**네트워크 게임 프로그래밍 프로젝트 추진서**

2019182002 경태환

2019180008 김상효

**목차**

1. **애플리케이션 기획**
2. **High-Level 디자인**
3. **Low-Level 디자인**
4. **팀원 별 역할 분담**
5. **개발 일정**

**1.애플리케이션 기획**

**게임 소개**

- 게임 이름 : 1945

- 장르 : 슈팅게임

- 플레이어 : 2p

2019182002 경태환 학생이 윈도우 프로그래밍 수업시간에 만든 1945 스트라이커 모작 게임. 플레이어는 이동하며 총알을 피하고 자동으로 발사되는 총알로 몬스터를 맞춰 파괴하는 게임. Window API 프로그래밍을 기반으로 제작되었고 동시에 2명의 플레이어가 접속하여 협동을 통해 몬스터를 파괴한다. 최종적으로 보스 몬스터를 파괴하면 게임이 끝나게 된다. 몬스터가 발사한 총알에 플레이어가 닿으면 목숨을 잃게 되고, 목숨을 모두 잃은 플레이어는 게임을 플레이 할 수 없다. 모든 플레이어가 목숨을 모두 잃으면 게임 오버.

**조작 방식**

* 방향키 : 플레이어 이동
* 총알 발사 : 일정 시간마다 자동으로 발사

**개발 환경**

* VisualStudio 2022
* Git
* Window API

**기존 게임 플레이 스크린샷**

**스크린샷, PC 게임, 비디오 게임 소프트웨어, 전략 비디오 게임이(가) 표시된 사진

자동 생성된 설명**

**2. High-Level 디자인**

**흐름도**

텍스트, 스크린샷, 도표, 번호이(가) 표시된 사진

자동 생성된 설명텍스트, 도표, 스크린샷, 평면도이(가) 표시된 사진

자동 생성된 설명

**정리**

**로고 레벨**

* **서버**

1. 서버 프로그램 실행
2. 플레이어 접속을 받는 리스닝 소켓 생성
3. 클라이언트 인원이 2명 접속 시 확인 후 레벨 변경

* **클라이언트**

1. 클라이언트 프로그램 실행
2. 로딩 화면에서 서버에 접속
3. 서버로부터 레벨 변경 정보를 받기 전 까지 대기

**메뉴 레벨**

* **서버**

1. 메뉴레벨 변경 여부를 클라이언트에 전달
2. 클라이언트로부터 캐릭터 선택 데이터를 수신
   1. 실시간으로 다른 클라이언트의 선택 정보 데이터를 클라이언트에게 보냄
3. 클라이언트로부터 게임 시작 정보를 수신

* **클라이언트**

1. 서버로부터 메뉴레벨 변경 여부를 받아서 레벨 변경
2. 캐릭터 선택 메뉴 렌더링
   1. 실시간으로 전달받은 다른 클라이언트의 정보를 이용하여 선택 캐릭터 표시
3. 캐릭터 선택 시 입력 데이터를 서버에 전송
4. 게임 시작 버튼 입력 여부를 서버에 전송

**게임플레이 레벨**

* **서버**
* 메인 쓰레드
  + 클라이언트로 전달받은 입력 데이터를 통해서 플레이어 이동, 오브젝트 생성 및 관리, 충돌처리 담당
* 쓰레드 1
  + 클라이언트 1과 통신하여 데이터를 수신
* 쓰레드 2
  + 클라이언트 2와 통신하여 데이터를 수신
* 쓰레드 3
  + 업데이트 된 게임 상태 데이터(오브젝트, 점수 등)를 모든 클라이언트에게 전송
* 동기화 설정
  + 클라이언트로부터 받아오는 키보드 입력 정보, 총알 발사 여부를 Mutex로 보호(클라이언트로 데이터를 받아와 저장하는 쓰레드와 메인 쓰레드에서 이 입력을 처리하기 위해 입력 받은 변수에 접근하면서 발생하는 Data Race방지)
  + 서버에서 처리하는 오브젝트들을 Mutex로 보호(서버에서 메인 쓰레드가 오브젝트들의 업데이트를 수행하는 도중 클라이언트에 데이터를 전송하는 쓰레드에서 오브젝트의 리스트에 접근해 발생할 수 있는 Data Race방지)
* **클라이언트**
* 메인 쓰레드
  + 키 인풋, 총알 발사 시 서버로 데이터 전송
  + 서버에서 처리된 데이터를 받아와서 오브젝트들을 갱신,
  + 서버로 보내지 않을 데이터들은 클라이언트에서 직접 처리
  + 갱신된 오브젝트들을 렌더링
  + 싱긅톤으로 서버와의 통신을 담당하는 클래스 객체를 만들어서 서버와 통신
  + 게임 종료 확인 시 필요한 정보를 출력하고 게임 종료

