졸업작품 개발 보고서

과목명: 게임제작프로젝트(2)

지도 교수: 서범주 교수님

프로젝트명: NOAH(노아)

작성자: 홍익대학교 세종캠퍼스 게임소프트웨어전공

B577007 김종혁

목차

1 서론

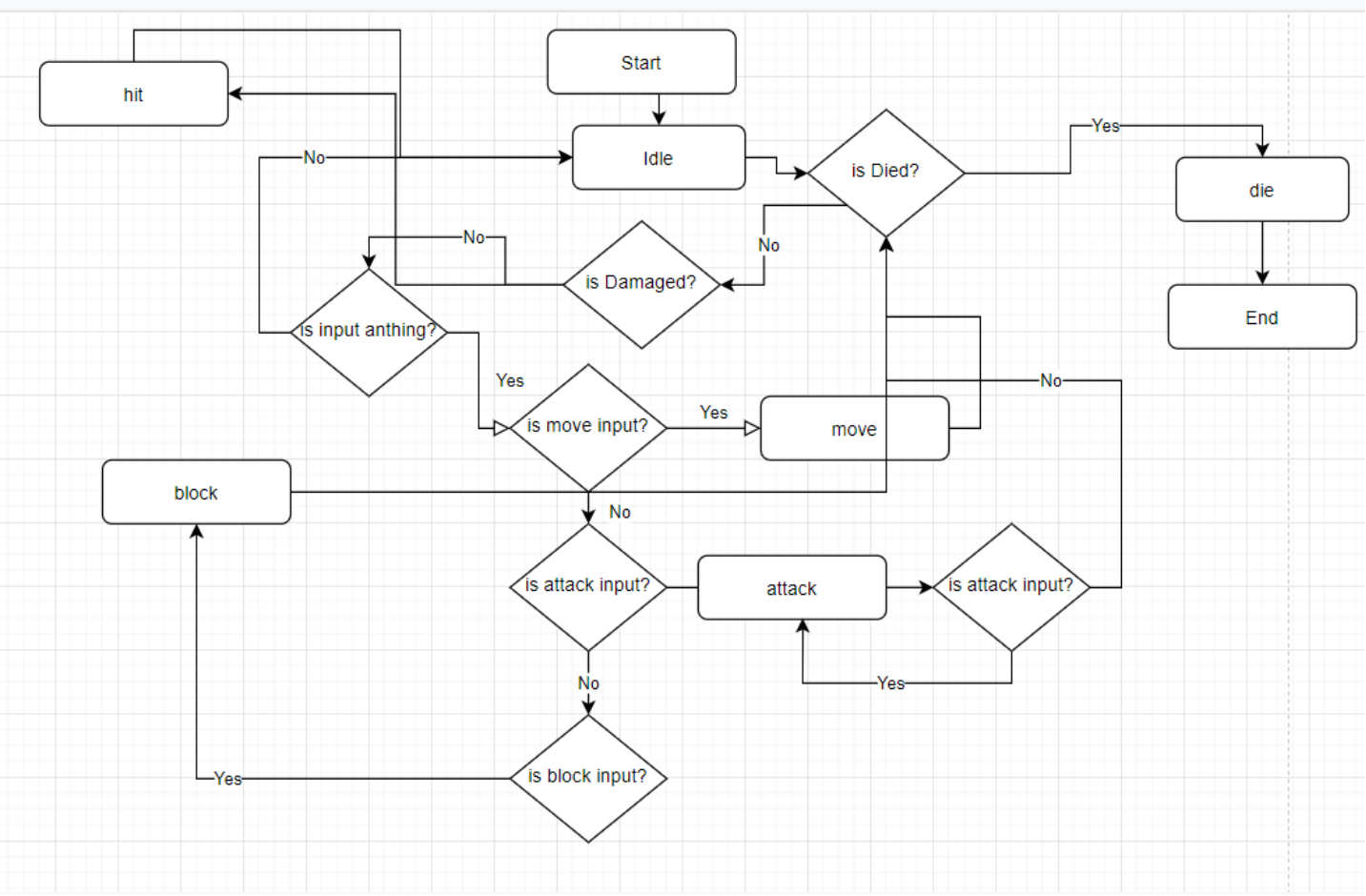
1.1 <개요>

- 장르: 3인칭 액션 RPG (3인칭 숄더뷰)
- 플랫폼: PC(Windows)
- 개발환경: Unity3D / C#
- 진행방식: 던전을 탐험하며 조우하는 몬스터와의 전투를 중심으로 전개. 최종적으로 마지막 스테이지에 있는 보스 몬스터를 처치하면 승리.
- 그래픽 리소스: 메인 – POLYGON – Dungeons Pack (Unity Asset Store)
  그 외 유니티 Asset Store 무료 리소스 사용
- 스토리: 사악한 몬스터들에게 납치된 공주를 구하러 가는 용사(플레이어)의 이야기

