|  |
| --- |
| **네트워크 게임 프로그래밍 추진 계획서** |



**2014182018 박찬희**

**2018182003 고태석**

**2018182022 윤세원**

**목차**

**1.애플리케이션 기획**...............................................................................................................................................................3

**1.1 기존 게임 개발 사항**

**1.2 개발 플랫폼**

**1.3 게임 설**

**1.4 추가 기능**

**2.High – Level 디자인**...........................................................................................................................................................6

**2.1 플로우 차트**

**2.2 서버 구현 내용**

**2.3 클라이언트 구현 내용**

**3.Low-level 디자인**............................................................................................................................................................... 9

**3.1 데이터 전송을 위한 패킷**

**3.2 서버**

**3.3 클라이언트**

**4**.**역할 분담**................................................................................................................................................................................17

**5**.**개발일정**.................................................................................................................................................................................17

**5.1 박찬희**

**5.2 고태석**

**5.3 윤세원**

**1.애플리케이션 기획**

**1.1 기존 게임 개발 사항**

2019년도 윈도우 게임 프로그래밍 텀프로젝트를 위해 만들었던 2D 횡스크롤 게임

제작자: 고태석

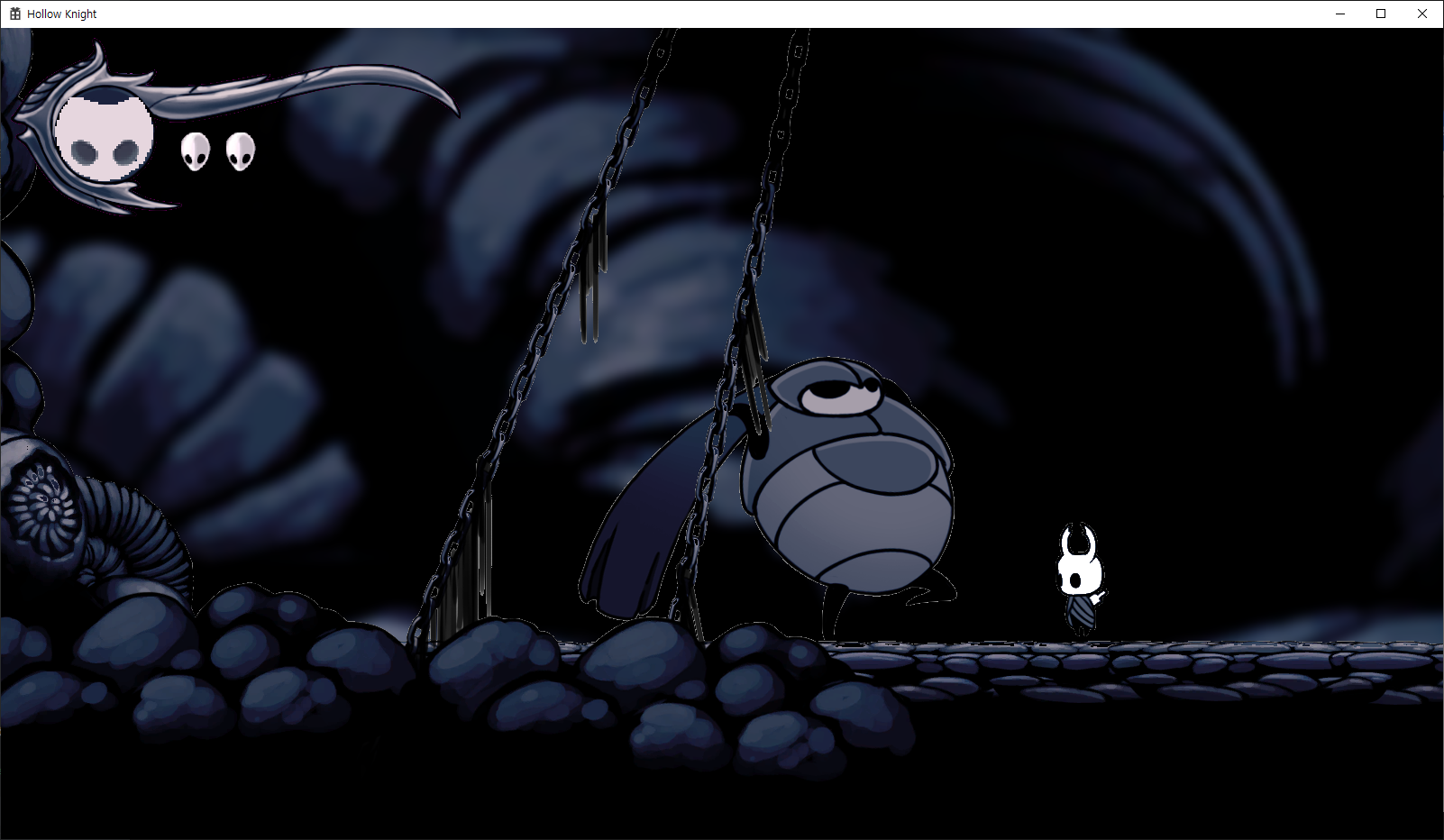
**1.2 개발 환경**

|  |  |  |
| --- | --- | --- |
|  | 클라이언트 | 서버 |
| IDE | MicroSoft Visual studio 2022 | |
| OS | Windows | |
| Language | C++, C | |
| VCS | Github | |
| Feature | Window API | TCP |

**1.3 게임 설명**

Team Cherry에서 제작하여 2017년 2월 25일 발매한 2D 횡스크롤 게임, Hollow Knight의 모작이다.

메인 맵이 있고 그 맵의 왼쪽 -> 오른쪽 방향으로 진행해 몬스터들을 제거하며 나아가 보스 몬스터를 잡으면 승리하는 게임이다.



**[그림1]** 기존 게임 캡처 장면

**조작방법**

- 이동: 방향키

- 점프: c

- 공격: x

**1.4 추가 기능**

**게임 클라이언트**

기존에 클라이언트에서 게임 내 객체들의 위치 이동, 충돌검사 부분을 서버 쪽으로 이전시켜 클라이언트 내에서는 SceneID에 따른 Scene의 변환과, 객체들의 렌더링, 키입력 부분만 수행하도록 변경한다.

**네트워크 기능**

1. 현재 1인 게임에서 3인까지 같이 플레이 가능한 멀티 플레이 게임으로 변경한다.

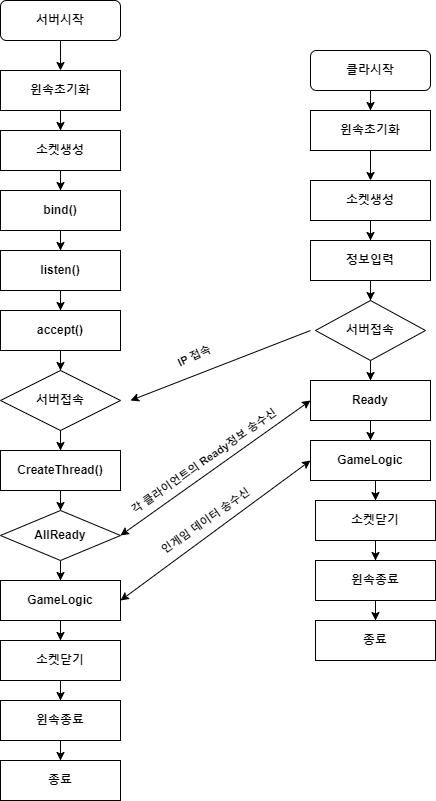
2. 게임 시작 시 서버 IP주소와 플레이어가 사용할 닉네임 정보를 입력하고 플레이어들이 대기 중인 화면으로 넘어간다. 플레이어들은 대기 화면에서 준비 완료 상태로 변경할 수 있다. 모든 플레이어가 준비 완료 상태가 되면 첫 번째로 들어온 플레이어가 START 버튼을 눌러 게임을 시작할 수 있게 한다.

3. 플레이어 간 구분은 플레이어의 캐릭터 위에 게임 시작 시 작성한 닉네임 정보를 사용하여 할 수 있도록 한다.

4. 게임 내 객체들의 로직(이동, 충돌체크)을 서버에서 계산하여 클라이언트에 전송하도록 변경한다.

**2. High – Level Design**

**2.1 플로우 차트**



**2.2 서버 구현 내용**

**서버 실행**

▶ 서버 실행시 winsock 초기화와 대기 소켓을 생성하고 bind 함수와 listen 함수를 통해 지역 IP주소와 지역 포트번호를 결정한다. 이후 클라이언트의 접속을 기다리는 대기 소켓의 상태로 만든다.