**기존 게임의 프레임워크**

**텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명**

**변경된 클라이언트 프레임워크**

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

**3. Low-Level 디자인**

**네트워크 구조 및 통신 프로토콜**

네트워크 모델 : 클라이언트 – 서버 모델

통신 프로토콜 : TCP

**전송할 데이터 구조체**

**클라이언트 -> 서버로 전송할 데이터 구조체**

|  |
| --- |
| 메뉴레벨에서 선택한 캐릭터 정보 |
| uint8\_t CharacterID; |
| 플레이어에 대한 키 입력, 총알 발사 여부 처리 |
| struct Key\_Input{  bool Left;  bool Right;  bool Up;  bool Down;  bool Shoot;  } |

**서버 -> 클라이언트로 전송할 데이터 구조체**

**이벤트(움직임)가 발생할 때에만 알맞은 데이터를 전송하기 위해 enum으로 이벤트 관리(enum 은 uint8\_t을 통해 1Byte로 이용)**

|  |
| --- |
| 서버 -> 클라이언트로 보낼 구조체에 포함할 enum값(uint8\_t) |
| enum EventType : uint8\_t {  PlayerChoice,  PlayerMoved,  MonsterSpawned,  MonsterDeleted,  BulletCollision,  MonsterBulletFired  HealthUpdate,  ScoreUpdate,  LevelChange,  GameEnd  }; |

**이벤트에 맞는 데이터 구조체**

|  |  |
| --- | --- |
| **PlayerChoice** | **상대 플레이어가 선택한 캐릭터(메뉴레벨)** |
| struct PlayerChoicePacket{  EventType Event;  uint8\_t Choiced\_Character;  } |
| **PlayerMoved** | **전송받은 키 입력으로 처리된 플레이어 위치** |
| struct PlayerMovePacket{  EventType Event; bool Player\_ID  float X;  float Y;  } |
| **MonsterSpawned** | **몬스터의 스폰 여부** |
| struct MonsterSpawnedPacket{  EventType Event;  uint8\_t Monster\_ID;  } |
| **MonsterDeleted** | **몬스터의 삭제 여부** |
| struct MonsterDeletedPacket{  EventType Event;  uint8\_t Monster\_ID;  } |
| **BulletCollision** | **플레이어 총알 -> 몬스터 충돌 여부(총알 삭제)** |
| struct BulletCollisionPacket{  EventType Event;  uint8\_t Bullet\_ID;  } |
| **MonsterBulletFired** | **몬스터 총알 발사 여부** |
| struct MonsterBulletPacket{  EventType Event;  uint8\_t Bullet\_ID;  } |
| **HealthUpdate** | **플레이어 체력 감소 여부** |
| struct HealthUpdatePacket{  EventType Event;  bool hit;  } |
| **ScoreUpdate** | **플레이어 스코어 업데이트** |
| struct ScoreUpdatePacket{  EventType Event;  unsigned short Score;  } |
| **LevelChange** | **레벨 변경 이벤트 여부** |
| struct LevelChangePacket{  EventType Event;  Uint8\_t LevelID;  } |
| **GameEnd** | **게임 종료 여부** |
| struct GameEndPacket{  EventType Event;  bool isEnd;  } |