1.2 <구현중점 사항>

- 가비지 컬렉션(Garbage Collection)발생을 염두에 두고 Object Pooling을 통해 투사체, 이펙트, 파티클을 재사용. 동적 메모리 할당으로 인한 메모리 누수와 GC발생을 방지하고자 함
- Managers, Player 등 중복해서 존재해서는 안되는 인스턴스에는 싱글톤 (Singleton) 패턴을 적용해 관리하고자 함
- 상속을 통한 중복 코드 제거, 유닛 클래스들의 그룹화를 위해 추상클래스와, 인터페이스를 활용해 클래스를 설계하고자 함
- Update method호출 최소화를 위해 피격 처리와 같은 부분은 피격당한 유닛에게 DamageMessage를 보내 처리하는 등, event 기반으로 처리하고자 함

2 구성도

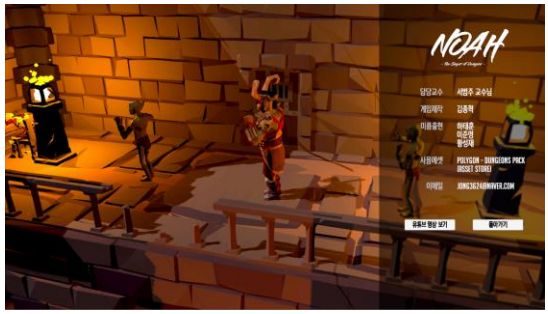


[그림1] 유닛(플레이어/적)에게 적용되는 플로우 차트

3 구현 내용

3.1<게임 씬(Scene)>

A. Title Scene: 새 게임 / 스테이지 선택 / 크레딧 보기 / 종료 기능 구현



[그림2] 타이틀(좌) / [그림3] 크레딧(우) 화면

B. Main Scene: 플레이어가 게임을 플레이하며, 미션을 진행하는 씬
전투/옵션/유저 인터페이스/대화/사운드 등 구현

C. Result Scene: 플레이어가 게임을 패배/승리 시 연출되는 씬
게임 재시작/타이틀 씬 이동 구현

3.2 <스테이지>

A. 튜토리얼 스테이지: 플레이어에게 게임의 스토리를 알려주고, 기본적인 상호작용/이동(걷기, 달리기)/공격/방어/포션 등의 조작법을 알려주는 스테이지 구현

B. 레벨 디자인 적용 스테이지: 스테이지별 레벨 디자인을 적용,
소수 몬스터 -> 소수 몬스터 + 함정 -> 다수 몬스터 -> 보스 몬스터가 등장하는 스테이지 구현

3.3 <유닛>

A. 클래스 설계: PlayerCharacter/Enemy 클래스는 Unit 추상 클래스를 부모클래스로 두며, 공격/방어/이동 등의 추상 메소드를 공유하도록 설계

- 유닛 피격 시 피격당한 유닛에게 Message를 보내 damageAmount, damager, attackType 등의 정보를 전달. 전달된 정보를 이용해 데미지 적용 및 hitParticle의 위치와 각도를 계산, 피가 분출되는 효과를 적용해 유닛 피격 구현
- 메시지를 이용해 이벤트 기반으로 데미지 처리 등을 구현. 이를 통해 불필요한 update method사용을 최소화하고 그에 따라 게임 속도를 향상하고자 함

B. 플레이어 캐릭터

- 공격
  - 4콤보로 공격할 수 있는 기본공격 구현
  - 달리기 중 공격할 수 있는 대쉬 공격 구현
  - 유닛 공격 시 Animation Event를 이용, 공경 애니메이션의 공격 동작이 시작/완료되는 시점에 beginAttack method를 호출해 공격 중, 공격완료 상태로 나눠 공격 로직 구현
- 방어: 방패를 들어 적의 공격을 막을 수 있는 방어기능 구현
  - 방어할 수 없는 공격도 존재, 이러한 공격은 방어 불가능하게 구현

- 이동
  - 걷기: 플레이어의 기본 이동 기능인 걷기 기능 구현
  - 달리기: 스테미너를 소모하며 빠르게 이동 가능한 기능인 달리기 기능 구현
- 스킬
  - 플레이어 주변 범위의 적들에게 광범위로 공격 가능한 스킬 구현
- 회피: 굴러서 적의 공격과 함정을 피할 수 있는 기능인 회피 구현
- 체력: 유닛의 사망 상태를 체크하는 자원
  - 체력이 0이 될 경우 유닛을 사망 상태로 변경, 들어오는 충돌처리를 무시하고 사망 애니메이션 재생/사라지는 효과를 적용해 구현
  - 포션을 이용해 회복 가능
- 스테미너: 달리기/방어 시 소모되는 자원
  - 스테미너가 0이 될 경우 달리기/방어가 불가능하게 구현
  - 플레이어 캐릭터가 달리기/방어 상태가 아닐 경우 회복

