**추진 계획서**



2020182048 김정학

2018184031 조창근

2018184032 조현석

**목차**

1. 프로그램 소개\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. HIGH-LEVEL 디자인\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
3. 서버 플로우 차트
4. 클라이언트 플로우 차트
5. LOW-LEVEL 디자인\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
6. 패킷 디자인
7. 상수 및 클래스 디자인
8. 서버 함수
9. 클라이언트 함수
10. 팀원 별 역할분담\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
11. 개발 환경\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
12. 개인별 일정\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
    * + 1. 프로그램 소개

* 인게임 예시

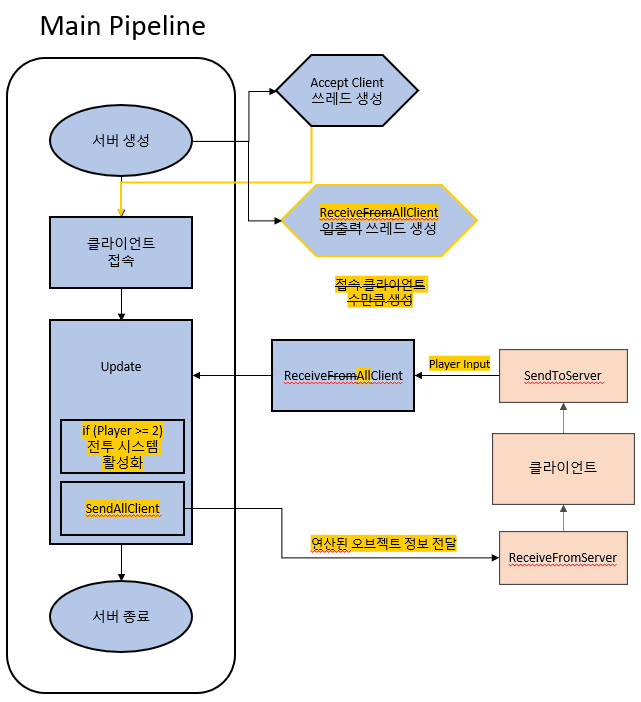


|  |  |
| --- | --- |
| 타이틀명 | “Heli World!” |
| 프로그램 개요 | 조창근 학우가 3D게임 프로그래밍 강의를 수강하면서 DirectX를 활용한 실습을 기반으로 재구성한 3D TPS 헬기 슈팅게임  (최소 2명) 최대 4명의 플레이어들이 미사일과 무작위로 스폰되는  아이템들을 이용한 데스매치 방식의 게임 |
| 조작법 | 플레이어 회전: 마우스 클릭 후 이동  플레이어 이동: WASD  플레이어 상승/하강: Q/E  미사일 발사: Space |

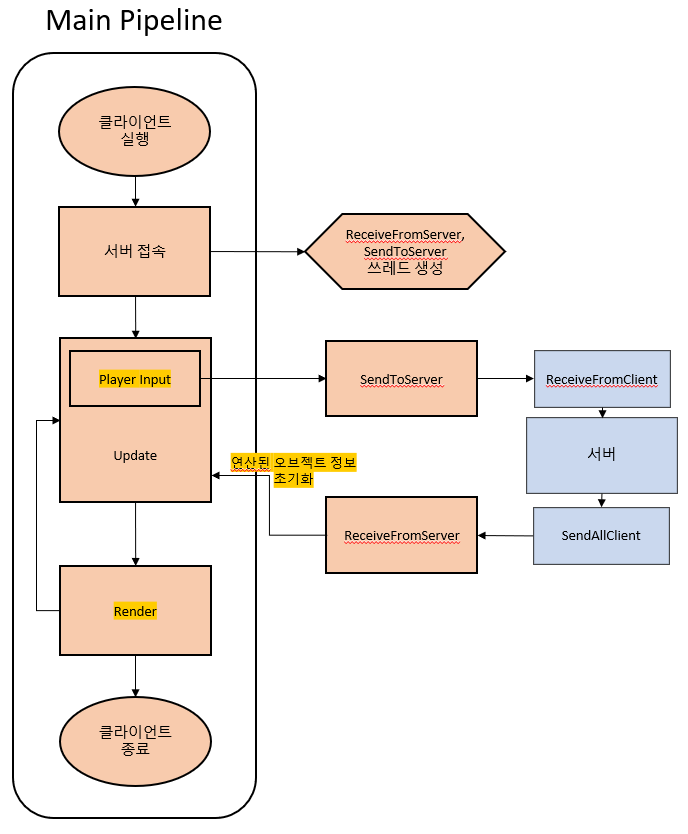
|  |  |
| --- | --- |
| 게임 구조 | Player Object  1. 4X4 World Transform Matrix  2. 충돌박스  3. HP  4. 미사일 발사 여부  Missile Object  1. 4X4 World Transform Matrix  2. 충돌박스  3. 활성화 여부  4. 충돌 및 폭발 여부  Item Object  1. 4X4 World Transform Matrix  2. 충돌박스  3. 활성화 여부 |
| 플레이 방법 | 1. 클라이언트를 실행하여 서버에 접속  2. 플레이어가 한 명이면 아이템 생성X, 기본 움직임만 가능  3. 최소 두 명의 플레이어가 입장하면 아이템 생성, 전투 가능  4. 최대 네 명의 플레이어가 미사일과 아이템을 활용하여 배틀 |

