REPORT

텀프로젝트 추진계획서 3.5

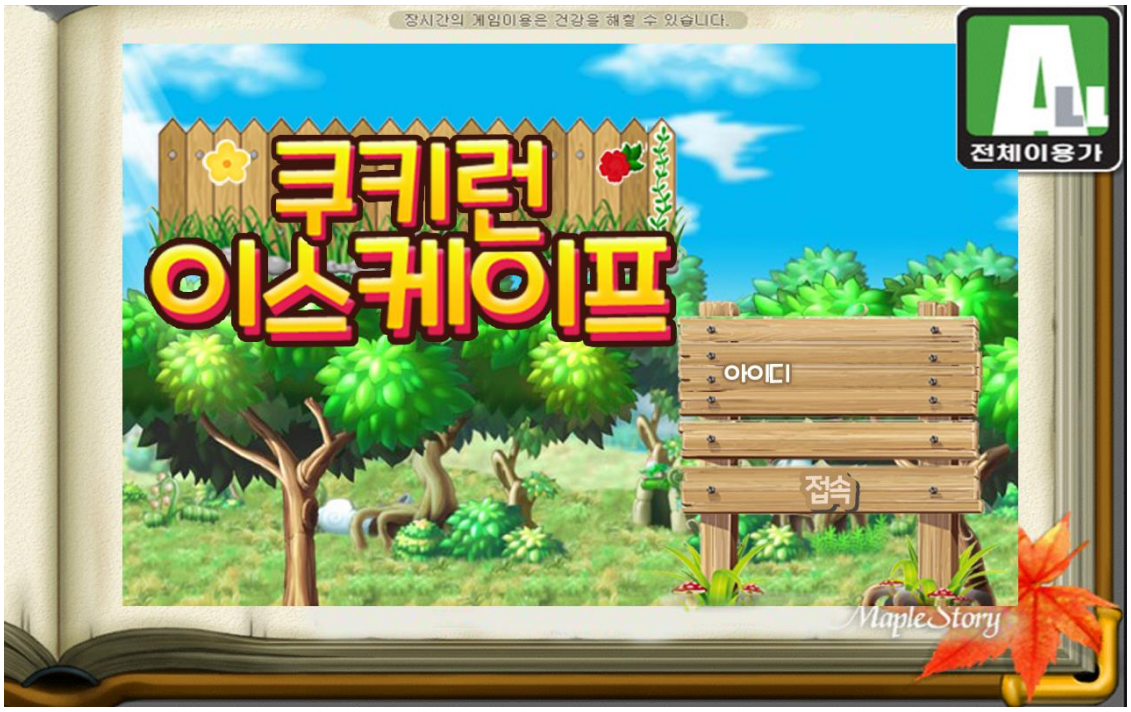
한국공학대학교
TECH UNIVERSITY OF KOREA

학과	게임공학과
과목명	네트워크 게임 프로그래밍(01)
학생	2019180025 우정연 2020182032 이세민 2020180034 정가온
제출일	2022.11.12

목차

Ⅰ. 애플리케이션 기획

1. 게임 스크린 샷



2. 프로그램 개요

이름	쿠키 이스케이프
교과목	윈도우 게임 프로그래밍
작업자	2020182032 이세민
장르	플랫폼 액션 게임
플레이어 수	3명
플레이 시간	2분

3. 소개

기존 프로젝트는 WinAPI를 사용해 개발되었다. 몬스터를 피해 열쇠를 획득하여 문 밖으로 탈출하는 게임이다. 네트워크 통신 기능을 추가하여 코인 획득 및 스코어 계산, 몬스터 처치, 승패 결정 등의 기능을 넣는 것이 애플리케이션 기획의 목표이다.

