**최규화**

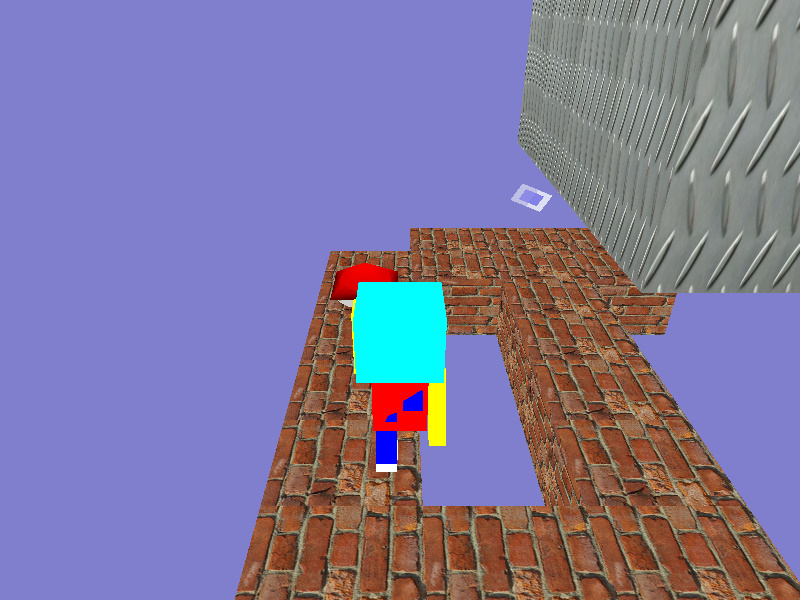
**[클라이언트]**

**[1].Win32 API 및 C++을 이용한 2D 메이플스토리 모작**

2학년 1학기 때 Win32 API 및 C++을 이용하여 메이플스토리 모작을 만들었습니다. 게임은 총 3개의 스테이지로 구성되어 있으며 1, 2 스테이지는 일반 몬스터가 나오고 마지막 스테이지에서는 보스 몬스터가 나오며 이들을 모두 처치하면 게임을 클리어하게 됩니다. 무기로서 표창을 4종류 사용할 수 있고 스킬을 4종류 구현하였습니다. 당시 Win32 API를 처음 배웠을 때라 주기적인 동작을 타이머 API를 사용해서 프레임이 끊기는 문제점이 있었습니다. 이 외에는 캐릭터 모션, 공격, 스킬, 몬스터 NPC의 AI(이동, 공격, 스킬), 화면 이동 및 스테이지 이동, UI, BGM(배경음악) 등 원하는 대로 잘 구현되었습니다. 당시 학교에 입학한 후 첫 게임 작품으로서 재미있게 만들었었고 학내에서 진행되는 과제전에 출품하기도 하는 등 많은 경험을 느낄 수 있게 만들어준 작품입니다.

**[2]. Python 을 이용한 2D 러닝 + 리듬게임 제작**

2학년 2학기 때 Python 을 이용하여 제작한 2D 러닝 + 리듬게임 입니다. 일반적인 러닝게임을 만드는 것이 지루하다고 생각되어 간단한 리듬게임 요소를 첨가한 게임입니다. 캐릭터는 스테이지를 이동하며 코인을 먹어 점수를 올리고 장애물인 몬스터를 피하며 이동하면서 상단에 나오는 알파벳에 맞는 키를 눌러 추가 점수를 획득합니다. 스테이지 끝에 도달하거나 캐릭터가 물에 빠져서 죽으면 게임랭킹을 보여주는 랭킹화면이 띄워지며 게임이 종료됩니다. 기존의 C++ 을 이용하여 게임을 만든 것 보다 스크립트 언어인 Python과 추가 모듈을 통해서 게임을 제작하여 훨씬 수월하게 2d 게임을 제작할 수 있었습니다. 예로 Windows API와 비교해서는 PNG 이미지 로드나 게임 로직에서 Loop작업의 프레임 조절, 씬 변경 등이 보다 수월했습니다.

