**2021 -2 Network Game Programming**

**Term - Project**

2017182007 김우빈

2017180023 우찬희

2017184034 최정윤

목차

1. 게임 소개
2. High – Level Design
3. Low – Level Design
4. 개발 환경
5. 역할 분담
6. 개발 일정
7. 게임 소개 : Enter the Gungeon (모작)



* 게임 장르 : 배틀 로얄

- 원작의 경우 로그라이크 장르를 취하고 있으나, 네트워크 프로그래밍을 적용하기 위하여 대전 형태로 변경하였습니다.

* 플레이어 :  3명

- 체력은 총 10칸 이며, 피격 당할 시 무기의 공격력 수치만큼 감소합니다.

* 조작법

- 이동 : 키보드의 W A S D 키로 이동하며, 두개의 방향을 조합 시 대각선으로 이동할 수 있습니다.

- 구르기 : 마우스 우클릭을 하면 해당 방향으로 빠르게 구르는 모션을 취하며, 구르는 동안 무적으로 판정됩니다.

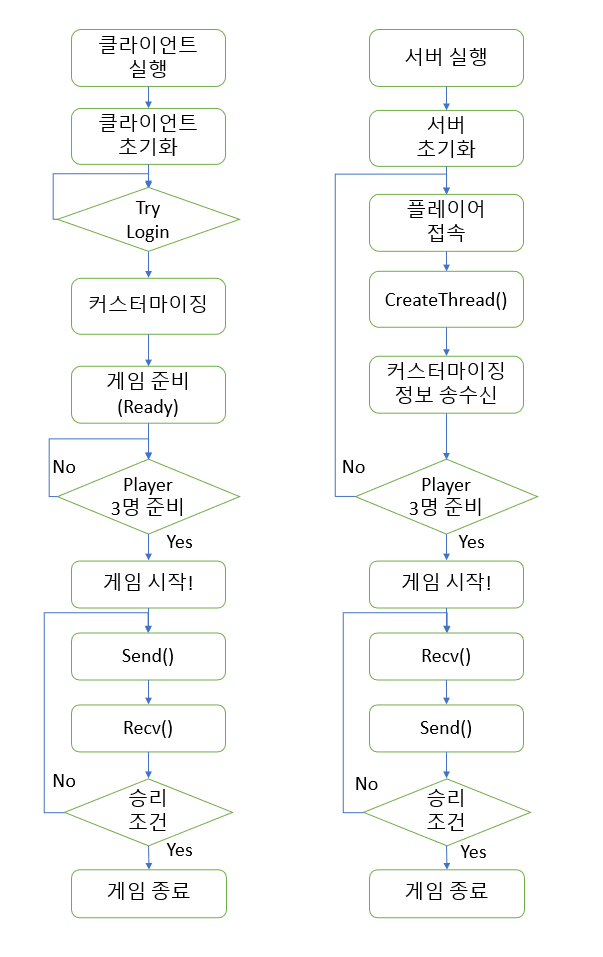
- 무기 발사 : 마우스 좌클릭을 통해 무기를 발사합니다.

* 게임 방식

- 맵 : 맵은 여러 개의 방으로 구성되어 있으며, 각 방에는 랜덤 한 무기가 담긴 보물상자가 놓여있습니다. 플레이어는 맵을 이동하며 무기를 얻고 다른 플레이어와 전투를 하게 됩니다.

- 무기 : 무기는 다양한 종류가 있으며, 무기의 공격력과 발사속도가 모두 다릅니다.

1. High – Level



* 클라이언트
* 클라이언트 실행 시 닉네임을 입력 후 서버에 로그인을 시도한다.
* 로그인 성공시 캐릭터 커스터마이징을 진행한다. (색깔)
* 커스터마이징이 완료되면 준비버튼을 눌러 다른 플레이어가 준비, 접속이 완료 될 때 까지 대기한다.
* 게임이 시작되면 클라이언트 60프레임을 기준으로 매 프레임 데이터를 송수신 한다. (PVP게임이므로 매프레임 데이터를 송수신 한다.)
* 클라이언트에선 충돌, 이동 처리를 하지 않고 키 입력만 받는다. 서버에서 처리한다.