▶ 클라이언트가 connect 함수를 통해 서버에 접근 시, accept된 클라이언트를 위해 소켓을 생성하고, 그 후 로비화면으로 전환하는 메세지를 클라이언트에 송신한다.

**플레이어 별 스레드를 이용한 데이터 통신**

▶ 접속된 플레이어 별로 스레드가 할당되어 통신 소켓이 생성된다. 스레드 내부에서 닉네임과 IP주소를 받는다.

▶ 스레드 내부에서 클라이언트가 보내온 플레이어의 정보, 몬스터의 정보, 키입력 정보 등을 수신 받아 게임 오브젝트들의 갱신을 수행하고, 충돌체크 함수를 통해 벽과 플레이어, 몬스터와 플레이어, 몬스터와 플레이어 공격 이펙트 간의 충돌 검사를 수행한다.

▶ 게임 종료 조건(플레이어 사망 또는 보스몬스터 처치)에 맞는지 체크후, 종료 조건이 만족했을 시 SceneID를 변경하도록 클라이언트에게 전송한다.

▶ 게임 종료 조건이 만족되지 않았을 경우 클라이언트들에 모든 플레이어의 정보와 몬스터의 정보를 전송한다.

▶ 게임 종료 메세지를 받게되면, 클라이언트에 해당하는 스레드의 소켓을 종료하고, 스레드를 제거한다.

**2.3 클라이언트 구현 내용**

**타이틀 화면**

▶ 콘솔 창에 접속할 서버의 IP주소와 닉네임을 입력한다.

▶ winsock을 초기화하고, connect함수로 서버에 연결을 요청한다.

**로비 화면**

▶ 서버로 닉네임과 IP주소를 전송한다.

▶ 서버에 접속해 있는 플레이어의 닉네임 정보가 화면에 출력된다.

▶ 플레이어들의 준비 상태를 지속적으로 전송한다.

▶ 플레이어들이 모두 START버튼을 누르면 게임이 시작된다.

**게임 플레이 화면**

▶ 게임이 시작되면 각 플레이어의 클라이언트는 서버로부터 플레이어, 몬스터들의 데이터를 실시간으로 전송받는다.

▶ 각 클라이언트의 키보드 입력 정보를 실시간으로 서버에 전송한다.

▶ 서버로부터 받은 플레이어 데이터와 몬스터 데이터를 반영해 화면에 출력한다.

▶ 보스 몬스터 처치 시 로비 화면으로 돌아간다.

▶ 타이틀 화면에서 ‘Exit’버튼을 누르면 서버와의 접속이 종료된다.

**3. Low – Level Design**

서버 구현 방식: TCP

몬스터와 플레이어의 움직임, 충돌체크가 2D횡스크롤 게임에서 가장 중요한 요소라고 생각하여 모든 데이터를 정확하게 주고 받을 수 있는 TCP를 사용한다.

**3.1 송수신 데이터**

**클라이언트 → 서버**

자신의 닉네임 정보

자신의 레디 정보

키입력 정보

**서버 → 클라이언트**

다른 플레이어 레디 정보

다른 플레이어 닉네임 정보

SceneId //Scene 을 넘기기 위해 사용

플레이어 좌표 값

플레이어 상태 정보 // 스프라이트 이미지 변경을 위해 넘겨준다

플레이어 방향 정보 // 바라보는 위치에 따라

플레이어 생존 정보

플레이어 체력 정보

~~플레이어 공격 이펙트 좌표 값~~

몬스터 타입 정보

몬스터 좌표 값

몬스터 생존 정보

몬스터 상태 정보

몬스터 방향 정보

|  |
| --- |
| **열거형 종류** |
| enum **PLAYERSTATE**  {  STATE\_ATT, STATE\_IDLE, STATE\_WALK, STATE\_HIT,  STATE\_JUMP, STATE\_FALL, STATE\_LAND, STATE\_DEAD,  STATE\_ATTD, STATE\_ATTU  };  enum **DIR**  {  LEFT, RIGHT  };  enum **SCENEID**  {  SCENE\_LOGO,SCENE\_ LOBBY, SCENE\_MENU, SCENE\_STAGE1  };  enum **MONSTERTYPE**  {  BITTLE,BUG,FLY,HUSH,HUSHKNIGHT,BOSS  };  enum **MONSTERSTATE**  {  STATE\_ATT, STATE\_IDLE, STATE\_WALK  };  enum **KEYTYPE**  {  KEY\_C,KEY\_X,KEY\_LEFT, KEY\_RIGHT, KEY\_DOWN, KEY\_UP,  }; |

