개발 포트폴리오

황성준

목차

본인 소개

이름 : 황성준
학력 : 단국대학교 소프트웨어학과 재학중 (2019년도 8월 졸업 예정)
지원 직무 : 게임 클라이언트

▶ 약력
2017년 – 종합설계에서 멀티플레이 액션 게임 ‘Dragon’s Treasure’ 개발 (학과 내 종합설계 경진대회 대상 수상)
2018년 – 개인 프로젝트에서 로그라이크 RPG 게임 ‘The End of The War’ 개발
– 아주대학교 창업동아리에서 퍼즐 어드벤쳐 게임 **‘우리는 신촌에’** 개발
– Bridge 게임 제작 연합 동아리에서 2D Platformer 게임 ‘Eclipse’ 개발
2019년 – 개인 팀 프로젝트에서 TCG 장르 게임 ‘Infinity CCG’ 개발 진행 중
– 종합설계에서 **‘참여형 교육을 위한 메신저 프로그램’** 개발 진행 중

▶ GitHub URL : https://github.com/hsj2586/Development-Portfolios

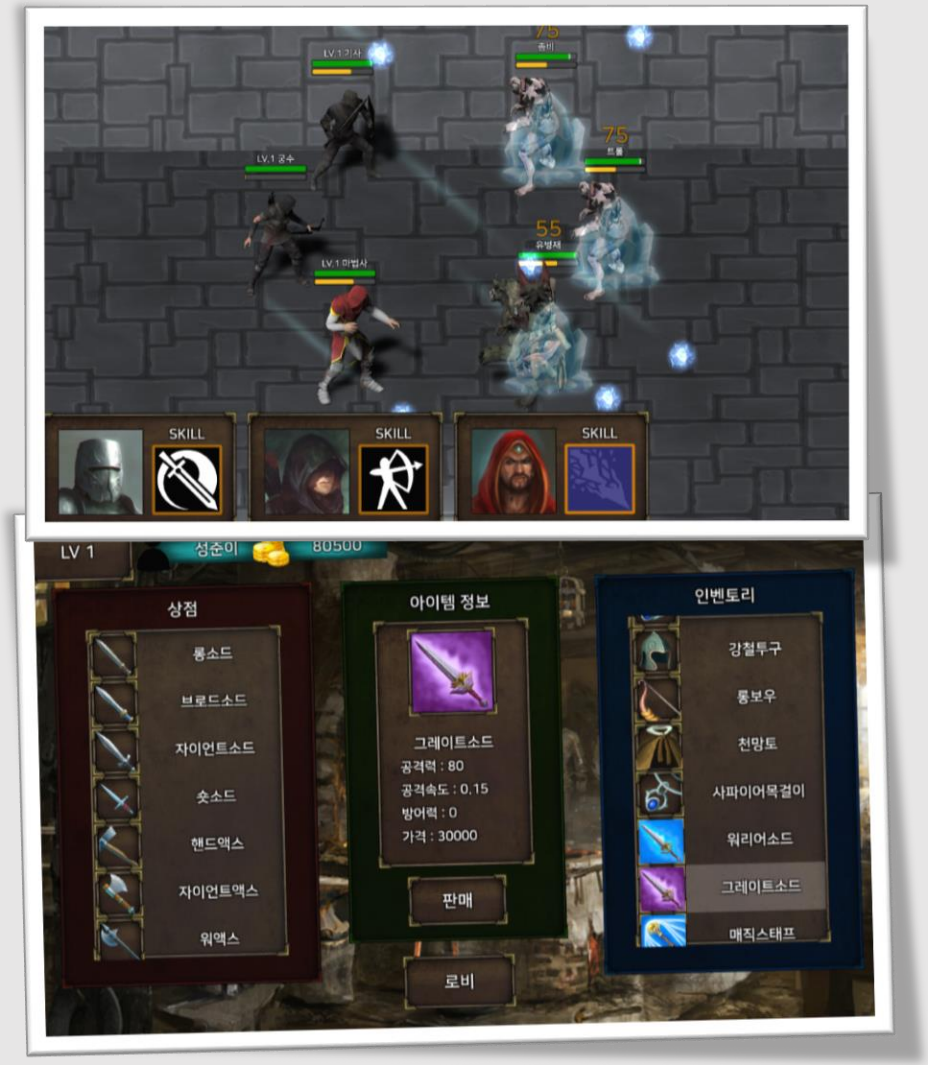
▶ Youtube URL (결과물 플레이 영상)
종합설계 ‘Dragons Treasure’ – https://www.youtube.com/watch?v=XNhZT1uVECw
개인 프로젝트 ‘The End of the War’ – https://www.youtube.com/watch?v=FjfytLLvLLY
아주대학교 창업동아리 **‘우리는 신촌에’** – https://www.youtube.com/watch?v=v-ZfY3sHq1Y
게임 제작 연합 동아리 브릿지 ‘Eclipse’ – https://www.youtube.com/watch?v=IL-2iHfKxe0