(클라이언트를 믿을 수 없다. 어떠한 행동을 할 지 모르기 때문이다.)

* 서버
* 클라이언트가 닉네임을 보내 접속을 시도할 시 중복되는 닉네임이 있는지 검사 후 없다면 Login\_Okay 패킷을 전송한다. (실패시에는 다시 닉네임을 입력 받도록 패킷 전송)
* 로그인 성공한 플레이어의 커스터마이징 정보를 계속 송수신 하면서 3명의 플레이어 모두가 준비 완료하기를 기다린다.
* 서버에서는 모든 클라이언트의 키 입력을 받아와 서버에서 이동, 충돌 처리 후 클라이언트에게 이동한 좌표, HP, 피격 정보를 보내준다. (매 프레임 데이터를 보냄)
* 생존자가 1명이 남을 때까지 위의 과정을 수행 후 게임 결과를 보내준다.

1. Low – Level

패킷 구조체 이름에 규칙을 만들어 사용한다.

sc로 시작하는 구조체는 서버 -> 클라이언트로 전송하는 패킷 데이터 구조체이며 cs로 시작하는 구조체는 클라이언트 -> 서버로 전송하는 패킷 데이터 구조체이다.

* 서버에서 클라이언트로 보내는 패킷

텍스트이(가) 표시된 사진

자동 생성된 설명





텍스트, 시계이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

* 클라이언트에서 서버로 보내는 패킷





* 클라이언트, 서버에서 동시에 사용하는 구조체

텍스트이(가) 표시된 사진

자동 생성된 설명

* 서버에서 가지고 있는 아이템 정보

텍스트이(가) 표시된 사진

자동 생성된 설명

* 클라이언트

**- Main**

**MainApp : 메인 게임 루프의 바탕이 되는 클래스, Scene과 Manager를 이곳에서 동작 시킨다.**

**- Manager**

**DeviceManager : Direct X 9 장치를 초기화하고 Direct X Sprite 객체를 생성한다.**

**InputManager : 키보드와 마우스 입력을 제어하는 매니저 클래스.**

**SceneManager : 게임 Scene을 제어하는 매니저 클래스.**

**NetworkingManager : 네트워크 송수신할 데이터를 관리하는 매니저 클래스.**

**ResourceManager : 리소스의 로드와 저장을 담당하는 매니저 클래스.**

**GameManager : 게임 전반적인 상태와 게임 오브젝트를 관리하는 매니저 클래스.**

**Renderer : 게임 렌더링을 관리하는 매니저 클래스**

**DataLoader : Data 파일에서 파싱한 데이터를 관리하고 적용시키는 매니저 클래스.**

**FrameManager : 게임의 프레임 수를 관리하고 제한하는 매니저 클래스**

**- GameObject**

**Player : 게임 플레이어 객체, State 클래스들을 통해 상태를 관리한다**

**Weapon : Player가 소유하게 되는 무기 클래스. 파싱한 데이터를 바탕으로 생성된 무기 정보를 바탕으로 이루어져 있다.**

**Bullet : Weapon에서 생성하는 총알 클래스. 발사된 무기에 따라 공격력, 발사 속도, 외형 등이 변경된다.**

**Map : 게임이 이루어지는 배경 클래스.**

**Chest : 무기가 담겨 있는 상자 클래스. 서버에서 받은 랜덤값에 따라 담겨있는 무기가 바뀐다.**

* 서버

Client\_thread: 각 클라이언트에 할당되어 해당 클라이언트와의 데이터 송수신을 담당한다.  
 Recvn(): 클라이언트가 매 프레임마다 보낸 자신의 데이터를 수신한다.  
 RrocessPacket(): 데이터(패킷)가 저장된 버퍼를 읽어 해석, 저장한다. 대부분   
 클라이언트 상태 정보를 수신한다. 이벤트 방식으로 보호된다.  
 send(): 아래 ServerProcess()로 업데이트된 state 데이터를 클라이언트에 전송한다.