**3.2 데이터 전송을 위한 패킷**

클라이언트, 서버간의 데이터를 송수신 할 때 다음과 같은 변수를 구조체에 담아서 보낸다.

|  |
| --- |
| 위치 정보 구조체 |
| struct **Info** // 위치 정보 및 출력될 사이즈  {  float fX;  float fY;  float fCX; //x크기  float fCY; //y크기  }; |
| 플레이어 정보 구조체 |
| struct **PlayerData**  {  Info info;  PLAYERSTATE playerState;  string name; // 캐릭터 위에 플레이 닉네임 표시용  int playerHp;  bool playerAlive;  ~~RECT playerAttEff; // 플레이어 공격 범위~~  //공격범위는 어차피 플레이어 위치에서 일정량 떨어진 위치에서 출력되니 보내지 않아도 된다고 판단함  DIR playerDir; //플레이어가 바라보는 방향  }; |
| 몬스터 정보 구조체 |
| struct **MonsterData**  {  MONSTERTYPE monsterType;  Info info;  MONSTERSTATE monsterState;  bool monsterAlive;  DIR monsterDir;    RECT monsterAttEff; // 보스몬스터 공격 이펙트  }; |
| 키 정보 구조체 |
| struct **KeyInfo**  {  bool keyUp; //키up시 true ,down시 False  KEYTYPE keyType;  }; |
| 클라이언트 정보 구조체 |
| struct **ClientInfo**  {  SCENEID SceneId;  char PlayerIp  char name;  bool isReady;  ~~int clientNum; // 몇 번째 플레이어인지 구분하기 위함~~    }; |
| ~~프레임 정보 구조체~~ |
| ~~struct~~ **~~FRAME~~**  ~~{~~  ~~int iFrameStart; // 스프라이트 x 시작 지점.~~  ~~int iFrameEnd; // 스프라이트 y축갯수~~  ~~int iFrameScene; // 스프라이트 갯수~~  ~~DWORD dwFrameSpeed;~~  ~~DWORD dwFrameTime; // 재생에 필요한 시간.~~  }; |

**3.3 서버**

접속 클라이언트 관리용

int clientCount = 0;

vector<PlayerData> PlayersData; //플레이어들의 정보를 저장하기 위한 컨테이너

Scene \*pScene; //화면 정보를 담을 Scene포인터

**송신 함수**

|  |  |
| --- | --- |
| **이름** | **설명** |
| void **SendClientInfo**(ClientInfo clientInfo) | 로비화면에서 사용될 함수로 클라이언트의 닉네임,IP주소, 준비 상태 여부를 송신해준다. |
| void **SendPlayerData**(vector<PlayerData> PlayersData) | 플레이어들의 위치, 스프라이트 이미지 사용을 위한 상태, 닉네임, Hp, 바라보고 있는 방향, 생존 정보를 송신하는 함수이다. |
| void **SendMonsterData**(vector<MonsterData> MonsterData) | 몬스터들의 타입, 위치, 스프라이트 이미지 사용을 위한 상태와, 생존 여부, 바라보고 있는 방향을 보내준다. |
| void **SendSceneId**(SCENEID sceneID) | 화면 정보를 클라이언트에게 송신하여 화면 전환을 위한 화면 아이디를 넘겨주는 함수이다. |

**수신 함수**

|  |  |
| --- | --- |
| **이름** | **설명** |
| void **RecvClientInfo**(ClientInfo clientInfo) | 로비화면에서 클라이언트의 정보를 수신하는 함수이다. |
| void **RevcKeyInfo**(KeyInfo keyInfo) | 플레이어가 입력한 키 정보를 수신하기 위한 함수이다. 게임 플레이에서 이동 방향키, 공격 키, 점프 키, 키가 눌렸는지의 여부를 처리한다. |