1번째 프로젝트 'The End of the War'

프로젝트 이름 : The End of the War
프로젝트 장르 : 게임 (로그라이크 RPG 게임)
프로젝트 인원 : 총원 1명
프로젝트 요약 : 턴제 형 전투 기반의 로그라이크 RPG게임 개발.
수행 역할 : 게임 개발 총괄
개발 환경 : C#, Unity, Visual Studio, JsonUtility
개발 기간 : 2018. 5. 5 ~ 2018. 6. 25

주요 개발 내용

- 아군과 적의 캐릭터 클래스 구성.
- JsonUtility를 통한 각종 데이터의 직렬화.
- Coroutine을 이용해 캐릭터에게 귀속되는 버프 시스템 구현.
- FSM, Queue 자료구조를 이용해 턴제 전투 시스템 구현
- Random 보상 시스템 구현



1번째 프로젝트 ‘The End of the War’

주요 개발 내용 – ‘턴제 전투 시스템’

① ‘턴 시스템’ 구현

캐릭터들이 각자 자신의 FSM에 의거해 동작하면서, 전투의 전체 흐름을 어떻게 제어할지 많은 고민이 있었습니다.
각 캐릭터들은 자신의 FSM에 의해서 독립적으로 동작하게 되는데 전투 전체의 흐름은 또 하나의 독립적인 메커니즘이기 때문이었습니다. 결과적으로는 각 캐릭터가 공격 동작을 취할 때마다 전체적인 전투를 관리하는 클래스인 Battle Manager의 flag 필드를 조작해, 공격을 하는 캐릭터 이외의 캐릭터들은 그에 따라 FSM이 busy waiting 상태가 되도록 구현했습니다.
공격을 마친 캐릭터는 flag를 풀고 다시 온전히 모든 캐릭터가 정상 동작을 취하도록 구현했습니다.

② Queue를 이용한 턴 동기화

턴제 전투 시스템 구현 과정에서 업데이트 단위로 도는 여러 개의 Coroutine 간에 충돌이 일어나 완벽한 순서를 제어하기 어려운 문제가 있었습니다. (예를 들어 공격속도가 같은 캐릭터가 동시에 turn을 얻으면 두 캐릭터가 동시에 공격 행동을 취하는 경우)
그래서 공격 기회를 얻었을 경우 행동 Coroutine을 바로 실행하는 것이 아니라, Queue 자료구조를 이용해 행동 자체를 순차적으로 Queue에 Enqueue하고 실행 후에 Dequeue하는 방식으로 턴 순서 동기화를 구현했습니다. Queue에 들어올 수 있는 행동으로는 공격, 스킬 등 턴을 가져오는 행위가 올 수 있습니다.