**클라이언트 함수, 변수**

|  |  |  |
| --- | --- | --- |
| **클라이언트 서버 패킷 송수신 객체 생성(싱글톤)** | **class ServerConnection{}** | |
| **데이터 전송 함수** | **ServerConnection::getInstance()-> SendData(EventID);** | |
| **데이터 수신 함수** | **ServerConnection::getInstance()-> RecieveData();** | |
| **데이터 처리 함수** | **ServerConnection::getInstance()-> UpdateData(EventID);** | |
| **클라이언트 메인 루프 초기화** | **CMainGame.Initialize()** | |
| **클라이언트 메인 루프** | 1. **CMainGame.Update()** 2. **ServerConnection::getInstance()-> RecieveData();** 3. **CMainGame.LateUpdate();** 4. **CMainGame.Render()** | |
| **오브젝트 관리**  **(싱글톤)** | **CObjectManager();** | |
| **총알** | **m\_ObjectList[BULLET];** |
| **몬스터** | **m\_ObjectList[MONSTER];** |
| **보스** | **m\_ObjectList[BOSS];** |
| **플레이어1** | **m\_ObjectList[PLAYER1];** |
| **플레이어 2** | **m\_ObjectList[PLAYER2];** |
| **레벨(Scene) 관리(싱글톤)** | **CLevelManager();** | |
| **키 입력 관리**  **(싱글톤)** | **CKey\_Manager();** | |
| **리소스 관리**  **(싱글톤)** | **CBmp\_Manager();** | |

**서버**

**메인 쓰레드 (main())**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **오브젝트 관리**  **(싱글톤)** | **CObjectManager();** | | | | |
| **총알** | | | | **m\_ObjectList[BULLET];** |
| **몬스터** | | | | **m\_ObjectList[MONSTER];** |
| **보스** | | | | **m\_ObjectList[BOSS];** |
| **플레이어1** | | | | **m\_ObjectList[PLAYER1];** |
| **플레이어 2** | | | | **m\_ObjectList[PLAYER2];** |
| **레벨 관리** | | **CLevelManager();** | | | |
| **임계 영역 관리** | | **Mutex 변수 사용** | | | |
| 1. **클라이언트로부터 받아온 키 입력 구조체** 2. **업데이트된 오브젝트들의 리스트** | | | |
| **충돌 처리 관리** | | | **CCollisionManager();** | | |
| **관리로직** | CCollision\_Manager::Collision\_Rect  (list<CGameObject\*> \_Dst, list<CGameObject\*> \_Src)   * 오브젝트의 리스트끼리 사각형 충돌을 확인한다. | |

**데이터 수신 쓰레드 2종 – 벡터로 관리(std::vector<std::thread> ClientThread)**

|  |  |  |
| --- | --- | --- |
| **데이터 수신 함수** | **receiveDataFromClient(ClientID)** | |
| **받은 데이터 구조체(임계영역 설정)** | **메뉴 레벨** | uint8\_t CharacterID; |
| **게임플레이 레벨** | struct Key\_Input{  bool Left;  bool Right;  bool Up;  bool Down;  bool Shoot;  } |

**데이터 전송 쓰레드(SendThread)**

|  |  |
| --- | --- |
| **데이터 전송 함수** | **sendDataToClients();** |
| **보낼 데이터 구조체** | **위쪽에 enum값으로 구분되는 이벤트에 따른 데이터 선택 후 전송** |

**4. 팀원 별 역할 분담**

|  |  |
| --- | --- |
| 경태환 | 김상효 |
| (서버)  서버 기본 프레임워크 제작 | **(클라이언트)**  **클라이언트 기본 구조 및**  **프레임워크 수정** |
| (클라이언트)  클라이언트 -> 서버  연결 로직 구현  서버 통신용 싱글톤 객체 구현 | **(클라이언트)**  **메뉴 레벨 변경 여부**  **(캐릭터 선택, 게임 시작 유무)**  **서버에 전달** |
| (서버)  메뉴 레벨 변경 여부 전달  메뉴레벨 정보 수신 및 저장 | **(클라이언트)**  **메뉴에서 서버에서 전달받은 정보를 바탕으로 캐릭터 표시** |
| (서버)  게임 플레이 레벨에서 사용할  메인 쓰레드 및 다른 쓰레드 제작 | **(클라이언트)**  **키 인풋, 총알발사 시**  **서버로 데이터 전송** |
| (클라이언트)  게임 종료 확인 시  필요한 정보를 출력 및 게임 종료 | **(클라이언트)**  **서버에서 받은 정보들을 바탕으로 업데이트 및 렌더링** |
| (서버)  플레이어 이동 및  게임 상태 데이터를 모든  클라이언트에 전송 | **(서버)**  **충돌처리** |
| (서버)  동기화 설정 | **(서버)**  **플레이 레벨에서 필요한 오브젝트**  **생성 및 삭제 관리** |

