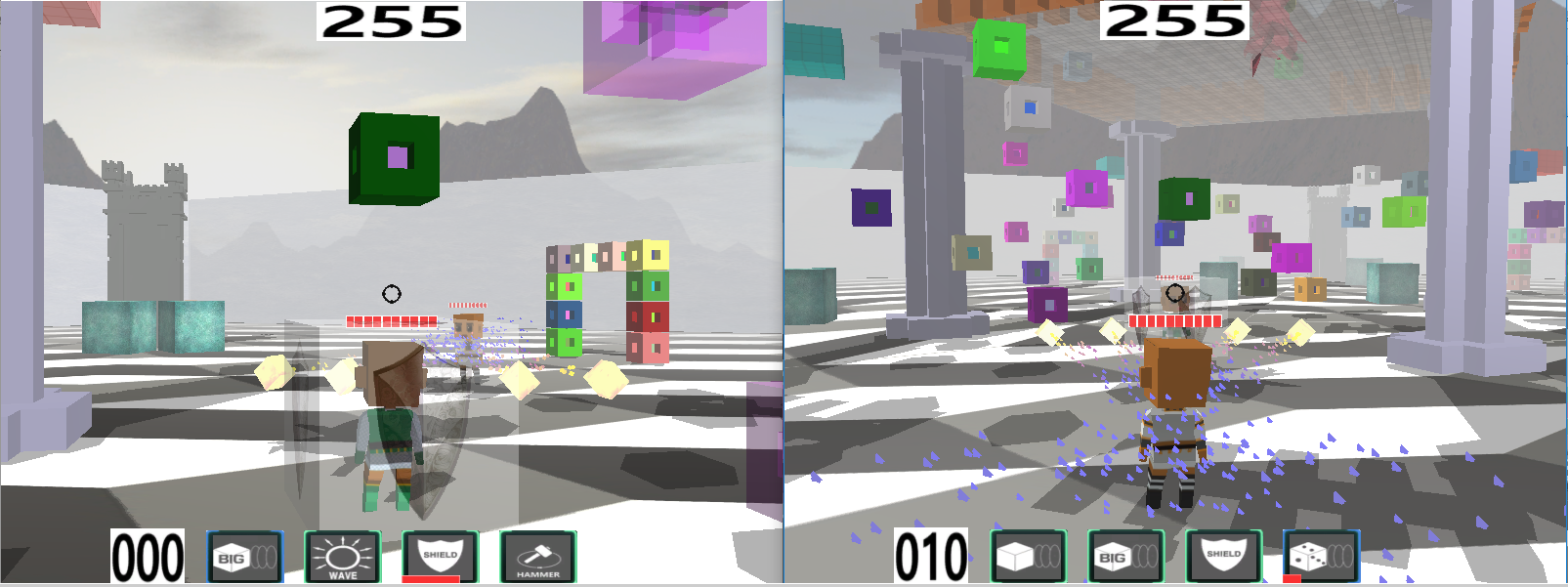
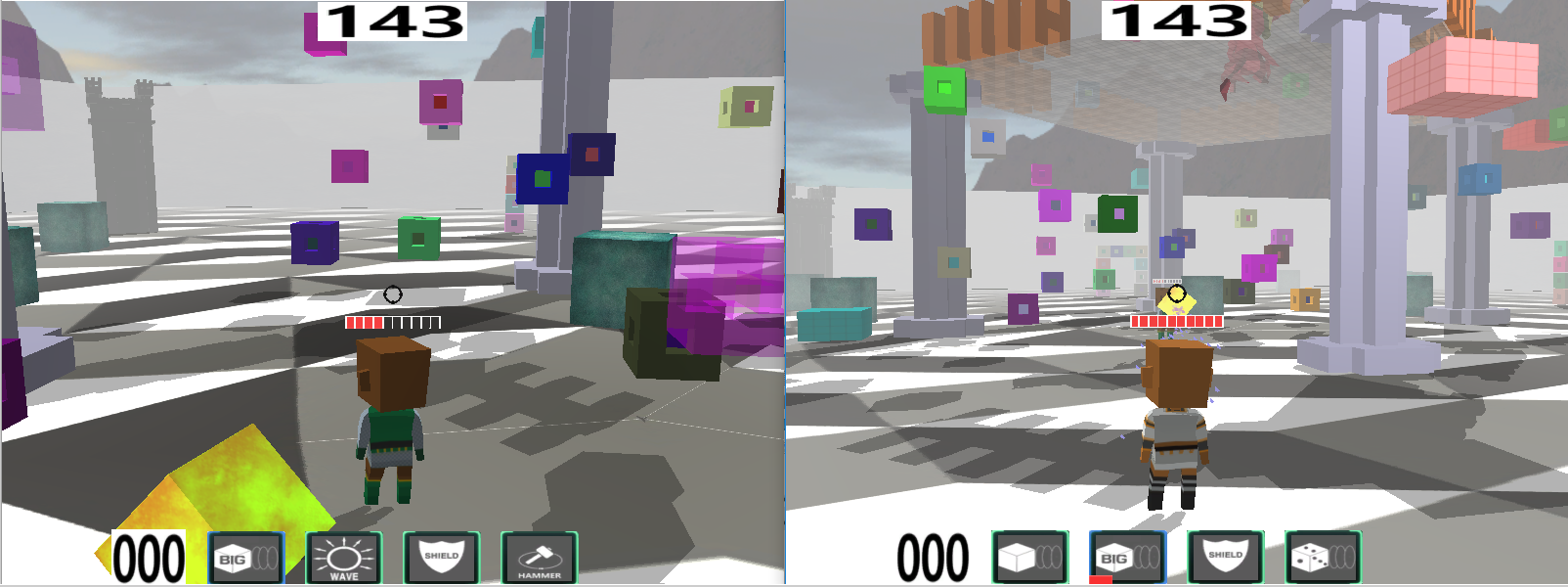
**[서버]**