2번째 프로젝트 ‘우리는 신촌에’

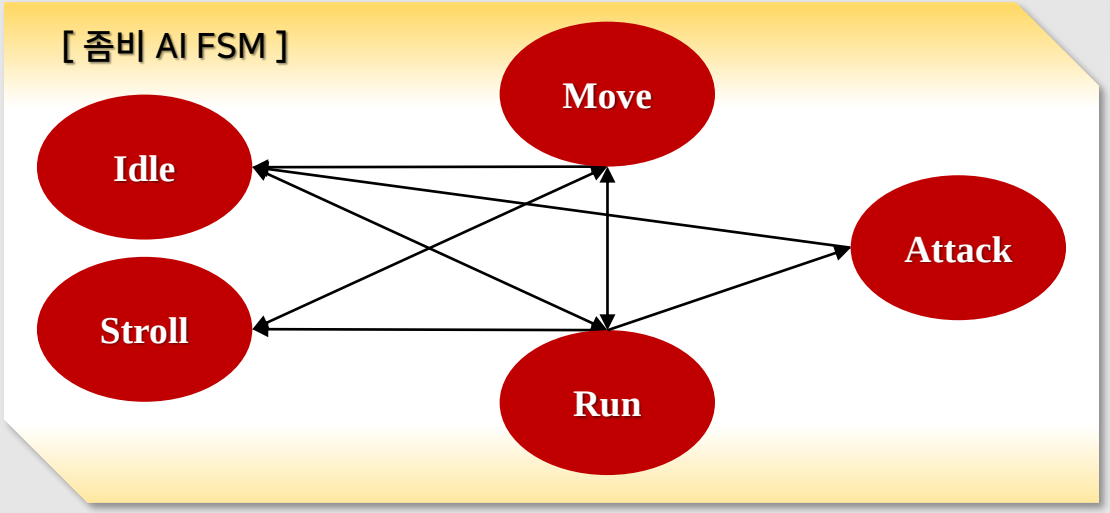
프로젝트 이름 : 우리는 신촌에
프로젝트 장르 : 게임 (어드벤쳐 퍼즐 모바일 게임)
프로젝트 인원 : 총원 6명 (개발 인원 2명)
프로젝트 요약 : 좀비 사태가 일어난 학교에서 학생들을 구출하는 선생님을 플레이 하는 퍼즐 게임 개발.
수행 역할 : 게임 로직 및 각종 시스템 구현.
개발 환경 : C#, Unity Engine, Visual Studio
개발 기간 : 2018. 5. 1 ~ 2018. 12. 30
버전 관리 : Bitbucket(Git), SourceTree
주요 개발 내용
- 좀비의 AI 및 소음 시스템 구현
- 맵의 Section 및 Lighting 시스템 알고리즘 구현
- 팝업 메시징, 말 풍선 기능 구현
- 오브젝트 상호작용 및 아이템 인벤토리 구현

2번째 프로젝트 ‘우리는 신촌에’

주요 개발 내용 – ‘좀비의 AI 및 소음 시스템 구현’

① 좀비 AI 구현

좀비의 행동은 FSM에 의거하여 행동하도록 구현했습니다.
좀비의 Path Finding 기능은 유니티에서 제공하는 Navigation을 이용해 구현했습니다.



발걸음 소음이나 이벤트 발생 등에 따라 소리의 정보를 담은 Sound 객체를 곳곳에서 발생시킵니다. 소음은 좀비를 Move 상태로 전이시키며, 소리의 가중치를 거리와 소리크기에 따라 계산해서 이동 여부를 결정합니다. 그리고 일정 시야(사거리) 내에 플레이어가 들어올 경우 추적합니다. 추적 중에 공격 사거리 내에 들어올 경우는 공격을 하도록 구현했습니다.