C. 적 캐릭터

- 플레이어 탐색/추적: 캐릭터 정면을 기준으로 시야각을 적용, 플레이어가 시야각에 들어왔을 경우 인지하며 플레이어를 쫓아오는 기능 구현
- 공격/방어: 플레이어를 공격하는 공격 기능과 플레이어의 공격을 막을 수 있는 방어 기능 구현
- 체력: 유닛의 사망 상태를 체크하는 자원.
  - 체력이 0이 될 경우 유닛을 사망 상태로 변경, 들어오는 충돌처리를 무시하고 사망 애니메이션 재생/사라지는 효과를 적용해 구현
  - 유닛이 전투 상태가 아닐 경우 최대 체력(max hp)으로 회복되게 구현

D. NPC

- 플레이어와 상호작용/대화가 가능한 NPC 구현
- NPC는 적 유닛으로부터 인지되지 않고, 플레이어 유닛에게 피해를 입지 않게 구현

3.4 <장비>

A. 무기: 유닛(플레이어/적)들이 이용할 수 있는 장비로 무기마다 각기 다른 데미지와 사정거리를 지니도록 구현

B. 방패: 유닛(플레이어/적)들이 이용할 수 있는 장비로 방패마다 각기 다른 방어율을 지니도록 구현 (단, 플레이어의 방패는 방어율이 존재하지 않는다)

3.5 <이펙트>

A. 클래스 설계: 오브젝트를 관리하는 ObjectManager 클래스를 구현, Object Pooling을 통해 이펙트(파티클)와 투사체를 관리

Object Poolint 구현 방식: 파티클과 투사체(이하 오브젝트)를 인스턴스 생성할 때, 리스트에 등록.
파티클 혹은 투사체를 사용할 때 리스트에서 사용하고 있지 않는 오브젝트를 확인, 존재하면 반환해 사용하게 한다.
만약 이미 생성해 놓은 오브젝트들이 모두 사용중인 상태라면 그 즉시 새로운 인스턴스를 만들어 반환해 사용한다.

B. 특수효과

- 유닛 피격 효과: 유닛들이 상대 유닛을 공격했을 때, 생성되는 이펙트
- 충돌(Crash) 효과: 유닛들이 단단한 물체(ex:벽, 바위)들을 공격했을 때, 생성되는 이펙트

C. 씬 페이드 인/아웃 (Scene Fade In/Out)

- 씬 전환 시 화면이 어두워지는 Fade In효과를 적용하고 그 후 씬을 전환, 다시 Fade Out 효과를 적용해 화면을 보이도록 구현
- Fade In/Out 효과를 이용해 화면 전환이 자연스럽게 보이도록 하고, 게임의 몰입도를 올리기 위해 구현함