ServerProcess(): 주 스레드에서 동작한다. 세 클라이언트 쓰레드에서 ProcessPacket()이 끝나면 이벤트 감지를 통해 실행되며 새로 변경된 데이터로 충돌체크 등의 연산을 실행하여 game\_state를 업데이트한다. 이후 event로 클라이언트 스레드를 깨워 send()를 실행하게 한다.

1. 개발 환경

* Windows 11 Pro
* Visual Studio 2019
* DirectX 9
* Windows SDK 10.0.19041.0
* GitHub (프로젝트 관리)

1. 역할 분담

|  |  |
| --- | --- |
| 김우빈 | * 서버 루프 제작 * 클라이언트와의 데이터 송수신 * 이모지 * 스레드 동기화 |
| 우찬희 | * 서버 프레임 워크 * 이모지 * 충돌 * 스레드 동기화 구현 |
| 최정윤 | * 게임 리소스 수집 * 게임 내부 전반적인 월드, 프레임워크 제작 * 서 와의 통신 * 이모지 |

1. 개발 일정 (상세)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **월** | **화** | **수** | **목** | **금** | **토** | **일** |
|  |  |  | **11월 3일** | **4일** | **5일** | **6일** | **7일** |
| **김우빈** |  |  | 리소스 정리 | 클라이언트 제작 | | | 진행상황 확인 및 수정 |
| **우찬희** |
| **최정윤** | 맵 제작 |
|  | **8일** | **9일** | **10일** | **11일** | **12일** | **13일** | **14일** |
| **김우빈** |  |  |  | 서버 인게임 메인 루프 정리 | | | 진행상황 확인 및 수정 |
| **우찬희** | 포트포워딩 연결 확인 | 클라이언트 연결 및 로그인 |  | 로비 제작 |
| **최정윤** | 맵 데이터 로드 | 서버 구조체 정리 |  | 예상되는 Data Race 정리 |
|  | **15일** | **16일** | **17일** | **18일** | **19일** | **20일** | **21일** |
| **김우빈** |  |  | 플레이어 인게임 초기 정보 송신 | 플레이어 데이터 수신 |  | Game\_state 송신 | 진행상황 확인 및 수정 |
| **우찬희** |  | 플레이어 - 지형 충돌체크 후 처리 | | 아이템 충돌체크 (무기 변경 |
| **최정윤** | 서버 오브젝트 생성 | 플레이어 데이터 변화 체크 후 업데이트 | |  |
|  | **22일** | **23일** | **24일** | **25일** | **26일** | **27일** | **28일** |
| **김우빈** |  |  | 지형 – bullet 충돌체크 후 처리 | | 죽은 플레이어 관전 처리 |  | 진행상황 확인 및 수정 |
| **우찬희** | 플레이어 – 플레이어 충돌체크 후 처리 |  | 게임 승리 조건 체크 및 게임 결과 송신 | |
| **최정윤** |  | 플레이어 – bullet 충돌체크 후 처리 | | |
|  | **29일** | **30일** | **12월1일** | **2일** | **3일** | **4일** | **5일** |
| **김우빈** |  |  | 이모지 제작 | 이모지 제작 | | 추가 구현 및 디버깅 | |
| **우찬희** |
| **최정윤** | 디버깅용 무적모드 제작 |
|  | **6일** | **7일** | **8일** | **9일** |  |  |  |
| **김우빈** | 추가 구현 및 디버깅 | 발표자료 작성 및 발표 준비 | | |  |  |  |
| **우찬희** |
| **최정윤** |