* + - 1. High-Level

1. 서버 플로우 차트



1. 클라이언트 플로우 차트



* + - 1. Low-Level

1. 패킷 디자인

* Client to Server Packet

**PlayerKeyPacket**

~~int~~ char PlayerKeyInput

POINT MousePosition

- 플레이어의 키 입력 정보를 가지고 있는 패킷

* Server to Client Packet

\* 서버가 전송한 패킷을 클라이언트에서 수신 시, 패킷 종류를 알 수 있는 식별 타입 보유

**Player~~Transform~~InfoPacket**

char PacketType

char PlayerNumber

~~Float4X4 PlayerTransformMatrix~~

float pitch, yaw, roll

float xDelta, yDelta, zDelta

**Player~~Info~~StatusPacket**

char PacketType

char PlayerNumber

char ActivatedMissiles

int PlayerHP

- 접속한 각 플레이어의 ID와 HP정보, 미사일 활성화 정보를 가지고 있는 패킷

- 접속한 각 플레이어의 ID, 회전 및 이동의 크기를 가지고 있는 패킷

**ItemInfoPacket**

char PacketType

~~char ItemType~~

~~Float4X4 ItemTransformMatrix~~

float x, y, z

bool Active

**MissileInfoPacket**

char PacketType

char PlayerNumber

~~Float4X4 MissileTransformMatrix~~

float pitch, yaw, roll

float xDelta, yDelta, zDelta

- 아이템의 종류와 위치, 활성화 정보를 가지고 있는 패킷

- 총 8개의 미사일 활성화 상태를 각각 1bit로 True/False 판별

- 미사일을 발사한 플레이어의 ID, 회전 및 이동의 크기를 가지고 있는 패킷

1. 상수 및 클래스 디자인

//서버에서 클라이언트로

//전송하는 패킷 타입

텍스트이(가) 표시된 사진

자동 생성된 설명

//클라이언트와 서버 간의

//미리 지정된 키 매크로

텍스트이(가) 표시된 사진

자동 생성된 설명

//게임 오브젝트, 베이스 클래스

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

1. 서버 함수

* DWORD WINAPI AcceptClient(LPVOID arg)
* 접속을 시도하는 클라이언트를 서버에 연결하고 해당 플레이어를 playerDataMap 컨테이너에 추가(playerDataMap는 unordered\_map 컨테이너 사용)
* Accept하는 경우 Receive~~From~~AllClient Thread 생성
* DWORD WINAPI Receive~~From~~AllClient(LPVOID arg)
* 각 클라이언트들의 키 입력 정보를 수신하는 함수
* 각 클라이언트로부터 키 패킷을 수신 도중 Update가 실행되지 않도록 모든 키를 수신 완료 후 이벤트를 신호상태로 바꾸고, ReceiveAllClient 시작 시에 이벤트를 비신호상태로 바꾸어 Update와 ReceiveAllClient를 동기화한다.
* void Update(float elapsedTime)
* 오브젝트들의 World Transform Matrix 연산 및 충돌 확인, 오브젝트들의 상태와 아이템 획득여부 업데이트 및 아이템 오브젝트 생성
* ~~Update 함수 실행 중에 SendAllClient 함수가 실행되지 않도록 동기화~~
* ~~DWORD WINAPI~~ void SendAllClient(~~LPVOID arg~~)
* Update된 오브젝트들의 연산 결과를 패킷화 후 접속한 각 클라이언트에게 전송
* ~~SendAllClient 함수가 진행되는 동안 Update 함수 연산으로 인한 값 변경이 일어나지 않도록 동기화~~

1. 클라이언트 함수

* bool ConnectServer(SOCKET\* sock, char\* serverIP,

u\_short portNumber)