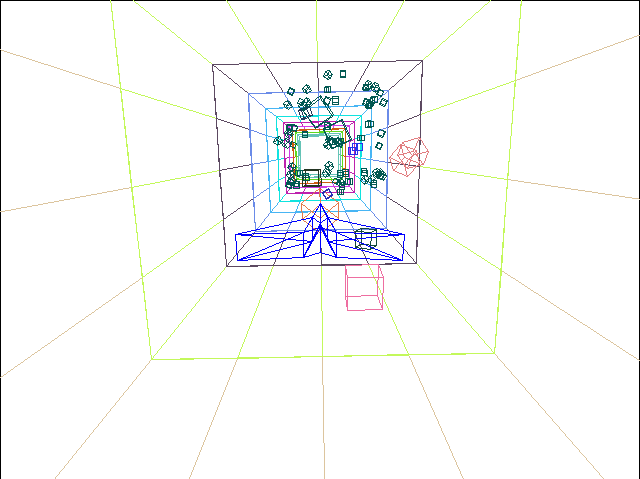
**[3]. OpenGL을 사용한 3D 게임 제작**

2학년 2학기 때 처음으로 만들게 된 3D 게임입니다. OpenGL 과 C++을 이용하여 제작하였습니다. 슈퍼마리오 게임을 모티브로 하고있으며, 기존 게임과는 다르게 2D 차원에서 3D 차원으로 변환이 가능합니다. 플레이어가 스테이지를 이동하다가 2차원시점에서는 보이지 않아 이동하지 못하는 길을 3차원시점으로 바꾸면 숨겨진 길이 나타나 이동이 가능합니다. 플레이어는 이동하면서 나오는 버섯 몬스터를 밟아 2단 점프가 가능하며 스테이지 끝에 존재하는 포탈을 통해서 다른 스테이지로 이동할 수 있습니다. 이 프로젝트는 팀 프로젝트로 진행되었으며 플레이어 캐릭터, 버섯 몬스터, 지형, 파티클, 포탈 오브젝트를 구현하였으며 캐릭터 스프라이트 조명효과와 씬, 시점 변환을 제작하였습니다. OpenGL에서 제공하는 함수들을 이용하여 3D 기하 오브젝트 제작, 오브젝트 간 충돌처리, 카메라 시점처리(원근 투영 및 직교 투영), 벡터를 이용한 플레이어 이동 및 회전, 조명 효과, 3D 오브젝트를 이용한 파티클 처리 등 3D 게임제작에 필요한 기초적인 지식들을 배울 수 있었습니다.

**[4]. OpenGL을 사용한 2D 게임 제작(OpenGL 3.3 이용)**

****

[3]에서 제작한 OpenGL은 고정형 파이프라인을 사용하여 제작한 3D 게임이지만 [4]번에서 제작한 게임은 고정형이 아닌 OpenGL 3.3 버전을 사용하여 만든 2D 게임이다. 3학년 2학기 때 게임소프트웨어 공학이라는 수업에서 진행했던 텀 프로젝트로 기획 단계부터 시작하여 프레임워크 제작 및 게임의 전반적인 내용을 모두 혼자 구현하였습니다. Super Cell의 클래시로얄이라는 게임을 모티브로 하여 제작하였습니다. 일정 시간마다 적기지와 아군기지에서 몬스터 NPC가 등장하여 자신의 위치에서 가장 가까운 위치에 있는 적의 기지로 이동하여 공격한다. 또한 각 기지에서 일정 주기마다 파티클 오브젝트가 발사되어 상대편의 기지에 데미지를 입힙니다. 기상 효과인 눈이 내리는 효과 또한 파티클 효과를 이용하여 구현된 것입니다.