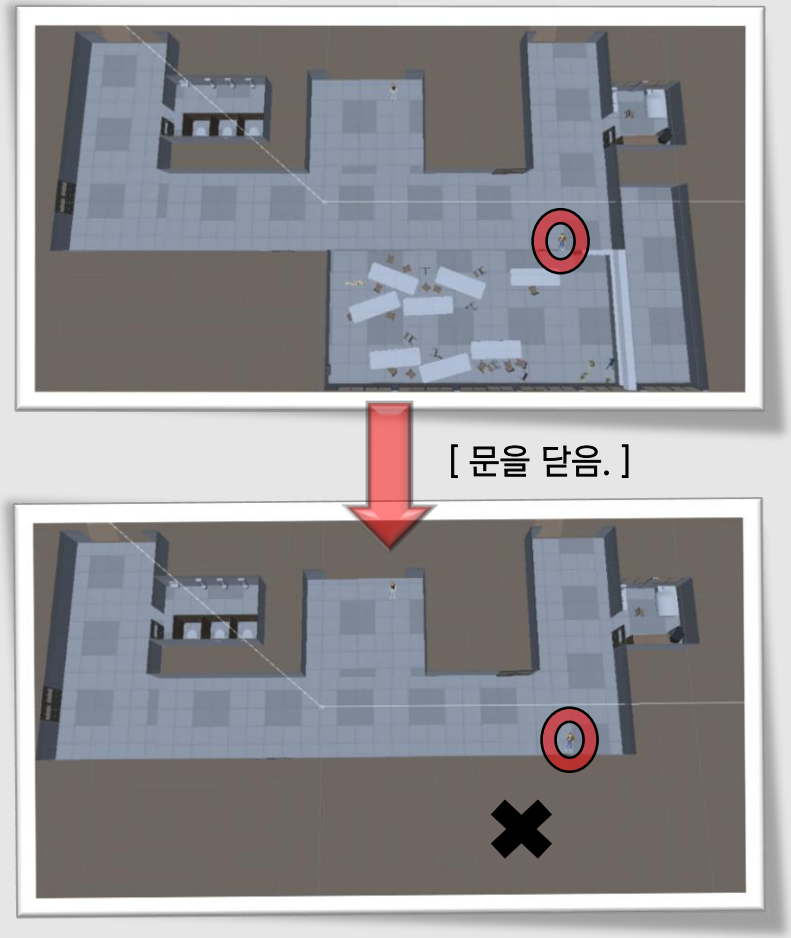
2번째 프로젝트 ‘우리는 신촌에’

주요 개발 내용 – ‘맵의 Section 및 Lighting 시스템 알고리즘 구현’

플레이어를 기준으로 막혀 있지 않은 맵 들을 렌더링할 수 있도록 Lighting 시스템의 알고리즘을 구현했습니다. 각 구역을 ‘Section’ 으로 나누어, 자신과 연결되어 있는 Section들을 리스트로 관리하도록 했습니다.

플레이어가 존재하는 Section을 중심으로 해서 Section 간에 ‘문’ 을 기준으로,
1) 문이 열리면 공간이 개방되는 것으로 판단하고 렌더링하며,
2) 닫히면 공간이 가로 막힌 것으로 판단해 렌더링하지 않습니다.

또한 Section은 자신에게 포함되어 있는 오브젝트들도 리스트로 따로 관리해서 렌더링 여부를 한꺼번에 결정하도록 구현했습니다. 렌더링을 하는 시점은 플레이어가 바닥 방향으로 매 Update마다 Raycast를 쏘아 자신을 포함하는 Section을 판단하며, 그에 따라 직접적으로 연결되어 있는 Section들에게 렌더링하라는 메소드를 호출하는 방식입니다. 만약 이미 렌더링 된 Section일 경우, flag값을 설정해 다시 렌더링 요청을 하지 않도록 했습니다.