* 소켓 생성 후 서버에 연결 시도
* 서버 연결 성공 시 ReceiveFromServer 쓰레드 생성
* DWORD WINAPI SendToServer(LPVOID arg)
* ~~키~~ 플레이어 입력이 있을 시에 입력 정보를 서버로 전송
* DWORD WINAPI ReceiveFromServer(LPVOID arg)
* 서버 접속 성공 후 서버로부터 PlayerInfoPacket을 수신
* 서버로부터 받은 패킷을 패킷 타입에 따라 분류하여 ~~오브젝트들의 위치와~~ 오브젝트들의 정보를 갱신
* 오브젝트들의 정보를 수신 도중 Update가 실행되지 않도록 모든 패킷을 수신 완료 시 이벤트를 신호 상태로 변경, ReceiveFromServer 시작 시 이벤트를 비신호상태로 변경 ReceiveFromServer와 Update를 동기화한다.
* void Update()
* 환경 요소(스카이 박스, 빌보드, 터레인)를 업데이트
* 서버로부터 받아온 변화량을 토대로 오브젝트들의 World Matrix 연산
* void Render()
* 갱신된 정보를 가지고 오브젝트들을 렌더링
  + - 1. 개인 별 역할 분담

|  |  |
| --- | --- |
| 공통 | ~~패킷 디자인,~~ 서버 연산 코드 작성, 플레이 테스트 |
| 김정학 | ~~서버 내에서 Update, SendAllClient 함수가 동시에 공유 자원에~~  ~~접근할 수 없도록 동기화~~  PlayerInfoPacket, MissileInfoPacket 작성  SendAllClient 함수 구현  ReceiveFromServer 함수 구현  ReceiveFromServer와 Update함수(클라이언트) 간의 동기화 |
| 조창근 | 클라이언트에 멀티플레이가 가능하도록 재구성  서버로부터 받은 데이터를 클라이언트 렌더링이 가능하도록  클라이언트 오브젝트의 정보 갱신 구현 |
| 조현석 | PlayerStatusPacket, ItemInfoPacket 작성  서버 내에서 Update 함수 구현  ReceiveAllClient 함수 구현  ReceiveAllClient와 Update함수(서버) 간의 동기화 |

* + - 1. 개발 환경

|  |  |
| --- | --- |
| OS | Window 11 |
| IDE | VisualStudio 2022 |
| API | DirectX12 |

6. 개발 일정

|  |  |  |  |
| --- | --- | --- | --- |
|  | **김정학** | **조창근** | **조현석** |
| ~11/8 | 조건부 패스로 인한 문서 수정 작업 진행 | | |
| ☑11/9 | ☑PlayerInfoPacket 작성 | ☑ConnectServer, ☑AcceptClient  함수 구현 | ☑PlayerStatusPacket  작성 |
| ☑11/10 |  |  |  |
| ☑11/11 | ☑Update(클라이언트) 구현 |  |  |
| ☑11/12 | ☑키 매크로 작성 및  ☑SendToServer 함수 구현 | ☑Receive~~From~~AllClient  함수 구현 |
| ☑11/13 | ☑주차 계획 점검 | | |
| 11/14 |  |  |  |
| ☑11/15 | ☑SendAllClient,  ☑ReceiveFromServer  함수 구현  ☑SendAllClient ReceiveFromServer 형식 정리 필요 | ☑ReceiveFromServer 함수와  ☑렌더링 함수 연결 | ☑서버 Update 함수 구현 시작 |
| ☑11/16 | ☑서버 Update 함수 구현  충돌반영 | |
| ☑11/17 |
| ☑11/18 | ☑충돌 관련 테스트 및 디버깅 | | |
| ☑11/19 |
| ☑11/20 | ☑주차 계획 점검 | | |
| 11/21 |  |  |  |
| ☑11/22 | ☑MissileInfoPacket 작성 |  | ☑ItemInfoPacket 작성 |
| ☑11/23 |  | ☑ReceiveFromServer 함수에  MissileInfoPacket, ItemInfoPacket 수신 구현 |  |
| 11/24 | ☑ReceiveFromServer와  ~~Update(클라이언트)의 동기화~~ | ☑Receive~~All~~FromClient~~와~~  ~~Update(서버)의~~ 끼리의 동기화 |
| 11/25 | ☑클라이언트 오브젝트  렌더링 테스트 |
| 11/26 | ☑서버 Update 함수  미사일 충돌처리 기능 구현 |  | ☑서버 Update 함수  아이템 상호작용 기능 |
| 11/27 | 주차 계획 점검 | | |
| 11/28 |  |  |  |
| 11/29 | SendAllClient 함수  부가 기능 패킷 전송 반영 | 클라이언트 부가 기능 구현 | 부가 기능 관련 패킷 작성 |
| 11/30 |
| 12/1 |  |  |  |
| 12/2~ | 테스트 및 디버깅 | | |