**5. 개발 일정**

**-김상효-**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **11월** | | | | | | |
| **월** | **화** | **수** | **목** | **금** | **토** | **일** |
|  |  |  |  | **1** | **2** | **3** |
|  |  |  |  | 클라구조  파악 | 클라 구조 파악 | 클라 구조 파악 |
| **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| 클라이언트 기본  구조 및 프레임  워크 수정  Client::Update(), | 클라이언트 기본  구조 및 프레임  워크 수정  Client::Late\_  Update() | (클라이언트)  메뉴 레벨에  Player2 추가  및 업데이트  Client::Render(); | (클라이언트)  메뉴 레벨에  Player2 렌더링  Client::Menu:: Update(),  Render() | (클라이언트)  메뉴 레벨 변경  여부(캐릭터 선택) 서버에 전달  Server Connection::  getInstance() -> SendData (EventID); | (클라이언트)  메뉴 레벨 변경  여부(캐릭터 선택) 서버에 전달  Server Connection::  getInstance() -> SendData (EventID); | 일주일 간 작업 한 부분 수정 및 보완 |
| **11** | **12** | **13** | **14** | **15** | **16** | **17** |
| (클라이언트)  키 인풋,  총알 발사 시  서버로 데이터  전송  Client::  Key\_Input();  Server Connection::  getInstance() -> SendData (EventID); | (클라이언트)서버  에서 받은 플레이어 이동정보를 바탕으로 렌더링  Client::  Server Connection::  getInstance() -> Receive  Data (EventID);  Server Connection:: getInstance() -> UpdateData (EventID); | (서버) 몬스터 생성 로직 만들기  Server::  Create\_ Monster(); | (서버) 몬스터 생성 로직 만들기  Server::  Create\_ Monster(); | (서버)  몬스터 – 플레이어 총알  충돌처리 구현  Server::  Collision::  Monster\_ PBullet\_ Collision() | (서버)  플레이어 –  몬스터 총알  충돌처리 구현  Server::  Collision::  Player\_ MBullet\_ Collision() | 일주일 간 작업한 부분 수정 및 보완 |
| **18** | **19** | **20** | **21** | **22** | **23** | **24** |
| (클라이언트)  서버에서 몬스터  생성 관련 업데이트를 하면 몬스터 추가하기  Client::  Server Connection::  getInstance() -> Receive  Data(EventID); Server Connection:: getInstance() -> UpdateData (EventID);  Client:: Add\_Object() | (클라이언트) 서버에서 몬스터  총알 생성을 하면 몬스터 총알 추가하기  Client::  Server Connection::  getInstance() -> Receive  Data(EventID); Server Connection:: getInstance() -> UpdateData (EventID);  Client:: Add\_Bullet() | (클라이언트)추가  된 몬스터 및 몬스터 총알 업데이트 및 렌더링  Client::  Update\_Monster  (), Update\_Bullet  (),Render\_Mons ter(),Render\_ Bullets(); | (클라이언트)총알  및 몬스터 충돌  정보를 서버에서  받아 삭제 처리  Client::  Server Connection::  getInstance() -> Receive  Data(EventID); Server Connection:: getInstance() -> UpdateData (EventID);  Erase\_Bullets();  Erase\_Monsters() | (클라이언트)  플레이어 체력  데이터 서버로부터 받아  플레이어 체력갱신  Client::  Server Connection::  getInstance() -> Receive  Data(EventID); Server Connection:: getInstance() -> UpdateData (EventID);  Update\_Player() | (클라이언트)  플레이어 사망 시 렌더링 할 리소스 편집 및 만들기 | 일주일 간 작업한 부분 수정 및 보완 |
| **25** | **26** | **27** | **28** | **29** | **30** | **12/01** |
| (서버)  보스 생성 로직  추가  Server::  Create\_ Boss(); | (클라이언트)  보스 관련 데이터 받아와서 업데이트하기  Client::  Server Connection::  getInstance() -> Receive  Data(EventID); Server Connection:: getInstance() -> UpdateData (EventID);  Add\_Boss();  Update\_Boss() | (클라이언트)  보스 관련 데이터 받아와서  렌더링 하기  Client::  Server Connection::  getInstance() -> Receive  Data(EventID); Server Connection:: getInstance() -> UpdateData (EventID);  Render\_Boss() | (클라이언트)  서버 구조 안 바꾸고 클라에서  추가할 만 한 것들(아이템) 추가  Create\_Items(); | (클라이언트)  서버 구조 안 바꾸고 클라에서  추가할 만 한 것들(아이템) 추가  Create\_Items(); | (서버,클라)  부족했던 부분 보강 | 한 주간 부족했던 부분 구현 |
| **12월** | | | | | | |
| **2** | **3** | **4** | **5** |  |  |  |
| 버그 수정, 미흡했던 점 보충,  발표 준비 | | | |  |  |  |