3번째 프로젝트‘Infinity CCG’(진행중)

프로젝트 이름 : Infinity CCG
프로젝트 장르 : 게임 (TCG 대전 모바일 게임)
프로젝트 인원 : 총원 2명 (개발 인원 2명)
프로젝트 요약 : 실시간 PVP 대전 턴제 카드 게임 개발.
수행 역할 : UnityEditor를 이용해 기획자 및 번역가를 위한
스테이지 정보 자동 생성 툴 구현.
개발 환경 : C#, Unity Engine, NPOI, Unity Editor, Visual Studio
개발 기간 : 2018. 9. 1 ~ (진행 중)
버전 관리 : GitLab(Git), SourceTree

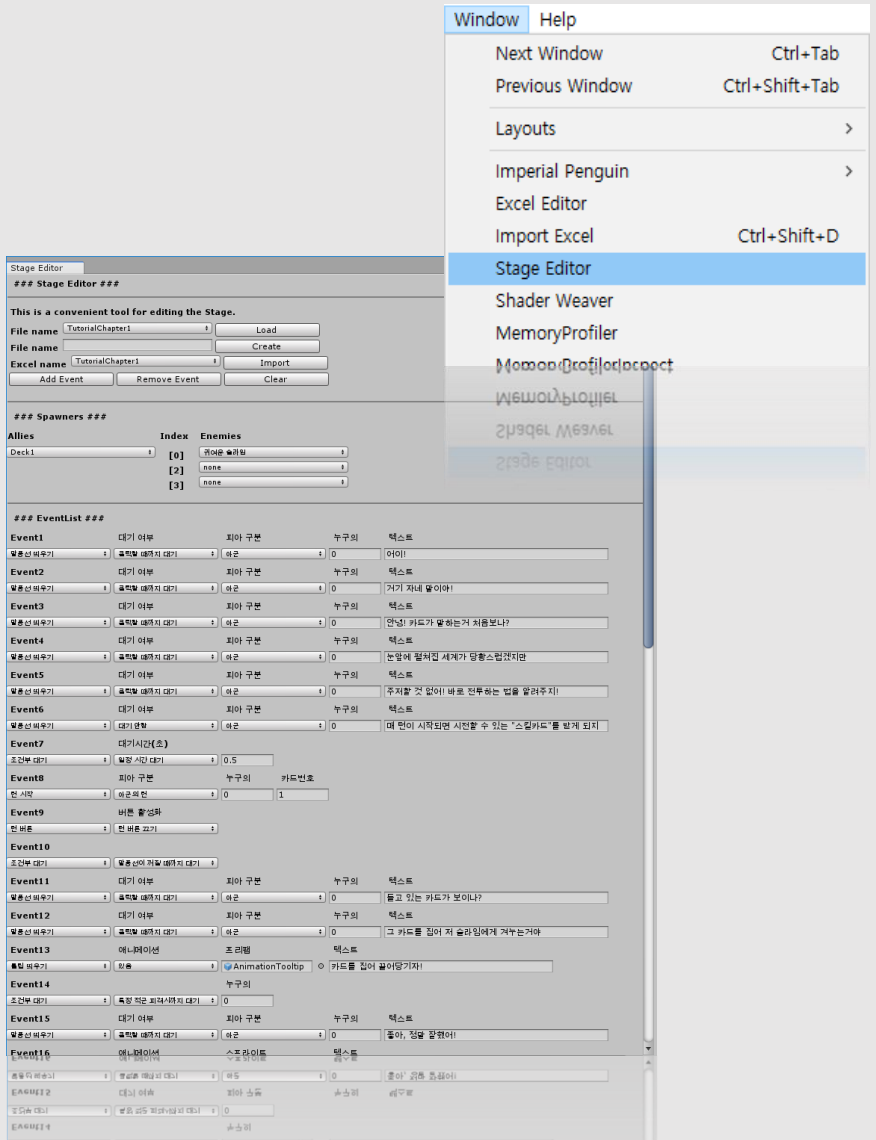
주요 개발 내용

- Unity Editor를 이용해 기획자를 위한 스테이지 정보 생성 툴 구현
- NPOI 라이브러리를 이용해 스테이지의 대사 데이터를 엑셀에 입력 및 수정 기능 구현
- 스킬 카드 기능 일부 구현

3번째 프로젝트 ‘Infinity CCG’ (진행중)

주요 개발 내용 – ‘Unity Editor를 이용해 스테이지 정보 생성 툴 구현’