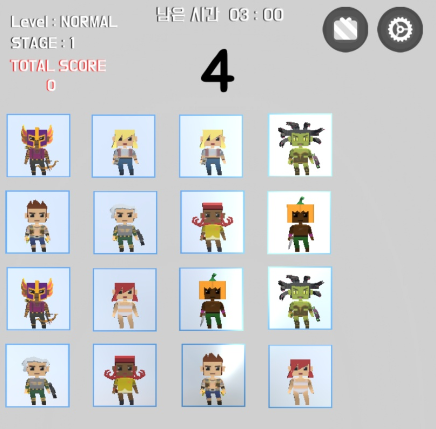
**[5]. DirectX 12을 사용한 기초적인 3D 게임 제작 지식 습득**

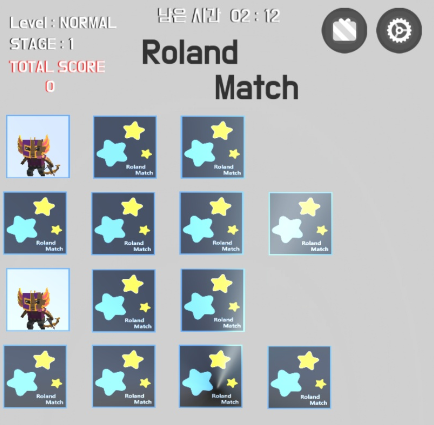
****

3학년 1학기 때 3D 게임프로그래밍 수업을 들으면서 DirectX12를 이용한 3D게임 제작에 대한 기초적인 지식을 습득 할 수 있었습니다. Direct3D 12 장치 생성부터, CPU와 GPU간 명령 대기열을 통한 데이터 업로드 및 업로드 된 데이터 사용. SwapChain을 이용한 전면 버퍼와 후면 버퍼 교환, 서술자 힙 생성 및 렌더 타겟 뷰, 깊이 스텐실 뷰 생성, 뷰 포트 설정, 파이프 라인 단계, 셰이더에 대한 지식, 조명, 텍스쳐 등 DirectX12이라는 그래픽라이브러리의 사용법에 대해서 전반적으로 배웠습니다. 실습 시간 중, 3D 정육면체 오브젝트 생성, 조명효과 구현, 클라이언트 프로그램의 부하를 줄이기 위한 객체 인스턴싱, Terrain 구현, 오브젝트에 텍스쳐 입히기, 오브젝트간 충돌처리(OBB 이용), 셰이더 작업 등을 하였습니다.

DirectX12 자체가 이론적으로 너무 어렵기 때문에 해당 그래픽라이브러리를 이용하여 게임제작을 하지는 못하였습니다.