**[1]. Boost asio와 멀티 스레드를 이용한 3인칭 슈팅 대전게임 [졸업작품] (2018년 제작)**

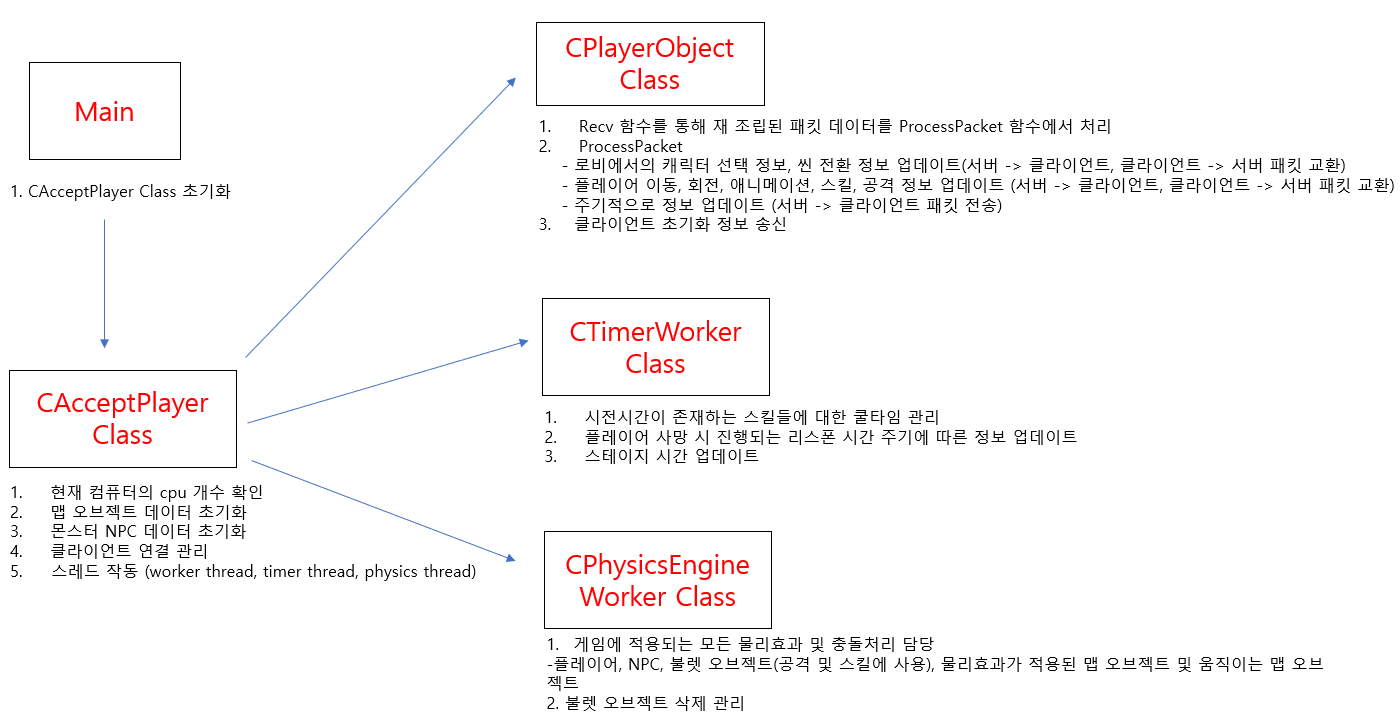
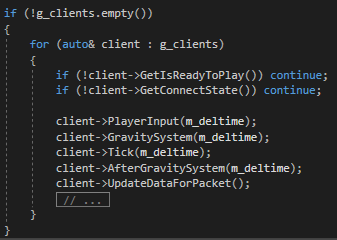
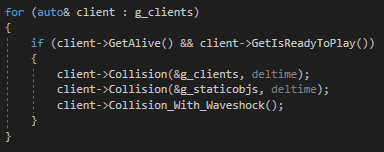
1. **기본 설명**

