후 EventStage로 연결된다.



[그림- 5.5,6] BattleStage Scene

- MainPlay에서 Battle을 선택 시 등장하는 전투화면.
- 무작위로 선정된 Enemy 개체와 전투를 진행. 각 개체의 Stat과 Skill(Enemy 한정)을 기반으로 전투를 진행한다.
- 첫 승리 확률은 50%에서 시작하여 각자의 공격별로 해당 확률에 영향을 주며 최종 확률 결정.
- 최종 확률 확정 시 해당 확률을 통해 승리 여부 결정. 승리 시 승리 보상을, 패배 시 패배 벌칙을 수행.

- 이후 EventStage로 연결된다.

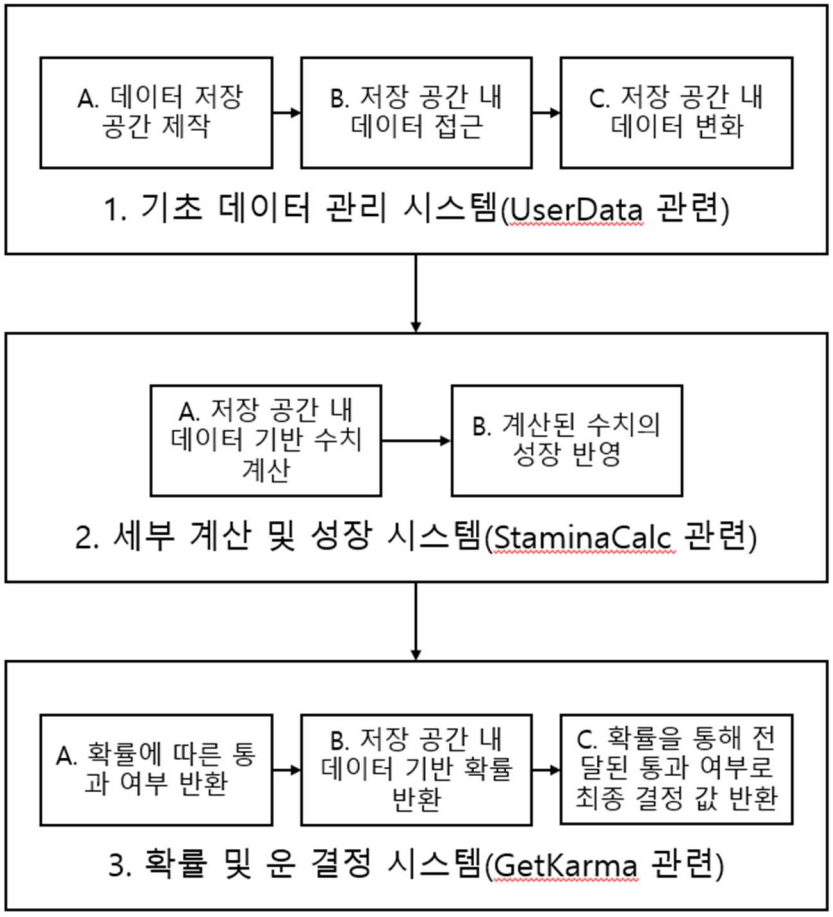


[그림- 5.7] EventStage Scene

- Training 혹은 Battle 이후에 이어지는 EventStage. 이전 단계에 따라 EventStage의 종류가 변화한다.
- 또한 플레이어의 Stat에 따라 정해지는 Karma 값에 따라 긍정적 혹은 부정적 성향의 Event가 출력된다.
- Option 선택을 통해 특정한 효과를 발동시킨 뒤 Event는 종료된다. 후에 MainPlay로 다시 연결되며 게임플레이 반복.

5.2 Test-Driven Development

5.2.1 Test-Driven Development 순서도



[그림- 5.8] TDD 진행 구조도

- 개발 과정에 도입된 TDD의 구조도.
- 1, 2, 3 블록, A ,B ,C 단계 순서로 각각 세부 사항에 대한 테스트 제작, 후 활용하여 개발 진행
- 해당 방식을 통해 각 단계별 Robust함, 객체지향적 코드를 제작, 차후 디버깅과 기능 추가를 원활하게 할 수 있었다.