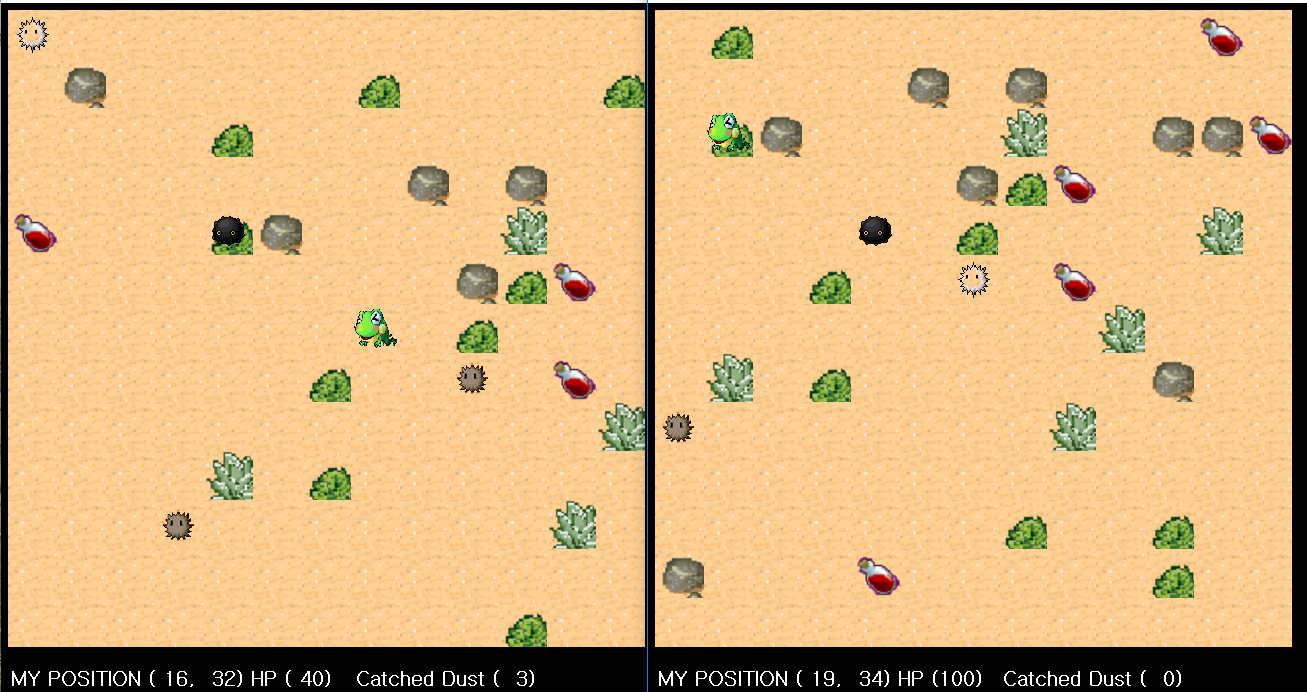
**[6]. Unity3D 엔진을 이용한 짝 맞추기 게임 제작**

****

****

해당 게임은 3학년 1학기 게임엔진 수업과 인터페이스 수업을 통해서 배운 Unity3D 엔진을 이용하여 8주간 진행 된 현장실습 기관에서 제작한 3D모델을 사용한 짝 맞추기 게임입니다. 게임 진행은 타이틀 화면에서 난이도(EASY, NORMAL, HARD)를 선택하고 각 난이도에 맞게 다양한 캐릭터가 표시된 카드들이 배치됩니다. 제한 시간 안에 같은 캐릭터 짝을 모두 맞추면 해당 스테이지에서 획득한 스코어를 보여주며 다음 스테이지로 이동합니다. 스테이지는 각 난이도 마다 50개이며 총 150개의 스테이지로 이루어져 있습니다.

**[서버]**

**[1]. IOCP와 멀티 스레드를 이용한 서버 제작**

4학년 1학기에 제작한 IOCP와 멀티 스레드를 이용한 서버 게임 프로그램입니다. IOCP를 서버모델로 사용하여 다중 접속이 가능합니다. 해당 게임은 사막에 떠다니는 먼지를 플레이어가 먹는다는 컨셉으로 제작되었으며, 사막에 떠다니는 폭발하는 먼지 몬스터 NPC(흰색 몹)를 피해서 일반 먼지 몬스터 NPC(갈색 몹)를 먹고 자신의 스코어를 올리는 게임입니다. 맵은 200 X 200 으로 제작되었으며, 사막 맵에는 통과가능한 지형(풀숲)과 통과하지 못하는 지형(바위, 선인장)이 존재합니다. 맵에 존재하는 빨간 물약을 먹으면 플레이어는 체력을 회복합니다. 해당 게임에 구현된 내용은 서버가 IOCP 소켓모델을 통해서 작동되며 클라이언트와 패킷을 주고받고 다량의 NPC와 플레이어의 접속에 따른 부하를 줄이기 위해서 플레이어와 몬스터 들에 대한 시야처리(7X7의 시야에 존재하는 플레이어 및 몬스터만 보임)를 하였습니다. 몬스터 NPC는 타이머 스레드를 이용하여 1초 간격으로 이동합니다. 플레이어와 지형, 몬스터 NPC, 아이템 간의 충돌처리 또한 시야처리를 통해 플레이어 주변 4방향(위, 아래, 오른쪽, 왼쪽)에 존재하는 것들만 충돌처리를 진행하도록 하였습니다. 이외에 MSSM(Microsoft SQL Server Management)를 이용하여 Database 를 구축하였으며 플레이어의 위치, 남은 체력, 먹은 먼지의 수, 아이디, 패스워드를 관리합니다. LUA 스크립트 언어를 이용하여 플레이어와 몬스터 NPC간에 충돌이 일어나면 몬스터 NPC가 “Catched”라는 메시지를 띄우도록 구현하였습니다.