해당 게임의 싱글 컨텐츠를 구현하기에 앞서 스테이지마다 매번 하드코딩 하는 것은 매우 비효율적이라고 판단해서 기획하게 됐습니다. UnityEditor 클래스를 이용해서 StageEditor라는 이름으로 구현했습니다. 이것을 이용해 간단한 편집을 통해서 스테이지에 필요한 모든 데이터들을 Binary Format으로 저장할 수 있도록 구현 했습니다. 또한 기존 스테이지 데이터를 불러와 수정 작업할 수 있도록 구현했습니다.

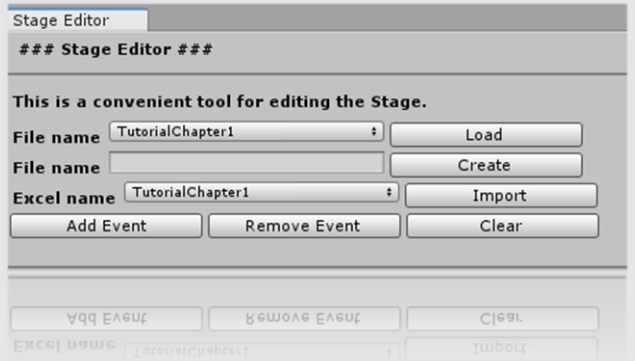


개발 초기에 ‘데이터 직렬화’에 대한 고민이 있었습니다. 처음에는 Json 형태로 구현하려고 의도 했습니다. 하지만 배열이나 리스트 같은 형태를 정상적으로 저장하지 못해 wrapping해서 사용해야 하는 한계가 있었습니다. 또한 사용자 정의 클래스나 object와 같은 타입들을 정상적으로 저장하지 못했습니다. 그래서 **모든 데이터 타입들에 대해서 저장 가능하며, 메모리적으로 효율적인 ‘Binary Format’을 채택**하게 되었습니다. 또한 프로젝트에 당장 기획자는 없었지만, 기획자를 영입했다고 가정해서 최대한 사용성이 좋은 Form을 구현할 수 있도록 노력했습니다.

3번째 프로젝트 'Infinity CCG' (진행중)

주요 개발 내용 – 'NPOI를 이용해 엑셀의 데이터 입력 및 수정 기능 구현'

앞서 언급한 StageEditor에서 대사 데이터의 입력과 함께 Create버튼을 누르면, Binary파일과 함께 엑셀에 대사 데이터가 저장되도록 구현했습니다. 또한 실제 게임에서 엑셀 데이터를 불러와 대사를 표현할 수 있도록, 엑셀 데이터 라인을 읽어와 사용할 수 있도록 구현했습니다. 엑셀 데이터는 Chapter값 기준으로, 대사 표현 순서는 Sequence 값 기준으로 표현하게 됩니다.



	A	B	C	D	E
1	Index	Chapter	Sequence	Script(kor)	Script(eng)
2	1	1	1	어이!	-
3	2	1	2	거기 자네 말이야!	-
4	3	1	3	안녕! 카드가 말하는거 처음보나?	-
5	4	1	4	눈앞에 펼쳐진 세계가 당황스럽겠지만	-
6	5	1	5	주저할 것 없어! 바로 전투하는 법을 알려주지!	-
7	6	1	6	매 턴이 시작되면 시전할 수 있는 "스킬카드"를 받게 되지	-
8	7	1	7	들고 있는 카드가 보이나?	-
9	8	1	8	그 카드를 집어 저 슬라임에게 겨누는거야	-
10	9	1	9	카드를 집어 끌어당기자!	-
11	10	1	10	좋아, 정말 잘했어!	-
12	11	1	11	적의 체력이 0 이하가 되면 게임에서 승리합니다!	-
13	12	1	12	더 이상 할게 없군.. 턴을 넘기자!	-
14	13	1	13	으악 볼거 아니군! 껄껄	-
15	14	1	14	턴이 시작되면 스킬에 필요한 마나를 2개 얻어	-
16	15	1	15	이번 카드로 적을 쓰러 뜨려 보자고!	-
17	16	1	16	앗 조금 아쉬운데? ㅋㅋ 다음을 노려보자!	-
18	17	1	17	좋아 이번에야 말로 끝내보자!	-
19	18	1	18	정말 잘했다 다음 장에서 보자!	-
20	19	2	1	오 저 멧돼지는 뭐야	-
21	20	2	2	자네 이번에도 잘할 수 있지?	-
22	21	2	3	적이 두 마리지만 우리도 두명이잖아?	-
23	22	2	4	재밌는 전투가 되겠어!	-
24	23	2	5	턴이 돌아온 캐릭터는 마나수정을 2개를 얻습니다.	-

