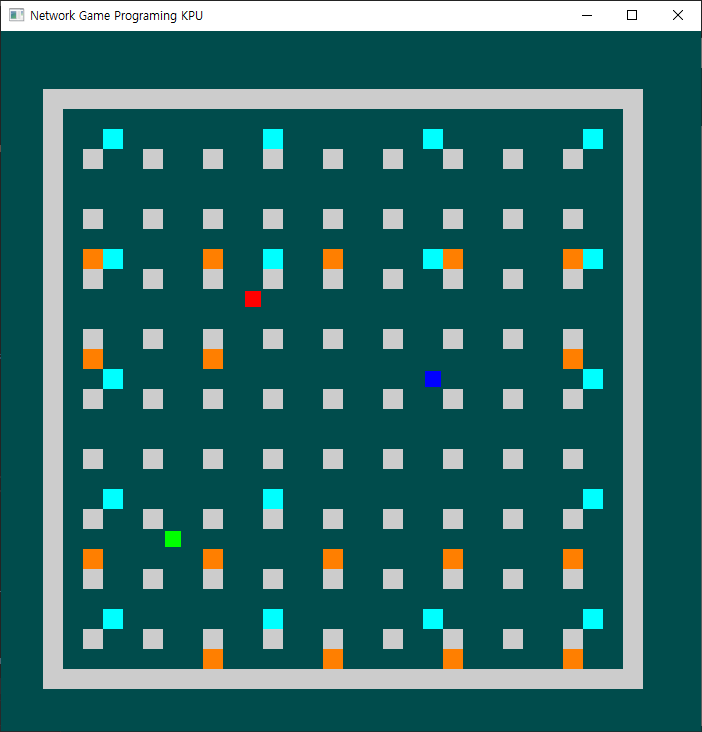
**게임 서버 프로그래머**

**포트폴리오**

**진재혁**

1. NGP : 네트워크 게임 프로그래밍 텀프로젝트
2. Veles : 졸업작품
3. 게임 서버 텀프로젝트 추가 (전투까지 구현) => 동접 측정에 관한 내용 필추

**1. NGP**



**Github:** <https://github.com/hyungtak/NGP_2020_2>

**사용 도구:** C++11, OpenGL

개발 환경: Visual Studio 2019 64비트

플랫폼: PC

장르: 아케이드, 액션

제작 인원: 3명

담당 업무: 게임 로직, 로비 구현.

* **게임 설명:**

NGP는 TCP/IP 기반 윈도우 소켓으로 개발하였습니다. 각 플레이어는 빨강, 파랑, 초록색의 색깔의 사각형으로, 방향키와 스페이스 바를 통해 조작합니다. 주황과 하늘색은 폭탄 개수와 폭탄 위력을 증가시키는 아이템으로, “봄버맨”이나 “크레이지 아케이드”와 유사한 게임입니다. 최대 3명까지 플레이 가능합니다.

* **구조**

서버가 실행되면 우선 Game Thread와 Lobby Thread를 실행합니다.

클라이언트가 모두 접속하면 모든 클라이언트에게 접속 완료 패킷을 전송합니다. Process Thread를 최대 플레이어 수만큼 생성합니다.

**a)** Lobby 스레드 \* 1 (본인 구현)

Lobby Thread는 클라이언트의 접속을 담당하며, 클라이언트가 접속할 때마다 Matching Queue에 소켓 정보를 추가합니다. 클라이언트가 모두 접속하면 모든 클라이언트에 씬 전환 패킷을 전송합니다. 이후 Process Thread를 3개 생성한 뒤, Lobby Thread는 종료합니다.

**b)** Scene Data 클래스 \* 1 (공동 구현)

맵의 여러가지 정보와 폭탄 및 장애물 위치, 플레이어 정보 등을 저장합니다.

Game Thread를 통해 처리된 정보들이 저장됩니다.

**c)** Game 스레드 \* 1

게임 로직과 타이머 등을 처리하는 스레드입니다. 일정 주기로 Scene Data를 업데이트합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

<< 본인이 작성한 코드 샘플 필요>>

**d)** Process 스레드 \* 3

Scene Data를 불러오고, 불러온 정보를 클라이언트에게 전송합니다.

클라이언트로부터 받은 키 입력 정보를 업데이트합니다.