패킷을 주고받는 WorkerThread에 4개의 스레드를 할당하고 플레이어의 접속을 받는 AcceptThread, 몬스터 NPC AI가 작동되는 TimerThread 이렇게 총 6개의 스레드를 이용하여 서버가 구동됩니다. 멀티 스레드에 의한 동기화는 Mutex를 사용하여 작업하였습니다.

**[2]. Boost asio와 멀티 스레드를 이용한 3인칭 슈팅 대전게임 제작 [졸업작품]**

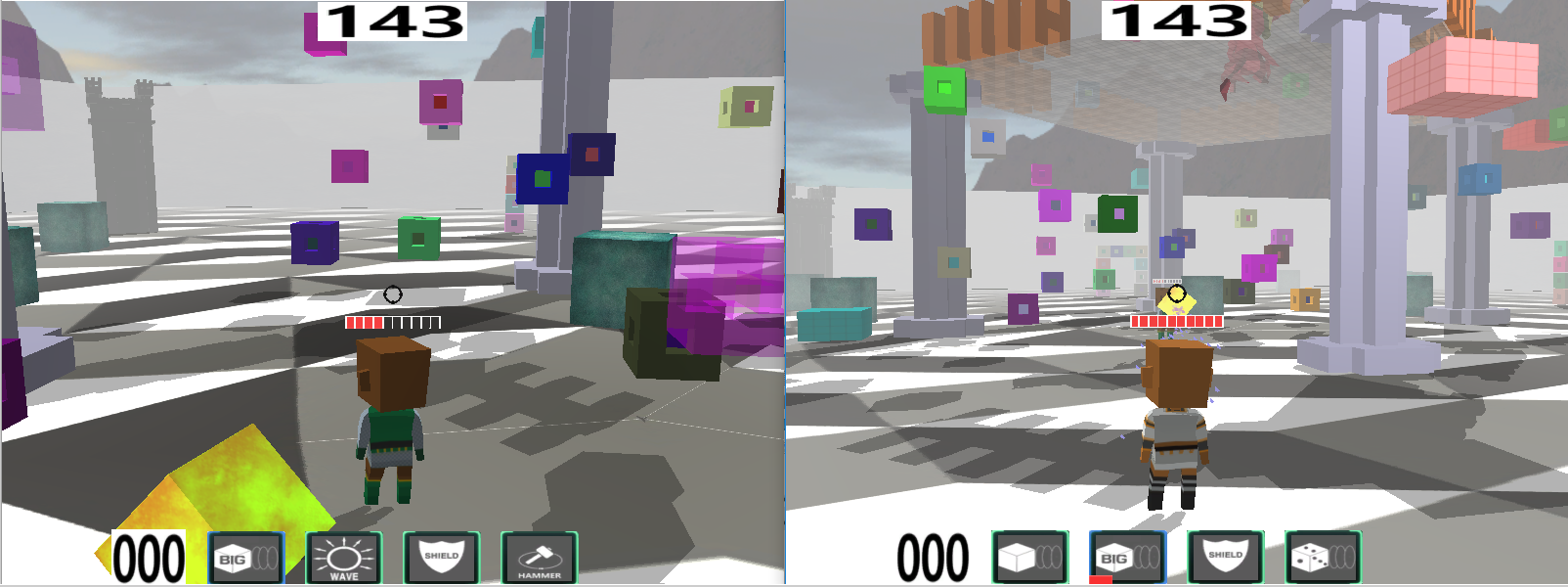
****

1. **기본 설명**

* **게임 제목: MagicCube**
* **깃 허브:** [**https://github.com/ChangminYoo/MagicWorld**](https://github.com/ChangminYoo/MagicWorld)