* **게임 제목: MagicCube**
* **깃 허브:** [**https://github.com/ChangminYoo/MagicWorld**](https://github.com/ChangminYoo/MagicWorld)
* **동영상링크:** [**https://youtu.be/YNxNwJYi2to**](https://youtu.be/YNxNwJYi2to)
* **제작 도구: C++ 11 , Boost asio Library, Direct3D12**
* **개발 환경: Visual studio 2017**
* **장르: 3인칭 슈팅 대전 게임**
* **플랫폼: PC**
* **제작 인원: 3명 (클라이언트 2, 서버 1)**
* **제작 기간: 약 12개월(기획 및 팀 조정 3개월, 제작 7개월, 추가 수정 2개월)**
* **역할 및 목표 : boost asio를 이용한 서버 프레임워크 및 서버 통신을 위한 클라이언트 프레임워크 제작, 물리엔진 및 각종 게임 내 모든 컨텐츠 서버 클라이언트 간 동기화. 비동기 입출력 IO인 Boost asio와 멀티 스레드를 이용한 대규모 접속 가능 서버 제작**
* **맵 크기: 300 x 300 (m2)**

**[게임 플레이 화면]**

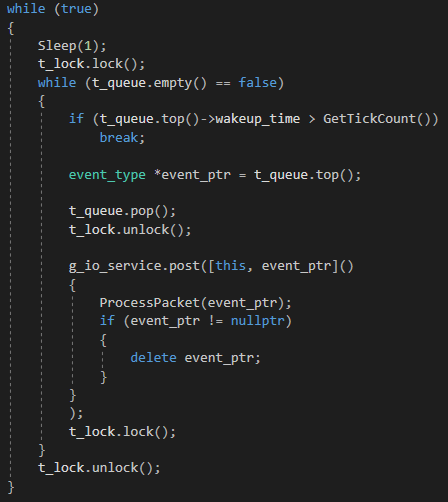
1. **세부 내용 설명**

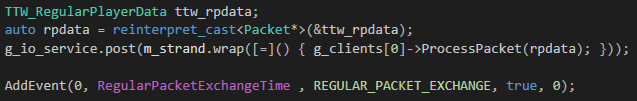
* **게임 목표: 대전 스테이지에서 제한 시간 (2분 30초) 안에 스킬을 사용하여 적 플레이어를 처치하거나 2층에 존재하는 몬스터를 처치하여 가장 많은 점수를 획득하여 1등을 하는 것이 최종 목표입니다.**
* **게임 내용: 최대 5명이 개인전으로 진행되는 3인칭 슈팅 대전 게임입니다. 플레이어는 4개의 고유 스킬을 지닌 10종류의 캐릭터 중 하나를 선택하여 게임을 플레이 할 수 있습니다. 맵에는 물리엔진이 적용된 오브젝트, 움직이는 오브젝트, 움직이지 않는 설치형 오브젝트가 존재하며 1층과 2층으로 나뉘어져 구성되어 있습니다. 2층에는 몬스터 NPC가 존재하며 플레이어는 다른 플레이어를 처치하거나 몬스터를 처치하여 자신의 게임 스코어를 올릴 수 있습니다. 스코어가 가장 높은 1등 플레이어에게는 게임에 유리한 버프 효과가 적용되며 스테이지 제한 시간이 모두 지나면 게임이 종료됩니다.**
* **내부 구현: 메인 스레드에서 현재 컴퓨터의 CPU 개수 체크, 맵 및 맵에 존재하는 오브젝트 데이터 설정, 몬스터 NPC 데이터 설정을 진행한 후 접속을 시작합니다. 현재 이 프로그램에서 작동되는 스레드는 메인 스레드를 포함하여 총 9개이며, 클라이언트와 서버 간 송수신되는 패킷 데이터를 처리하는 6개의 워킹 스레드, 물리엔진 및 충돌처리를 담당하는 1개의 물리 스레드 그리고 타이머에 따라서 주기적인 데이터처리를 하는 1개의 타이머 스레드로 구성되어 있습니다.**