**2. VELES**



**Github**: <https://github.com/dnjsgnlwo/BOBY>

**사용 도구:** C++11, DirectX12

**개발 환경:** Visual Studio 2019 64비트

**플랫폼:** PC

**장르:** 로그라이트, FPS

**제작 인원:** 3명

**담당 업무:** 클라이언트-서버 동기화, 충돌처리, 몬스터 AI, 전투 시스템

**동영상 링크:** https://www.youtube.com/watch?v=G-9cBvwS9z8

**게임 설명:** VELES는 IOCP로 구현되었으며, 최대 4인까지 동시에 플레이 가능한 FPS + 로그라이트 게임입니다. 라이플, 샷건, 저격총 중 3가지의 무기를 선택할 수 있습니다. 각 무기들은 발사 속도와 탄창, 공격력 등 특징을 갖고 있습니다. 여러 가지 아이템들을 랜덤한 확률로 얻으면서 던전을 클리어하는 게임입니다.

* **서버 구조**

1. AI 스레드

오브젝트의 상태에 따라서 오브젝트의 위치나 AI를 제어합니다. 또한, 타이머 역할을 하는 스레드입니다.

1. Worker 스레드

클라이언트-서버 통신과 관련된 작업을 진행하는 스레드입니다. 모든 패킷의 처리, 해석을 진행합니다.

1. DataBase 스레드

MSSQL 2019 Express를 사용하였습니다. 데이터베이스 처리는 동기로 이루어지기 때문에 속도가 느려질 수 있으며 게임을 지연시키거나, 버그를 유발할 수 있어 추가로 스레드를 생성하여 처리하였습니다. 또한 데이터베이스 처리 속도 향상을 위해 stored procedure를 적극적으로 사용했습니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

**3. GSP**

사진 추가 예정

**Github:** https://github.com/wlsqc123/gameserverPM

**사용 도구:** C++11, SFML

개발 환경: Visual Studio 2019 64비트

플랫폼: PC

장르: 롤플레잉

제작 인원: 1명

게임 설명: GSP는 IOCP로 구현되었으며, 몬스터를 처치하여 캐릭터를 성장시키는 롤플레잉 게임입니다. 몬스터와 플레이어는 각각 30000마리, 1000명이 접속할 수 있습니다. 몬스터는 근처에 있는 플레이어를 공격하고, 플레이어가 몬스터를 처치하면 일정 경험치를 획득할 수 있습니다.

**1. 섹터 분할**

최대 동시 접속자 수를 늘리기 위해 섹터를 나누고, 동일 섹터에 위치한 플레이어의 시야에 보이는 NPC의 AI만 동작하도록 구현하였습니다. 추가로, 네트워크 전송을 줄이기 위해 각 플레이어의 시야 내에 존재하는 오브젝트의 정보만을 전송하도록 했습니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

**2. NPC AI (전투 시스템)**

텍스트이(가) 표시된 사진

자동 생성된 설명

* **제작하면서 어려웠던 점.**

1. DB 연동

DB 테스트를 위한 솔루션에선 멀쩡하게 잘 실행되던 함수가 게임 서버에선 실행되지 않는 문제점이 있었습니다. 모든 자료형과 함수명을 동일하게 수정하였음에도 빌드 오류가 발생했습니다. 이유는 기존 사용하던 데이터 형식과 ODBC의 C 데이터 형식을 혼동하여 발생한 실수였는데, 테스트 솔루션은 멀티바이트 문자 집합을 사용했지만 서버 솔루션에선 유니코드 문자 집합을 사용하면서 문제가 발생했습니다. 해결 방법은 간단했지만 이 문제를 해결하기 위해 두 가지 방법을 모두 사용해볼 수 있었고, ODBC 데이터 형식에 대해 공부할 수 있었습니다.

2. Worker Thread 동기화

얕은 지식으로 멀티스레드를 사용하면서 힘들었던 점이 많았습니다. 동기화 문제가 계속해서 문제를 일으켰고 디버깅에는 더 큰 어려움이 있었습니다. NGP 게임에선 클라이언트의 수만큼 스레드를 생성했기 때문에 큰 문제는 없었지만 IOCP 게임 서버를 구현하면서 예상치 못했던 문제들이 발생했습니다. 이 문제를 해결하는 과정에서 동기화 방법을 익힐 수 있었습니다. << 수정 예 추가 필요 >> 다음과 같이 데이터레이스가 발생할 수 있는 멤버들을 여러가지 방법으로 보호하였습니다.

텍스트이(가) 표시된 사진

자동 생성된 설명