* **제작 도구: C++, C++ STL, Boost asio 라이브러리, DirectX12**
* **개발 환경: Visual studio 17**
* **장르: 3인칭 슈팅 대전 게임**
* **제작 인원: 3명 (클라이언트 2, 서버 1)**
* **제작 기간: 약 12개월(기획 및 팀 조정 3개월, 제작 7개월, 추가 수정 2개월)**
* **목표: 비동기 입출력 IO인 Boost asio와 멀티 스레드를 이용한 대규모 접속 가능 서버 제작**

1. **세부 내용 설명**

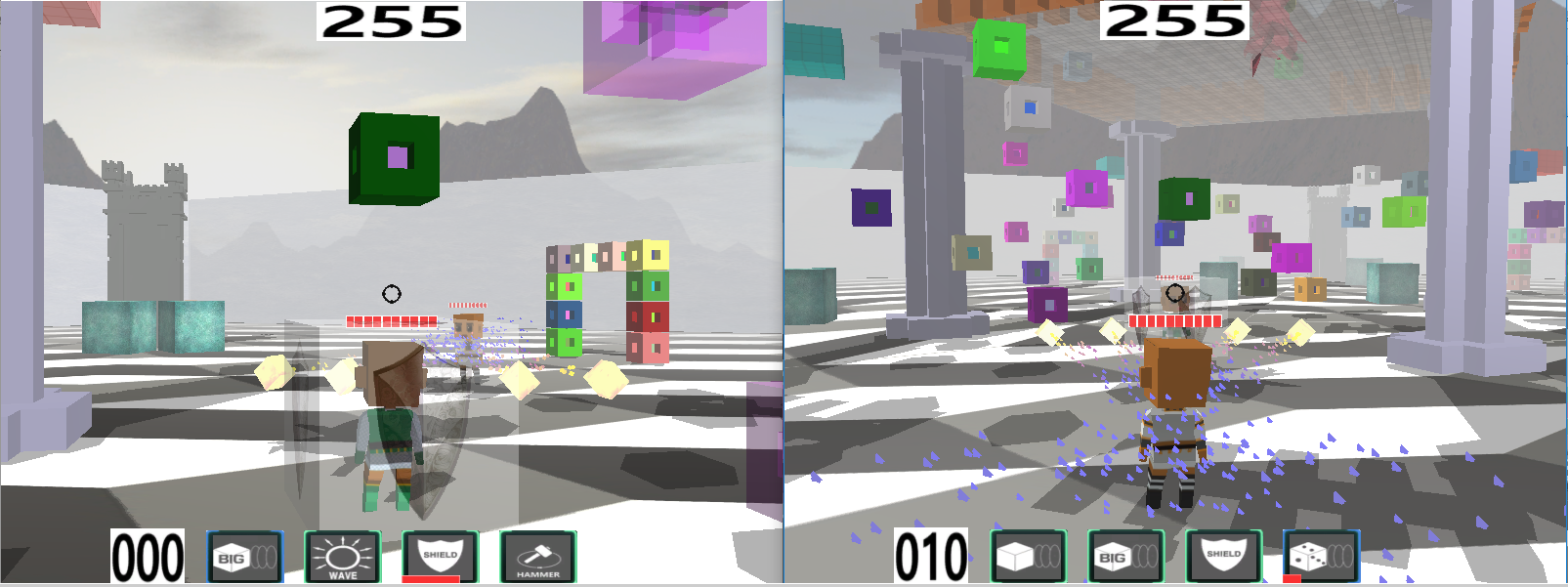
* **게임 목표: 대전 스테이지에서 제한 시간 (2분 30초) 안에 스킬을 사용하여 적 플레이어를 처치하거나 2층에 존재하는 몬스터를 처치하여 가장 많은 점수를 획득하여 1등을 하는 것이 최종 목표입니다.**