5.2.2 Test-Driven Development 세부사항

1. **기초 데이터 관리 시스템**
   - **A. (+B) 데이터 공간 제작 및 접근 테스트**

NewGame	새로운 UserData를 만든다
GetStatValue	UserData 내 특정 Stat 값을 반환한다

단계별 테스트	1. 새로운 UserData를 생성한다. 2. 해당 UserData에 초기값을 설정한다. 2-1. 해당 초기값들을 체크한다.
테스트 단계 1	`userController.NewGame();` - 데이터를 저장할 새로운 UserData 생성
테스트 단계 2	- 정상적인 초기값(전체 StatValue 200 고정)으로 초기화되었는지 확인 - 결과 phyStat normally initialized intStat normally initialized staStat normally initialized speStat normally initialized medStat normally initialized - weaStat normally initialized

[표- 5.1] 데이터 공간 제작 및 접근 테스트

C. 저장 공간 내 데이터 조작 테스트

ChangeStat	UserData내 특정 Stat값을 변화시킨다.
단계별 테스트	1. 특정 Stat값을 일정 수준 증가시킨다. 2. 특정 Stat값을 일정 수준 감소시킨다.
테스트 단계 1	- phyStat을 50 증가시켜 증가 기능의 테스트 - 결과 - phyStat increased successfully. Current value: 250
테스트 단계 2	- intStat을 50 감소시켜 감소 기능의 테스트 - 결과 - intStat decreased successfully. Current value: 150

[표- 5.2] 저장 공간 내 데이터 조작 테스트

2. 세부 계산 및 성장 시스템

A. 저장 공간 내 데이터 기반 수치 계산

StaminaCalc	staStat 값에 따라 수치를 조절

단계별 테스트	1. staStat값이 일정 수치 이하일 때 전달받은 인자에 감소를 적용한다. 2. staStat값이 일정 수치 이상일 때 전달받은 인자에 증가를 적용한다.
테스트 단계 1	- 현재 Stat이 ReferenceStatValue의 절반일 때(Reference 값 미만인 경우)의 테스트. - 결과 - increasement calculated successfully. Expected value: 4 Result: 4 - 본래 증가값인 8 의 절반으로 계산
테스트 단계 2	- 현재 Stat이 ReferenceStatValue의 이상일 때 테스트. - 결과 - increasement calculated successfully. Expected value: 12 Result: 12
테스트 단계 3(예외 사항 추가 테스트)	- 현재 Stat이 ReferenceStatValue와 같을 때 테스트. - 결과 - Stamina is at reference value. Expected value: 8 Result: 8

[표- 5.3] 저장 공간 내 데이터 기반 수치 계산

B. 계산된 수치의 성장 반영

StaminaCalcChangeStat	Encapsulation을 위해 StaminaCalc의 직접적인 호출 없이 특정 Stat의 ID만 인자로 받아 해당 Stat을 계산된 증감폭을 반영하여 성장
단계별 테스트	1. 함수 호출 시 해당 stat이 성장한다. 2. Stat의 성장폭이 staminaCalc를 반영한 수치이다.
테스트 단계 1	- 함수가 호출되었을 때, 지정된 stat이 실제로 증가하였는지의 테스트 - 결과 - intStat increased succesfully. Current value: 206
테스트 단계 2	- 기본 Stat으로 설정되었을 때, intStat을 인자로 전달하여 함수 호출 시 성장폭 반영의 테스트 - 결과 - Expected intStat value: 206 Result: 206 - 200인 staStat을 감안하여 기본 8의 증가값에서 감소된 6이 증가된

	모습

[표- 5.4] 계산된 수치의 성장 반영

3. 확률 및 운 결정 시스템

A. 확률에 따른 통과 여부 반환

IsSuccessful	확률을 전달받아 True False의 결과값을 반환
단계별 테스트	1. 0을 전달받았을 때 False를 반환한다. 2. 100을 전달받았을 때 True를 반환한다. 3. 50을 전달받았을 때 True False 반환 비율이 반영된다.
테스트 단계 1	- 0프로의 확률일 때 항상 False를 반환하는지의 테스트 - 결과 - 0% success rate. Result: false
테스트 단계 2	- 100프로의 확률일 때 항상 True를 반환하는지의 테스트 - 결과 - 100% success rate. Result: true
테스트 단계 3	- 50프로의 확률일 때 1000번의 iteration을 통하여 생성된 True False 비율이 50프로에 근사하는지의 테스트 - `float successRateAchieved = (float)successCount / itera` - 해당 코드를 통해 확률 계산 - 결과 - Success rate achieved: 50.4