**[ 서버 구조를 이루고 있는 주요 클래스들 ]**

**[ 모든 오브젝트에 적용되는 물리효과 및 충돌처리를 하는 PhysicsUpdate 함수]**

**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. 몬스터 NPC는 FSM방식으로 구현되어 자신과의 일정거리에 플레이어가 접근하면 대상으로 이동하여 공격을 하고 그 위치를 벗어나면 다시 제자리로 돌아옵니다.**

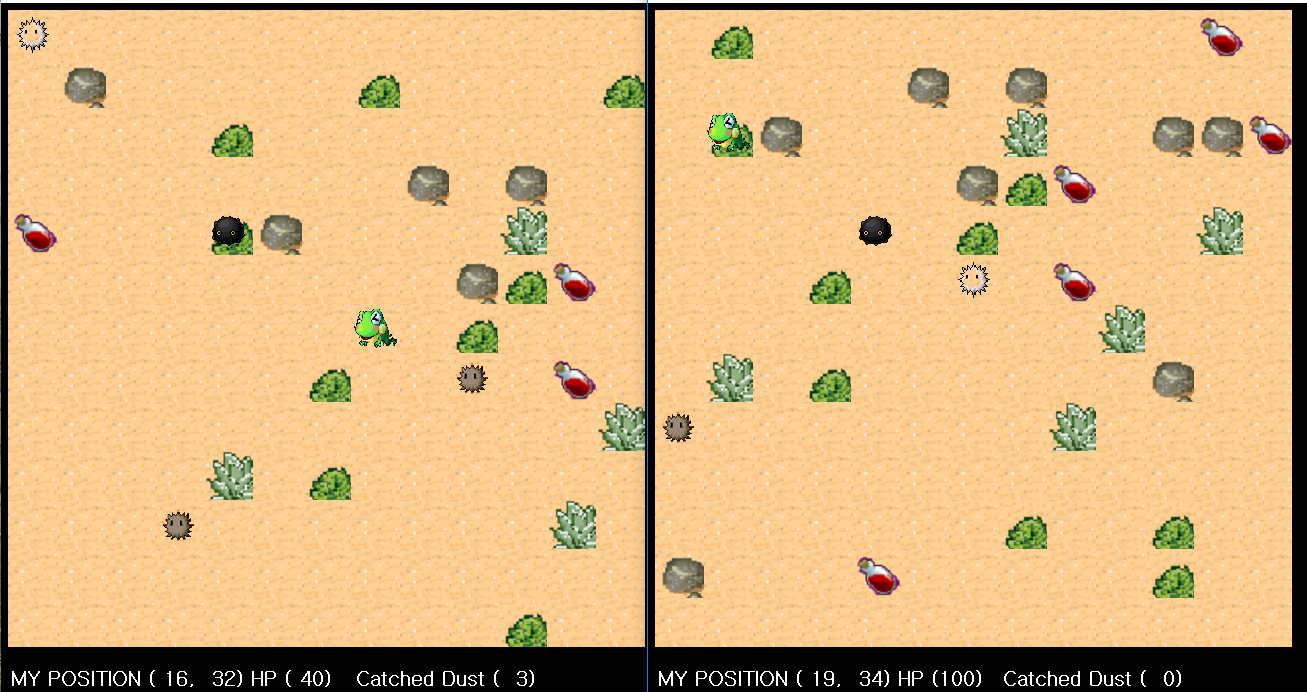
****

**[ priority\_queue 에 넣어진 이벤트들을 시간에 따라 빼내 실행시키는 부분 ]**

**[ post 기능을 통해 CPlayerObject 클래스의 ProcessPacket을 호출하는 부분 ]**

**CTimerWorker 클래스에서 위처럼 priority\_queue에 처리해야 될 이벤트를 넣고 자신의 실행주기가 되면 CTimerWorker 클래스의 ProcessPacket 함수에서 스킬 쿨타임 업데이트, 리스폰 등과 같은 이벤트들이 처리됩니다. 기존에는 이 부분에 주기적으로 업데이트가 필요한 게임 상의 모든 정보를 업데이트하고 이들을 CPlayerObject의 SendPacket 함수를 통해 클라이언트로 송신하는 기능까지 도맡아 했습니다. 이 부분에서 잦은 SendPacket 함수 호출 및 컨테이너 탐색을 통한 오버헤드로 인하여 우선순위 큐에 들어간 다른 작업들 보다 처리 시간이 길어지게 되어 뒤 이은 작업들의 처리순서가 쌓이게 되고 이로 인하여 제때 처리되어야 할 작업들이 지연되어 처리되는 현상이 발생했습니다. 예로, 2인 플레이 도중 키보드를 이용하여 스킬 키를 눌러 공격을 하였는데 스킬 쿨타임 처리부분이 지연되어 바로 스킬을 사용할 수 없는 현상이 발생했습니다.**