****

**[플레이 화면]**

* **게임 내용: 최대 5명이 개인전으로 즐길 수 있는 3인칭 슈팅 대전게임 입니다. 총 10 종류의 캐릭터 중 원하는 캐릭터를 선택하여 스테이지에 입장 할 수 있습니다. 10 종류의 캐릭터는 보유하고 있는 스킬이 다르며 캐릭터 당 4개의 스킬을 사용할 수 있습니다. 대전 스테이지인 맵은 1층과 2층으로 나뉘어져 있으며 2층에는 임프 몬스터 NPC가 1마리 존재합니다. 맵에는 각종 장애물이 존재하며 물리엔진이 적용되어 플레이어 공격에 맞으면 움직이는 Rigidbody 오브젝트가 존재합니다. 플레이어는 1층과 2층 모두 이동 할 수 있으며 2층으로의 이동 수단으로서 위, 아래로 움직이는 엘리베이터를 사용해 이동 할 수 있습니다. 게임 플레이 시간은 총 2분 30초로 이 시간 안에 죽더라도 5초 뒤에 처음 시작한 자리에서 다시 리스폰되어 싸울 수 있습니다. 플레이어는 자신이 보유한 4개의 스킬을 이용하여 적 플레이어를 처치할 수 있으며 각각의 스킬은 고유 데미지 및 쿨타임을 가지고 있습니다. 플레이어는 플레이어 이외에 2층에 존재하는 몬스터 NPC도 플레이어 간 협동을 통해서 처치할 수 있습니다. FSM이 적용된 몬스터 NPC는 플레이어가 일정거리 접근 시 플레이어를 따라가 공격하며, 일정거리 이상 떨어지게 되면 다시 원래 자리로 돌아갑니다. 게임 내 랭킹 시스템 및 버프 효과를 도입하여 적 처치 시 10 스코어, 몬스터 처치 시 100 스코어를 획득할 수 있으며 스코어, 킬과 데스 수를 이용하여 1등 플레이어를 정해서 버프 효과를 줍니다. 1등 플레이어는 자신의 캐릭터 크기가 커지며, 체력 상승 및 속도 상승 버프 효과를 받습니다.**
* **제작 일정 및 이슈 : 게임기획 및 팀 조정을 12월 ~ 2월까지 진행하였습니다. 이 기간 중간 팀원 중 클라이언트 담당 한명이 이탈하여 팀 조정에 좀 더 많은 시간을 할애하게 되었습니다. 추후 2월 ~ 8월 중순까지 본격적으로 게임을 제작하였습니다. 서버 모델로 사용하는 boost asio를 이용하여 대전게임에 맞게 서버 프레임워크를 2월 중순부터 6월까지 제작하였습니다. IOCP가 아닌 boost asio를 서버 모델로 사용하여 게임을 제작한 것이 처음이었고 MMOPRG가 아닌 대전게임 형식에 맞게 프레임워크를 짜다 보니 서버 – 클라이언트 통신 패킷 송수신 에러 , 스레드 작업 배분방식 결정(물리엔진, 충돌처리 기능 및 타이머기능 등과 패킷 송수신을 전담하는 부분간 스레드 배분 문제), 클라이언트에서 서버에서 송신된 패킷 처리 하는데 발생하는 버그 등 많은 문제를 겪으며 프레임워크를 작업 하게되어 예상했던 시간 보다 1개월 정도가 더 소요되었습니다. 6월 말부터 8월까지 서버 프레임워크 및 서버 패킷 수신용 클라이언트 프레임워크를 토대로 이전에 추가 연동하지 못했던 부분인 스킬 및 스킬 쿨타임, 물리효과가 적용된 오브젝트, 몬스터 npc, 스테이지 타이머, 랭킹 시스템 등 콘텐츠를 추가 연동하고 로비 제작 및 연동을 하였습니다. 9월 ~ 10월까지 서버의 성능을 더 높이기 위한 추가작업을 진행하였습니다. 서버에서 패킷 송수신처리를 담당하는 부분인 worker thread의 스레드 수를 증가시켰고 기존에 timer thread에 기능 부하를 가져올 수 있는 클라이언트로의 주기적인 패킷 송신 부분을 worker thread에서 해당 작업을 담당하도록 수정하였습니다. 또한 중복 처리되는 부분이나 필요 없는 부분을 정리하였고 패킷 송수신의 부하를 줄이기 위해서 프로토콜 부분에서 패킷 데****이터의 크기를 줄이고 초당 송수신 횟수도 조정하였습니다.**