개발 초기에는 레퍼런스가 많은 오픈소스인 'Quick Sheet'를 사용하려 했습니다. 하지만 스테이지 에디터의 경우 데이터를 엑셀에 입력해야 하는 기능이 필수적 이었고, 대다수 엑셀 연동 API는 read 위주의 기능만 제공할 뿐, write기능은 제공하지 않는 것으로 확인했습니다. 또한 Microsoft-Office의 Excel Library는 Unity 에디터와의 충돌이 문제인 것인지 정상적으로 Import하지 못하는 문제가 있었습니다. 그래서 엑셀 write 기능을 지원하면서, 충분히 유용하게 엑셀을 다룰 수 있는 API를 Searching했고 그 결과, NPOI를 채택하게 되었습니다.

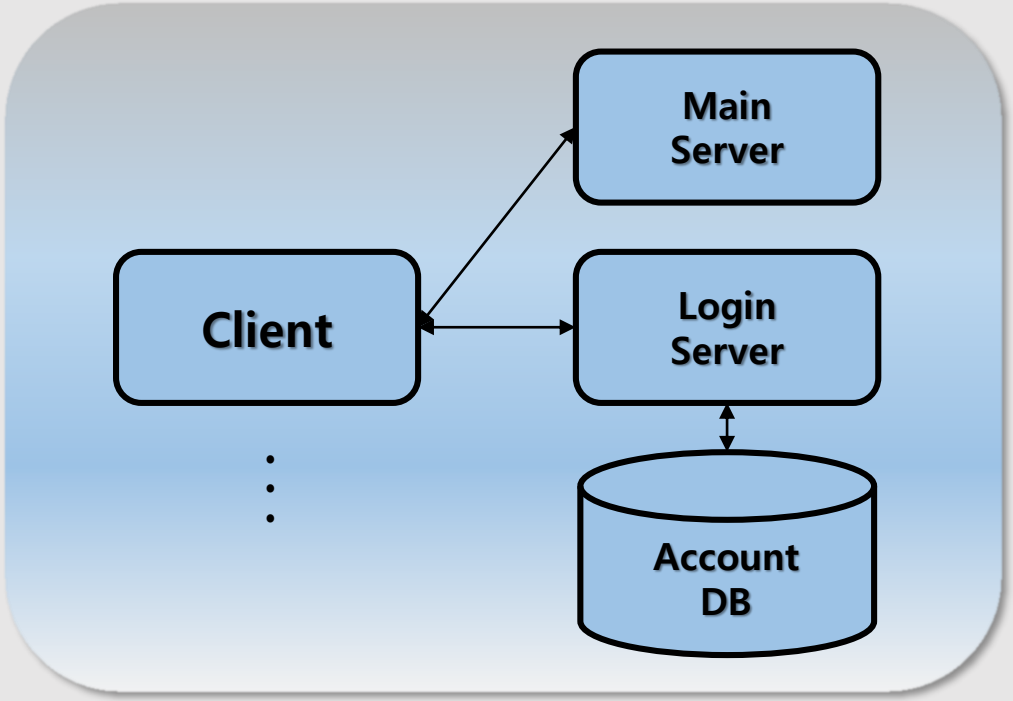
4번째 프로젝트 '참여형 교육 솔루션' (진행중)