**위와 같은 이유로 처리량이 많아 오버헤드가 큰 부분을 상대적으로 스레드 수가 많아 패킷 송수신을 도맡아 하는 worker thread에서 처리하도록 post 기능을 사용하여 CPlayerObject의 ProcessPacket를 호출하여 작업을 처리하였고 컨테이너 탐색 부분도 통합되어 탐색해도 될 부분들을 압축하여 기존 처리시간보다 평균적으로 좀 더 빠른 시간 안에 처리될 수 있도록 수정하였습니다.**

** [2]. IOCP와 멀티 스레드를 이용한 2D 대전 게임 (2018년 제작)**

1. **기본 설명**

**- 게임 제목: Dust Eating**

**- 깃 허브:** [**https://github.com/alclsTit/2018GameServer**](https://github.com/alclsTit/2018GameServer)

**- 동영상 링크:** [**https://youtu.be/eZ5cpKh1OM0**](https://youtu.be/eZ5cpKh1OM0)

**- 제작 도구: C++11, Direct3D9, IOCP**

**- 개발 환경: Visual studio 2017**

**- 장르: 대전 게임**

**- 플랫폼: PC**

**- 제작 인원: 1명**

**- 제작 기간: 약 2개월 (기획 및 그래픽데이터(스프라이트, 이미지) 수집 1주, 제작 6주)**

**- 목표 및 역할: IOCP를 이용한 서버 프레임워크 제작, 멀티 스레드를 이용한 작업 분담처리, 시야처리, 충돌처리, 몬스터 NPC 구현, 맵 및 오브젝트 구현, DB 연동, LUA를 이용한 스크립트 연동. 비동기 IO를 지원하는 IOCP 소켓모델과 멀티 스레드를 이용한 대규모 접속가능 서버 제작.**