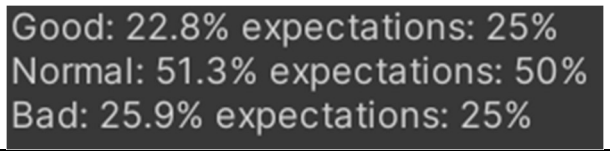
[표- 5.5] 확률에 따른 통과 여부 반환

B. 저장 공간 내 데이터 기반 확률 반환

PossibilityCalc	UserData내 특정 Stat의 값에 따라 확률을 계산하여 반환
단계별 테스트	1. Stat이 최대 일 때 최대 확률을 반환한다. 2. Stat이 최저 일 때 최저 확률을 반환한다. 3. 여러 Stat을 인자로 전달했을 때 해당 Stat을 기반으로 확률을 계산한다. 4. 기본 확률을 50%로 설정할 시 적용하여 계산한다. 5. Stat이 일정 수치 이하일 때 적용하여 확률을 반환한다. 6. Stat이 일정 수치 이상일 때 적용하여 확률을 반환한다.
테스트 단계 1	- 최대 Stat 값으로 설정했을 때 최대 확률 반환의 테스트 - 결과 - Expected: 80 Result: 80
테스트 단계 2	- 최저 Stat 값으로 설정했을 때 최저 확률 반환의 테스트 - 결과 - Expected: 9.999999 Result: 9.999999 - Float value를 다뤄 사소한 차이 존재
테스트 단계 3	- Stat 값이 기준 값 이상인 경우 기준 확률 이상 반환 여부의 테스트 - 결과 - Expected: 54.28571 Result: 54.28572 - Stat type을 speStat으로, startAtHalf를 true로 설정하여 여러 stat 인자의 전달, 기본 확률 50% 설정 시의 케이스도 테스트
테스트 단계 4	- Stat 값이 기준 값 이하인 경우 기준 확률 이하 반환 여부의 테스트 - 결과 - Expected: 15 Result: 15
테스트 단계 5(예외 사항 추가 테스트)	- Stat이 기준 값과 동일한 경우 기준 확률 반환 여부의 테스트 - 결과 - Expected: 20 Result: 20

[표- 5.6] 저장 공간 내 데이터 기반 확률 반환

C. 확률을 통해 전달된 통과 여부로 최종 결정 값 반환

GetKarma	2번에 걸친 통과 테스트를 통해 Good, Normal, Bad의 결과를 반환
단계별 테스트	1. 설정된 Stat에 따라 예상과 같은 Good, Normal, Bad의 결과값을 반환한다.
테스트 단계 1	- 최대값으로 Stat 설정 시 2번에 통과 계산에 따른 확률로 각각 25%, 50%, 25%의 결과 반환 여부의 테스트 - 결과  -

[표- 5.7] 확률을 통해 전달된 통과 여부로 최종 결정 값 반환

6. 결론 및 향후 계획

6.1 결론

플레이어가 자신만의 이야기를 만들어 나가고, 게임 자체의 배틀과 성장 시스템을 통해 변수를 창출해 매번 다른 플레이를 유도하는 등의 기능을 통해 즐거운 경험을 전달한다. 또한 이벤트와 적 등의 게임 컨셉에 익숙한 그리스-로마 신화를 차용하여 내러티브 경험도 전달하고자 했다.

6.2 추후 발전 방향

- 이벤트, 적, 아이템 등의 세부 사항이 부족하다. 다채로운 경험을 위해 가장 중요한 부분으로 최우선 수정 필요.
- 시각적 디테일이 부족하다. 기본적인 UI 개선과 더불어 이벤트 등에 활용될 이미지가 필요.
- 단조로운 게임 플레이의 개선이 필요하다. 새로운 메인 게임 요소 혹은 이펙트 추가 등으로 유저의 만족도 상승 필요.

7. 참고 문헌

[1] ‘Steam의 Slay the Spire’, https://store.steampowered.com/app/646570/Slay_the_Spire/?l=koreana

[2] ‘Unity Learn’, https://learn.unity.com/