**사용 함수**

|  |  |
| --- | --- |
| **이름** | **설명** |
| void **PlayerToMapCollide**(RECT Player,vector<RECT>wall) | 맵과 플레이어간 충돌 체크를 위한 함수 |
| void **MonsterToPlayer Collide** (RECT Monster,RECT Player ) | 몬스터와 플레이어의 충돌체크를 위한 함수 |
| void **MonsterToEffect Collide** (RECT Monster,RECT Effect ) | 몬스터와 플레이어 공격 모션 충돌 체크를 위한 함수 |
| void **PlayerUpDate**(KeyInfo KeyInfo) | 전달받은 키 입력에 따라 State 정보와 위치 정보를 갱신하는 함수 |
| void **MonsterUpdate**() | 몬스터의 위치 정보와 상태 정보를 갱신하는 함수 |
| bool **MonsterDetectPlayer**() | MonsterUpdate()함수 내에서 사용될 플레이어 탐지 함수, 몬스터가 설정된 범위내에 플레이어가 있을 시 true,없을시 false를 리턴한다. |
| void **GameEndCheck**() | 게임 종료 조건에 따른 게임 종료 여부를 검사한다. |

**서버에서 사용 할 클래스**

|  |  |
| --- | --- |
| **이름** | **설명** |
| class **Scene** | 프로그램의 화면 구성 클래스  현재의 Scene에 따라 화면을 전환한다. |

**데이터 동기화 용 스레드 함수**

|  |  |
| --- | --- |
| **이름** | **설명** |
| DWORD WINAPI **ClientThread**(SOCKET client\_sock) | 각 클라이언트와 통신을 담당 할 스레드 함수  해당 함수 내에서 수신함수들의 값을 수신하고, 송신함수들의 정보를 송신한다. |

**3.4 클라이언트**

**송신 함수**

|  |  |
| --- | --- |
| **이름** | **설명** |
| void **SendClientInfo**(ClientInfo clientInfo) | 로비화면에서 클라이언트의 정보를 송신하는 함수이다. |
| void **SendKeyInfo**(KeyInfo keyInfo) | 플레이어가 입력한 키 정보를 송신하기 위한 함수이다.  게임 플레이에서 이동 방향키, 공격 키, 점프 키, 키가 눌렸는 지의 여부를 처리한다. |

**수신 함수**

|  |  |
| --- | --- |
| **이름** | **설명** |
| void **RecvClientInfo**(ClientInfo clientInfo) | 로비화면에서 사용될 함수로 클라이언트의 닉네임, IP주소, 준비 상태 여부를 수신해준다. |
| void **Recv PlayerData**(vector<PlayerData> PlayersData) | 플레이어들의 위치, 스프라이트 이미지 사용을 위한 상태, 닉네임, Hp, 바라보고 있는 방향, 생존 정보를 수신하는 함수이다. |
| void **Recv MonsterData**(vector<MonsterData> MonsterData) | 몬스터들의 타입, 위치, 스프라이트 이미지 사용을 위한 상태와 생존 여부, 바라보고 있는 방향을 수신한다. |
| void **Recv SceneId**(SCENEID sceneID) | 화면 정보를 서버 수신하여 화면 전환을 위한 화면 아이디를 변경하는 함수이다. |

**클라이언트에서 사용 할 클래스**

|  |  |
| --- | --- |
| **이름** | **설명** |
| class **Scene** | Scene들을 생성하기 위한 부모클래스  가상 함수로 Render()를 두어 화면에 출력 되도록 한다. |
| class **Logo**  class **Menu**  class **CStage1** | Scene에 상속 되어 있는 하위 클래스들 |
| Class **SceneMgr** | Scene 정보를 관리할 클래스, 해당 매니저는 싱글턴을 사용하여 인스턴싱 되어있다. |
| Class **KeyMgr** | Key 정보를 관리할 클래스, 해당 매니저는 싱글턴을 사용하여 인스턴싱 되어있다. |

**4. 역할 분담**

|  |  |  |
| --- | --- | --- |
| **박찬희** | **고태석** | **윤세원** |
| 서버 보조 | 클라이언트 메인 | 서버 메인 |
| 레디 송수신 구현 | 클라이언트측 송수신 구현 | 충돌체크 함수 구현 |
| 충돌체크 함수 보조 | 공유 자원 설계 | 스레드 함수 설계 |
| 몬스터 로직 구현 | 플레이어 로직 구현 | 몬스터 로직 구현 |
| 동기화 문제 해결 | | |