**- 맵 크기: 200 x 200 (pixel)**

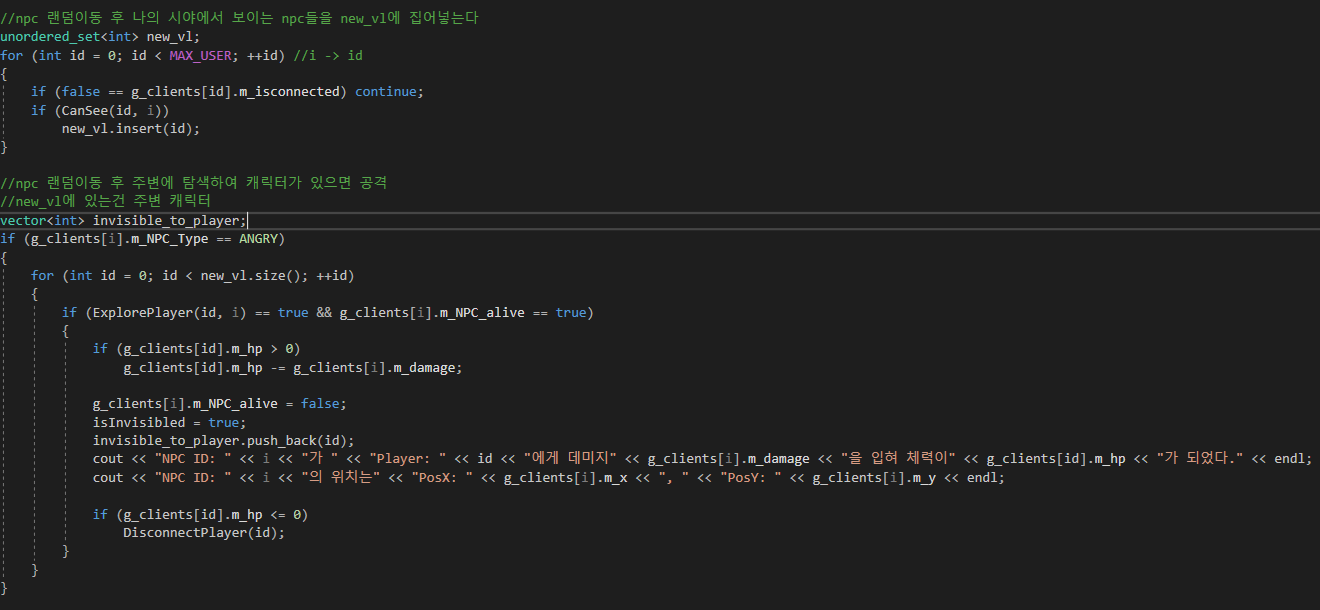
**- 최대 동시 접속: 평균 1500명**

1. **세부 내용 설명**

* **게임 목표: 체력을 유지하며 맵 상에 떠다니는 먼지 몬스터 NPC를 상대 플레이어 보다 더 많이 잡아먹어 더 많은 먼지를 없애는 것이 최종 목표입니다.**
* **게임 내용: DB에 저장된 로그인정보를 통해 게임에 접속하여 게임을 진행합니다. 맵 상에 존재하는 각종 장애물들을 피해서 떠다니는 몬스터 NPC를 잡아먹어야 합니다. 플레이어 자신은 검은 성게로 표시되며 상대방은 개구리 이미지로 표시되어 나타납니다. 2종류의 몬스터 NPC 중 한 종류는 플레이어를 공격해 피해를 입히며 플레이어는 맵 상의 체력 포션을 먹어가며 모든 체력이 소진되기 전까지 상대 플레이어보다 더 많은 먼지 몬스터 NPC를 잡아먹어야 합니다.**
* **내부 구현:**

**[1]. 스레드 할당 및 각 스레드에서 하는 작업**

**플레이어가 움직일 때마다의 업데이트되는 시야, 플레이어 위치, 충돌 처리 , 1초 마다 랜덤하게 변화되는 위치 및 그에 따른 시야처리 등 지속적으로 업데이트 되는 데이터를 클라이언트와 송수신 해야 되는 많은 부하가 생길 수 있는 작업을 담당하는 worker thread에 가장 많은 스레드를 6개를 할당하였습니다. 클라이언트의 접속 처리를 담당하는 accept thread는 1개로 구성되어 있습니다. 주기적인 행동을 하는 몬스터 NPC에 대한 데이터처리를 하는 부분인 timer thread 또한 1개로 구성되어 있습니다.**

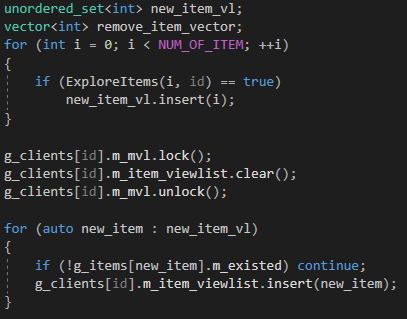
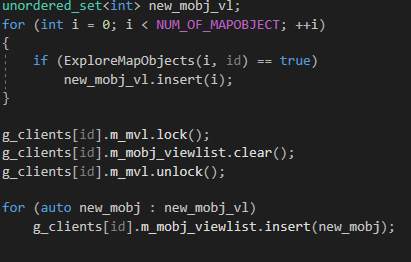
**[2]. 랜덤이동 한 몬스터 NPC의 시야처리 및 플레이어 공격 부분**

3

1

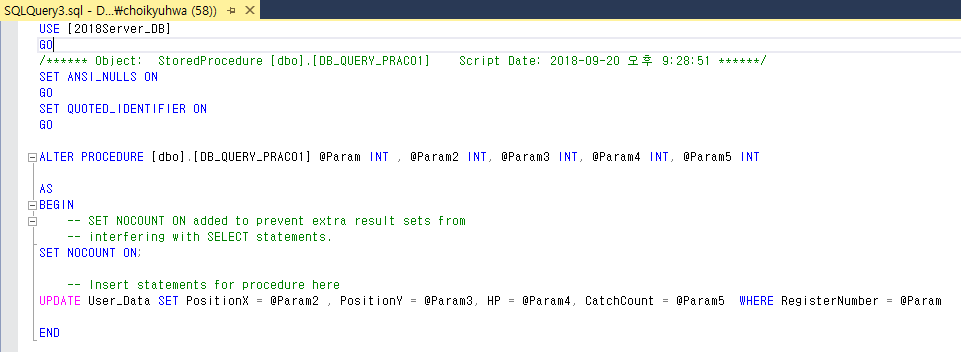
2

**몬스터 NPC들은 1초마다 상, 하, 좌, 우 중 한 곳으로 랜덤하게 좌표이동을 합니다. 해당 게임에서는 몬스터가 2500마리 정도되는 많은 수이므로 시야처리를 통해서 플레이어 주변(7X7 pixel)에 있는 몬스터 NPC들에 대해서만 이동을 하고 화면에 보이며 충돌처리를 진행합니다.**