**-경태환-**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **11월** | | | | | | |
| **월** | **화** | **수** | **목** | **금** | **토** | **일** |
|  |  |  |  | **1** | **2** | **3** |
|  |  |  |  | (서버)프레임워크  제작(레벨 매니저, 레벨 클래스 제작) | (서버)프레임워크 제작(오브젝트 매니저, 게임 오브젝트 제작) | (서버) 클라이언트에서 구현된 오브젝트 객체들 제작 |
| **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| (서버) 클라이언트에서 구현된 오브젝트 객체들 제작 | (서버)서버용 플레이어 1,2 제작 | (클라이언트)  서버와 통신할  Server  Connection 객체 생성 | (서버) 두 클라이언트의 데이터를 수신 하는 쓰레드 제작, 플레이어 선택 정보 받기 | (서버) 수신하는 쓰레드를 통해서 레벨 변경 | (서버)클라이언트로 데이터를 송신하는 쓰레드를 이용해서 클라이언트 레벨 변경 | 한 주간 미흡했던 부분 구현 |
| **11** | **12** | **13** | **14** | **15** | **16** | **17** |
| (서버)수신하는 쓰레드를 통해서 키 입력 정보 받고 플레이어 움직이기(임계영역) | (서버)송신 쓰레드를 통해 플레이어 이동 정보를 클라이언트로 넘겨주는 기능 구현 | (서버) 클라이언트에서 받은 플레이어 구조체를 통해 총알 발사 확인 | (서버)이벤트에 맞게 클라이언트로 데이터를 전송할 수 있는 구조 구현 | (서버)이벤트에 맞게 클라이언트로 데이터를 전송할 수 있는 구조 구현 | (서버)이벤트에 맞게 클라이언트로 데이터를 전송할 수 있는 구조 완성 | 한 주간 미흡했던 부분 구현 |
| **18** | **19** | **20** | **21** | **22** | **23** | **24** |
| (서버) 오브젝트 리스트 임계영역 설정 후 클라이언트로 몬스터 생성 관련 데이터 전송 | (서버)클라이언트로 몬스터 생성 관련 데이터 전송 구현, 클라이언트랑 싱크 맞추기 | (서버) 클라이언트로 몬스터 총알 발사 여부 보내서 연동하기 | (서버) 구현된 충돌처리로 몬스터들의 삭제되면 관련 데이터 전송, 클라이언트에서 확인 | (서버)충돌처리로 플레이어 체력이 감소되면 데이터 전송, 클라이언트에서 확인 | (서버)  플레이어 이동, 충돌처리, 몬스터와의 총알 충돌처리 미흡했던 부분 구현 | 한 주간 부족했던 부분 구현 |
| **25** | **26** | **27** | **28** | **29** | **30** | **1** |
| (서버) 보스 오브젝트 관련 데이터 클라이언트로 보내기 | (서버) 보스 오브젝트 관련 데이터 클라이언트로 보내기 | (서버) 클라이언트로 종료 데이터 보내기, 스코어 보내기 | (클라이언트) 서버로부터 받아온 종료 여부로 게임 종료화면 구현(리소스 편집) | (클라이언트) 서버로부터 받아온 종료 여부로 게임 종료화면 구현 | (서버,클라이언트) 부족했던 부분 보강 | 한 주간 부족했던 부분 구현 |
| **12월** | | | | | | |
| **2** | **3** | **4** | **5** |  |  |  |
| 버그 수정, 미흡했던 점 보충,  발표 준비 | | | |  |  |  |