**5. 개발 일정**

**5.1 박찬희**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **일** | **월** | **화** | **수** | **목** | **금** | **토** |
|  |  | **11/1** | **11/2** | **11/3** | **11/4** | **11/5** |
|  |  |  |  |  | **예비군 훈련** |  |
| **11/6** | **11/7** | **11/8** | **11/9** | **11/10** | **11/11** | **11/12** |
|  |  |  |  |  |  |  |
| **11/13** | **11/14** | **11/15** | **11/16** | **11/17** | **11/18** | **11/19** |
|  |  | **※적어도 로비 화면구현까지는 완료** |  |  |  |  |
| **11/20** | **11/21** | **11/22** | **11/23** | **11/24** | **11/25** | **11/26** |
|  |  |  |  |  | **※적어도 플레이어 이동, 공격, 매맵과 플레이어의 충돌 처리 완료** |  |
| **11/27** | **11/28** | **11/29** | **11/30** | **12/1** | **12/2** | **12/3** |
|  |  |  |  |  |  |  |
| **12/4** | **12/5** | **12/6** | **12/7** |  | | |
| **최종 테스트 및 보완, 보고서 작성** | | | |

**5.2 고태석**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **일** | **월** | **화** | **수** | **목** | **금** | **토** |
|  |  | **11/1** | **11/2** | **11/3** | **11/4** | **11/5** |
|  |  |  |  |  | **예비군 훈련** | **종합설계기획** |
| **11/6** | **11/7** | **11/8** | **11/9** | **11/10** | **11/11** | **11/12** |
| **종합설계기획** | **시험기간** | | **클라이언트 프레임워크 수정** | **Scene 클래스 멤버 함수 로직 수정** | **LobbyScene()구현** | **테스트 후 보완** |
| **11/13** | **11/14** | **11/15** | **11/16** | **11/17** | **11/18** | **11/19** |
|  | **SendClientInfo, Recv SceneId 구현** | **※적어도 로비 화면구현까지는 완료** |  | **SendKeyInfo()** | **SendPlayerData 구현** | **Recv PlayerData구현** |
| **11/20** | **11/21** | **11/22** | **11/23** | **11/24** | **11/25** | **11/26** |
| **테스트 후 보완** |  | **Player 관련 로직 버그 수정** | **Player 관련 로직 버그 수정** |  | **현재까지 미흡 부분 보완**  **※적어도 플레이어 이동, 공격, 매맵과 플레이어의 충돌 처리 완료** | **3차테스트** |
| **11/27** | **11/28** | **11/29** | **11/30** | **12/1** | **12/2** | **12/3** |
| **테스트 후 보완** | **ClientThread()**  **동기화 관련 문제 해결** | | | | |  |
| **12/4** | **12/5** | **12/6** | **12/7** |  | | |
| **최종 테스트 및 보완, 보고서 작성** | | | |

**5.3 윤세원**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **일** | **월** | **화** | **수** | **목** | **금** | **토** |
|  |  | **11/1** | **11/2** | **11/3** | **11/4** | **11/5** |
|  |  |  |  |  | **예비군 훈련** | **종합설계기획** |
| **11/6** | **11/7** | **11/8** | **11/9** | **11/10** | **11/11** | **11/12** |
| **종합설계기획** | **시험 기간** | | **서버 프레임워크 설계** | **서버 프레임워크 제작** | **RecvClientInfo, SendClientInfo 구현** | **테스트 후보완** |
| **11/13** | **11/14** | **11/15** | **11/16** | **11/17** | **11/18** | **11/19** |
| **다렉 과제** | **Send SceneId,**  **MonsterUpdate**  **구현** | **※적어도 로비 화면구현까지는 완료** | **다렉 과제** | **다렉 과제 보고서 쓰기** | **RecvKeyInfo() 구현** | **SendMonsterData 구현, Recv MonsterData구현** |
| **11/20** | **11/21** | **11/22** | **11/23** | **11/24** | **11/25** | **11/26** |
| **몬스터 로직 테스트 후 보완** |  | **맵과 플레이어의 충돌 체크** | **플레이어와 몬스터의 충돌체크 구현,**  **플레이어 공격과 몬스터 충돌체크 구현** |  | **MonsterDetectPlayer 구현**  **※적어도 플레이어 이동, 공격, 매맵과 플레이어의 충돌 처리 완료** | **3차 테스트** |
| **11/27** | **11/28** | **11/29** | **11/30** | **12/1** | **12/2** | **12/3** |
| **테스트 후 보완** | **공유자원 문제 해결** | | | | |  |
| **12/4** | **12/5** | **12/6** | **12/7** |  | | |
| **최종 테스트 및 보완, 보고서 작성** | | | |