프로젝트 이름 : 참여형 교육 솔루션
프로젝트 장르 : 메신저 프로그램
프로젝트 인원 : 총원 3명 (개발 인원 3명)
프로젝트 요약 : 수업 시간에 교수와 학생 간 원활한 소통을 위한 메신저를 개발.
수행 역할 : 서버 구축 및 데이터 통신 구현.
개발 환경 : C++, Qt, Visual Studio, WinAPI, WinSock, MySQL, Jsoncpp
개발 기간 : 2019. 3. 1 ~ (진행 중)
버전 관리 : GitHub(Git), SourceTree
주요 개발 내용
- WinSock을 활용해 소켓 프로그래밍 및 서버-클라이언트 구축
- 소켓 리디렉션 및 메시지 통신 구현
- MySQL 기반의 회원 정보 DB 구현
- Qt 기반의 GUI 환경 구현

4번째 프로젝트 '참여형 교육 솔루션' (진행중)

주요 개발 내용 – 'WinSock을 활용해 소켓 프로그래밍 및 서버-클라이언트 구축'

WinAPI, WinSock2 라이브러리를 이용해서 소켓 통신 기반의 서버를 구축했습니다.



최초에 클라이언트 실행 이후 로그인 서버에 연결되며, 소켓 할당 후 Recv Thread, Send Thread를 각 1개씩 생성해 통신하도록 구현 했습니다. Account DB는 MySQL을 통해 구현했으며, 계정정보를 데이터 테이블 형태로 가지도록 했습니다. 클라이언트가 로그인 서버에 로그인 요청을 할 경우, 로그인 서버는 DB에 입력 정보에 따라 조회를 한 후에 클라이언트에 결과를 반환하도록 했습니다. 클라이언트가 로그인 서버로부터 승인을 받으면 메시지에 포함되어 있던, IP와 Port를 통해서 Main Server에 접속을 시도하도록 했습니다.

4번째 프로젝트 ‘참여형 교육 솔루션’ (진행중)

주요 개발 내용 – ‘소켓 리디렉션 및 메시지 통신 구현’

① ‘소켓 리디렉션’ (로그인 서버에서 메인 서버로 연결 재설정) 구현 문제 해결

최초에 연결 구현 방식은 소켓 할당 후, 송수신 스레드를 각각 생성하는 것이었습니다. 같은 방식으로 클라이언트가 로그인 서버로부터 메인 서버의 IP, Port를 받았을 때, 새로 소켓을 할당하고 새로 송수신 스레드를 생성한 후에 기존의 소켓을 말소하고 스레드를 강제 종료하면 된다고 생각 했습니다. 하지만 스레드에 강제 종료를 할 경우 비정상적 종료로 인식해 프로그램이 죽어버리는 문제가 발생했습니다. 그래서 “기존의 소켓과 스레드를 재활용하자” 라는 생각으로 바꿨습니다. 하지만 기존 소켓의 IP와 Port를 강제로 재할당할 경우, 비동기로 돌고있던 수신 스레드에서 오류를 일으켰습니다. 수신 스레드에서 작동하는 WinSock의 recv()는 현재 소켓 연결정보를 기준으로 동작하기 때문이었습니다. 그래서 recv() 함수에 예외 처리를 해줌으로써 소켓 재할당을 구현했고, 최종적으로 ‘소켓 리디렉션’을 구현했습니다.

② 메시지 통신

오픈 소스인 Jsoncpp(Json Formatter)를 이용해 메시지 패키징을 구현했습니다. 클라이언트와 서버 간 모든 데이터 통신은 Json을 기반으로 식별할 수 있도록 하도록 했습니다. 메시지를 일관성 있게 JSON Parsing해서 목적에 맞게 각각의 기능을 수행할 수 있도록 메시지 통신을 구현했습니다.

감사합니다