** [1] 부분은 new\_vl 컨테이너에 랜덤이동한 몬스터 NPC의 시야에 포착된 플레이어 아이디를 넣습니다. [2] 부분에서는 업데이트 된 new\_vl 컨테이너를 탐색하면서 몬스터 NPC의 공격 시야에 해당 플레이어가 포착되면 플레이어의 체력에 데미지를 주는 기능을 합니다. [3] 부분에서는 플레이어에게 데미지를 주어 죽은 몬스터 NPC를 invisible\_to\_player 컨테이너에 집어넣고 추후 해당 컨테이너를 탐색하면서 몬스터 NPC를 컨테이너 안에서 제거합니다.**

**[3]. 아이템 및 맵 오브젝트 시야처리**

**몬스터 NPC 이외에도 게임 상에 아이템은 1000개 , 맵 오브젝트는 3000개 존재하므로 이들에 대한 시야처리 또한 진행하였으며 플레이어의 시야에 보이는 오브젝트들만 따로 컨테이너에 담아 이들을 이용해서 작업을 진행하도록 구현하였습니다.**

**[4]. MSSQL의 Stored Procedure을 이용한 DB 연동**

**플레이어가 접속을 끊었을 때 해당 게임에서의 마지막 플레이어 위치, 체력 그리고 지금까지 잡은 먼지 몬스터 NPC 수를 업데이트하여 DB에 저장하게 끔 하기 위해 간단한 쿼리문을 Stored Procedure(저장 프로시저)를 이용하여 작성하였습니다.**

**[클라이언트]**

**[1]. Win32 API 및 C++을 이용한 2D 메이플스토리 모작 (2016년 제작)**

1. **기본 설명**

**- 게임 제목: Project M**

**- 깃 허브: -**

**- 동영상 링크:** [**https://youtu.be/R707-BxcN-8**](https://youtu.be/R707-BxcN-8)

**- 제작 도구: C++, Win32 API**

**- 개발 환경: Visual studio 2013**

**- 장르: 2D RPG**

**- 플랫폼: PC**

**- 제작 인원: 1명**

**- 제작 기간: 약 3개월**

**- 목표: 어릴 적에 재미있게 플레이 해봤던 메이플 스토리를 직접 제작**

**- 역할: C와 C++언어를 이용하여 사용할 클래스 구현 및 프레임워크 제작, 플레이어 구현(이동, 점프, 공격, 4종류의 스킬), 몬스터 구현(이동, 공격, 스킬 사용, AI, 리스폰), 아이템 구현(표창), 퀘스트 구현(단순한 텍스트 출력), 스테이지 전환 및 화면 이동 구현, 인터페이스 구현, BGM 추가**