3.6 <미션>

A. 플레이어 미션: 스테이지 별로 플레이에게 미션이 주어지고, 미션을 클리어 해야만 다음 스테이지로 이동할 수 있게 구현

미션을 클리어하는 과정에서 플레이어가 게임을 학습할 수 있게 레벨디자인을 적용해 구현

B. UI: 화면 우측 상단에 미션의 제목과 내용을 표기함

3.7 <대화/대사>

A. 대화: 플레이어가 NPC/오브젝트와 상호작용하며 이야기를 나누는 기능인 대화 구현

B. 대사 파일: json파일로 작성, parsing/load를 통해 사용할 수 있도록 구현

3.8 <UI>

A. 체력 바/스테미너 바: 화면 중앙 하단 위치. 플레이어의 남은 체력과 남은 스테미너를 표시

B. 스킬 아이콘: 화면 우측 하단에 위치. 플레이어 스킬의 쿨타임을 Fill 방식으로(시계 방향으로 흑백처리 되며) 표시

C. 포션 아이콘: 화면 좌측 하단에 위치. 포션의 남은 쿨타임을 Fill 방식으로 표시

D. 미션 박스: 우측 최상단에 위치. 현재 미션의 제목과 내용을 표시. 미션이 생길 경우 좌측으로 등장. 없을 경우는 우측으로 사라지게 구현

E. 텍스트 박스: 화면 중앙 상단에 위치. 현재 대사를 출력

[그림4] UI 참조 이미지

3.9 <카메라>

A. 궤적 카메라: 기본 세팅이 되는 카메라. 구형태의 궤적을 따라 플레이어 주변에 위치. 마우스의 움직임에 따라 카메라 위치가 바뀌게 구현.

B. 타겟팅 카메라: 플레이어가 적을 타겟팅 했을 때 사용되는 카메라. 타겟팅 된 적을 목표로 향함

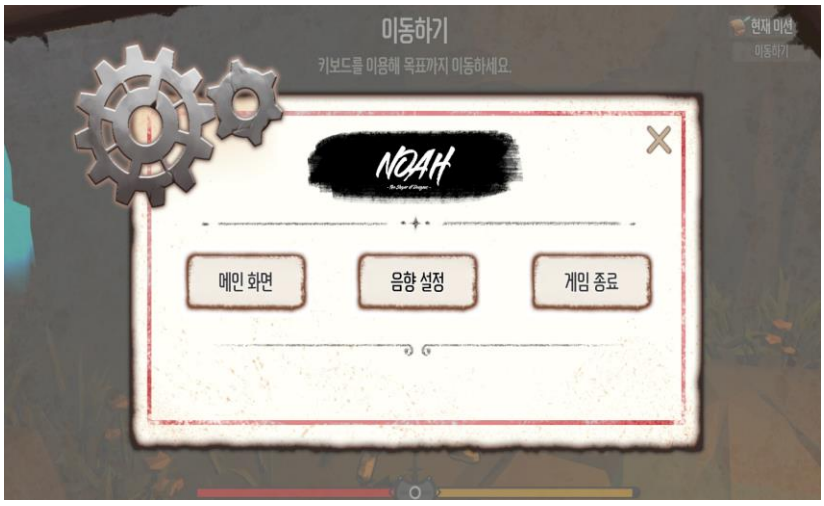
C. 씬 연출 카메라: 인트로/엔딩 혹은 특수 이벤트의 경우에 사용되는 카메라. Unity Timeline을 이용 연출을 중점적으로 구현

- 씬 연출 카메라를 특수 이벤트 이용, 플레이어에게 몰입감과 긴장감을 주기 위해 구현함

4 결과

4.1 <플레이 화면>

[그림5] 전투 화면(스테이지1)



[그림6] 옵션화면

[그림7] 전투 화면(스테이지2)

4.2 <애로 사항>

A. 유닛 공격 시 한 번에 데미지가 여러 번 적용되는 문제

- 문제점: 유닛이 공격 시 공격 한 번에 데미지가 여러 번 적용돼 데미지가 비정상적으로 들어가는 문제발생
- 해결방안: 이를 해결하기 위해 데미지가 적용된 오브젝트는 list에 add시키고 체크해 데미지가 여러 번 적용되지 않게 해결
- 결과: 공격 한 번에 데미지가 한 번만 적용되도록 해결

B. 공격 적용 시 생성되는 피의 각도/위치 문제

- 문제점: 공격 시 생성되는 피가 분출되는 각도와 위치가 일정하게 나오게 됨. 그로 인해 유닛의 공격 방향에 따라 피가 튀기지 않고, 위치가 부정확함
- 해결방안: 공격 시 DamageMessage에 공격이 적용된 위치와 방향을 추가적으로 전송. 그 값을 이용 계산해 피가 튀어야 되는 방향과 위치를 계산함
- 결과: 정상적인 위치와 방향으로 피가 튀도록 해결

C. 불필요 유닛

- 문제점: 현재 스테이지에 당장 사용되지 않는 유닛들이 배치되어 게임속도를 저하하는 문제발생
- 해결방안: 현재 스테이지에 사용되지 않는 유닛들을 active를 false로 설정, 필요한 경우에만 true로 변경시켜 사용하게 함 (ex: 플레이어가 일정 미션을 도달, 다음 구역으로 이동할 때, 유닛들을 사용가능한 상태로 변경)

- 결과: 게임속도가 향상됨 (단, 이는 본인의 체감으로 느껴지는 속도로 정확한 속도 향상은 측정이 필요함)

4.3 <느낀점>

구현 전에 클래스 설계 단계의 중요성을 느꼈다. 클래스와 프로그램 흐름을 정확하게 설계하지 않고 구현한 경우, 나중에 오히려 시간이 많이 소요되게 됐다. 코드가 완벽히 설계되지 않은 부분은 코드 자체가 다소 난잡하게 구현됐다. 따라서 코드를 구현하기전에 설계 단계에 더욱 집중해야 한다고 느꼈다.

또한, 프로그래밍뿐만 아니라 오브젝트 배치, 애니메이션, UI, 연출 등 여러 가지를 다루면서 그에 대한 지식을 얻을 수 있었다. 프로그래밍 실력을 갈고 닦음과 더불어 툴 사용에 대한 지식에 대해서도 숙련도를 올려야 한다는 점을 느낄 수 있었다.

4.4 <기타사항>

A. 게임영상 Youtube 주소
https://www.youtube.com/watch?v=ER-xdzH7Vu8&t=142s