**[로비 화면(캐릭터 선택창)]**

* **게임 흐름도: 타이틀 화면 -> 로비 화면(캐릭터 선택) -> 로딩화면 -> 메인 스테이지(게임 플레이) -> 종료(시간 초과)**

**[플레이 화면]**

* **내부 구현: 내부 구조에 대한 흐름도는 main 스레드에서 서버 접속 이전에 진행되는 작업 과정인 현재 컴퓨터의 CPU 개수 체크, 맵 및 맵에 존재하는 오브젝트 데이터 설정, 몬스터 npc 데이터 설정을 진행합니다.**

**해당 초기화작업 완료 후 boost asio의 async\_accept를 이용하는 Accept 함수**

**부분에 접근하여 클라이언트와의 접속처리를 한 뒤 Callback되는 handler 부분에서 서버에서 관리 할 클라이언트 부분을 설정하고 boost asio의 async\_read\_some을 이용하는 RecvPacket함수 호출을 한 뒤 해당 비동기 IO에 의해 Callback 되는 handler 부분에서 클라이언트부터 오는 패킷 데이터를 재조립하여 수신된 패킷을 처리하는 부분인 ProcessPacket 함수를 처리합니다.**

**이 부분은 현재 서버로 받아온 패킷 데이터가 남아있을 때까지 해당 패킷 데이터를 ProcessPacket 함수의 인자로 넘겨주는 루틴을 돌면서 처리됩니다. ProcessPacket 함수에서는 넘어온 패킷 데이터에 설정된 패킷 타입에 따라서 데이터를 처리합니다. 로비 씬, 씬 변환, 플레이어의 기본 동작(이동, 회전, 점프, 애니메이션, 공격) 및 스킬, 주기적인 패킷 처리(변화된 플레이어 정보, 공격 정보, 스킬 정보, NPC 정보, 움직이는 맵 오브젝트 및 물리효과가 적용된 오브젝트 정보)에 대한 데이터 처리를 진행합니다. 해당 작업을 하는 worker thread는 총 8개의 thread들 중 6개의 스레드를 사용하여 작업하고 있습니다.**

**다음으로 게임 내 적용되는 물리효과 및 충돌처리를 전담하는 physics thread에 대해서 알아보면, 해당 스레드는 1개의 스레드를 이용하여 루프를 돌면서 작업을 진행합니다. 이 부분에서 게임 내 모든 오브젝트(플레이어 오브젝트, 플레이어 공격에 의해 생성된 불렛 오브젝트, 맵 오브젝트, NPC 오브젝트)에 적용되는 중력, 가속도 등에 대한 물리효과처리 및 Rigidbody 오브젝트에 설정된 물리 값들에 따른 물리효과 처리를 진행합니다. 이 외에 게임 내에서 진행되는 모든 충돌처리를 진행합니다. 이 부분에서 FSM이 적용된 몬스터 NPC의 데이터 또한 업데이트 됩니다.**

**서버에서 주기적으로 시간을 체크하여 진행되어야 하는 작업은 timer thread에서 전담하며 해당 스레드는 1개의 스레드가 할당 되어있습니다. 이 부분에서는 실드, 파동파 같은 일정시간 동안 시전되고 사라지는 스킬 및 캐릭터 리스폰에 대한 작업을 담당합니다.**