**[2]. Python 을 이용한 2D 러닝 + 리듬게임 제작 (2016년 제작)**

1. **기본 설명**

**- 게임 제목: Run your beat**

**- 깃 허브: https://github.com/alclsTit/2D-Game-Programming**

**- 제작 도구: Python, Pico-2d(수업시간에 제공받은 모듈)**

**- 개발 환경: PyCharm**

**- 장르: 2D 러닝 + 리듬 게임**

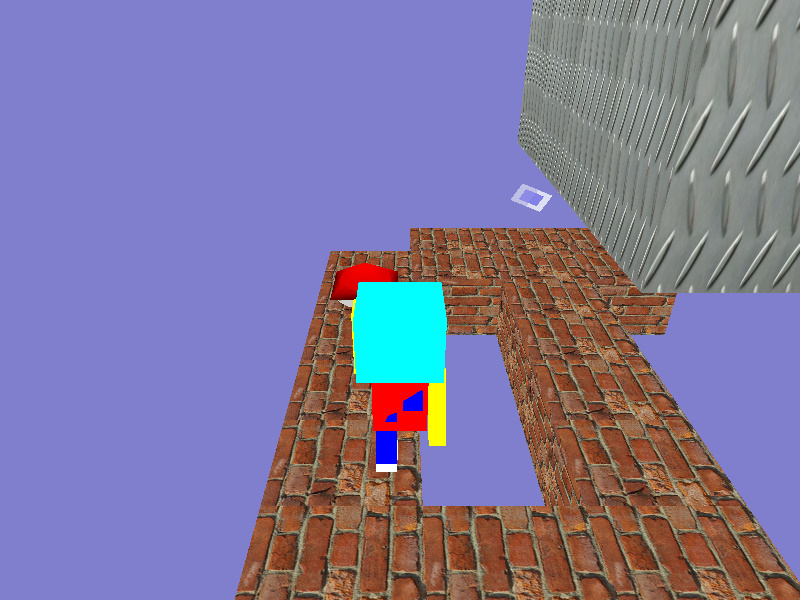
**- 플랫폼: PC**

**- 제작 인원: 1명**

**- 제작 기간: 2개월 ~ 3개월**

**- 목표: Python과 제공받은 모듈을 이용하여 러닝과 리듬을 합친 새로운 게임을 제작**

**- 역할: Python 모듈인 pico-2d를 이용하여 PNG 스프라이트, 이미지 출력. 스테이지 및 각종 오브젝트 구현, 플레이어 구현, 몬스터 NPC 구현, 리듬게임 요소 구현(상단에 표시되는 A,S,D,F, 폭탄, 황금돼지에 맞는 키를 음악에 맞게 눌러 스코어를 획득), 랭킹 시스템 구현, 로비 씬 구현(플레이할 노래 선택), 종료 씬 구현**

**[3]. C++, OpenGL을 사용한 3인칭 3D 탐험 게임 제작 (2017년 제작)**

1. **기본 설명**

**- 게임 제목: Cube Mario**

**- 깃 허브: -**

**- 제작 도구: C++ 11, OpenGL**

**- 개발 환경: Visual Studio 2013**

**- 장르: 3D 액션 게임**

**- 플랫폼: PC**

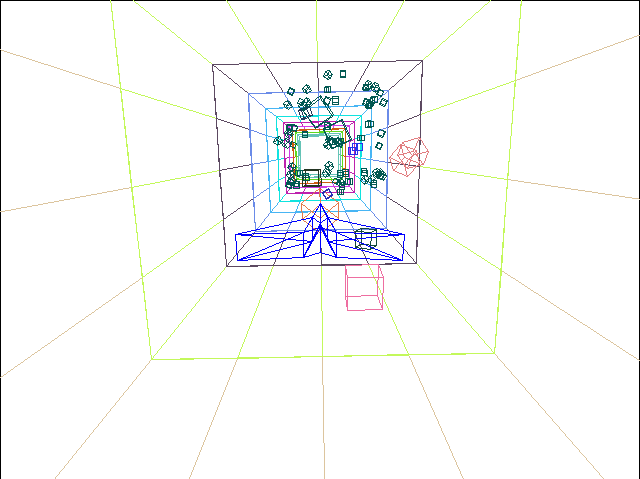
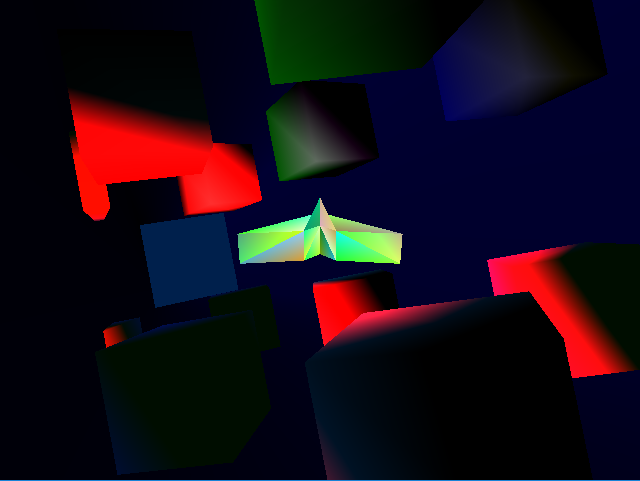
**- 제작 인원: 2명**

**- 제작 기간: 2개월 ~ 3개월**

**- 목표: OpenGL을 이용하여 플레이 가능한 3D 게임 제작**

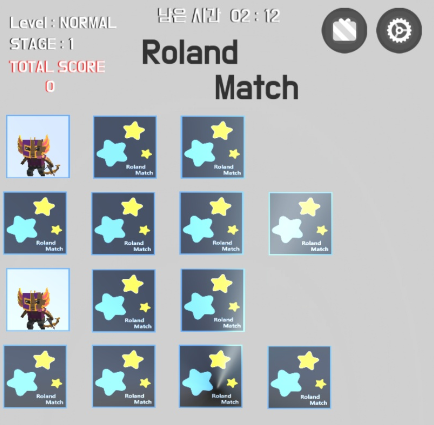
**- 역할: 플레이어 캐릭터 구현. 버섯 몬스터 구현 및 버섯과의 충돌처리 부분 구현. 캐릭터 사망 시 생성되는 큐브 형태의 파티클 구현. 스테이지 제작. 텍스쳐 수집 및 적용**

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

****

**3학년 1학기 때 3D 게임프로그래밍 수업을 들으면서 DirectX12를 이용한 3D게임 제작에 대한 기초적인 지식을 습득했습니다.**

**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.**

**[5]. Unity3D 엔진을 이용한 짝 맞추기 게임 제작 (2017년 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.**