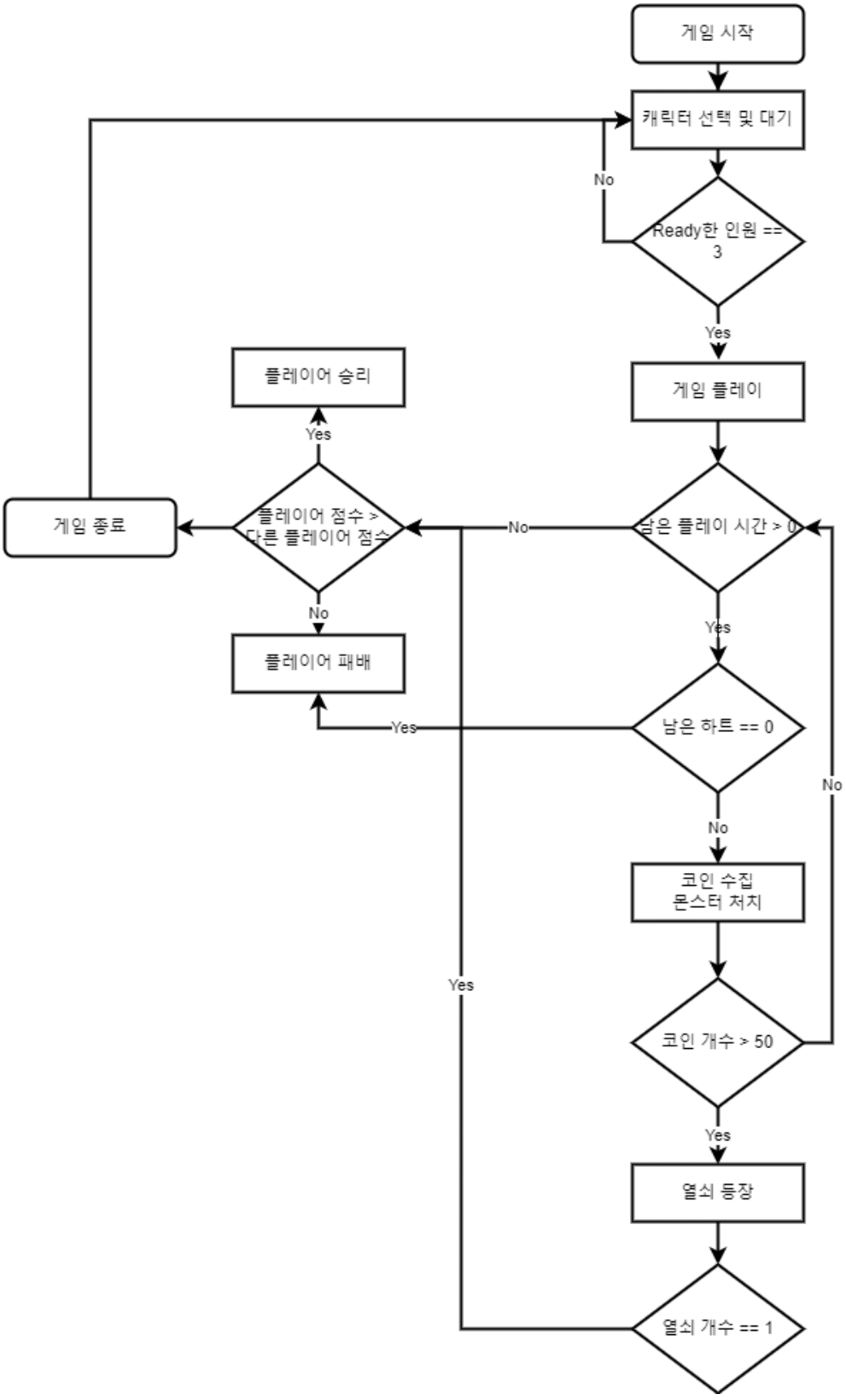
4. 특징 및 내용

- 수정 전 게임의 특징
  - ‘쿠키런’과 ‘마리오’ 게임의 특징을 조합한 게임이다. 쿠키런 캐릭터를 주인공으로 하여 탈출하는 내용이다.
  - 몬스터의 머리를 밟아 몬스터를 죽일 수 있는 마리오의 특징을 가져왔다.
  - 세 가지 캐릭터 중 하나를 선택할 수 있다.
  - 체력은 하트로 표시되며, 총 세 개의 하트를 가지고 있다. 몬스터와 충돌하면 하트가 감소하며, 0개가 되면 게임 오버된다.
- 수정 후 게임의 특징
  - 대기화면에서 캐릭터 3종 중 하나를 선택할 수 있다.
  - 캐릭터마다 가지고 있는 능력치가 다르다.
  - 플레이어들의 총 코인 획득 수가 50개를 초과하면 열쇠를 획득할 수 있다. 그 열쇠를 사용해 포탈을 열어 맵을 탈출할 수 있다.
  - 쿠키의 체력은 하트로 표시되고, 하트는 3개가 기본이지만 캐릭터의 능력치별로

상이할수 있다. 몬스터와 충돌 시 하트가 한 개씩 깎인다.

- ◆ 맵의 코인은 플레이어 모두에게 공통으로 보인다. 코인을 많이 먹을수록 점수가 높아지고, 점수가 제일 높은 플레이어가 우승을 하게 된다.
- ◆ 열쇠를 먹으면 5점이 올라가고, 코인을 먹으면 1점이 올라가고 몬스터를 죽이면 2점이 올라간다.
- ◆ TCP를 사용하여 멀티플레이를 지원한다.
- ◆ 이벤트를 사용해 스레드 동기화를 지원한다.

5. 게임 진행 흐름도


게임 시작
캐릭터 선택 및 대기
No
Ready한 인원 == 3
Yes
게임 플레이
남은 플레이 시간 > 0
No
Yes
남은 하트 == 0
Yes
No
플레이어 패배
No
Yes
플레이어 점수 > 다른 플레이어 점수
Yes
플레이어 승리
게임 종료
Yes
코인 수집
몬스터 처치
코인 개수 > 50
No
Yes
열쇠 등장
열쇠 개수 == 1
Yes

1. 게임을 시작하고 아이디를 입력하면 캐릭터 선택 창으로 넘어간다.
2. 캐릭터를 선택하면 접속 인원이 3명이 될 때까지 대기한다.
3. 접속 인원이 3명이 되면 게임을 시작한다.
4. 주어진 시간 내에 각 플레이어별로 코인을 획득하거나, 몬스터를 처치하여 점수를 올린다.
5. 모든 플레이어의 획득 코인 총 합산이 50개 초과가 된다면 열쇠가 맵에 등장한다.
6. 열쇠를 획득한 플레이어에게 추가 점수가 지급되며 그 열쇠를 사용해 포탈을 열 수 있다.
7. 포탈에 들어가거나 주어진 시간이 초과되면 게임이 종료된다.
8. 게임이 종료된 후, 플레이어별 점수에 따라 승패를 결정해 시상식을 진행한다.
9. 초기화면으로 돌아간다.

6. **조작 방식**

- 키보드 좌우 화살표: 좌우 이동
- 스페이스바: 점프
- 1, 2 키: 게임 오버 시 다른 플레이어에게 카메라 전환

7. **점수 획득 방식**

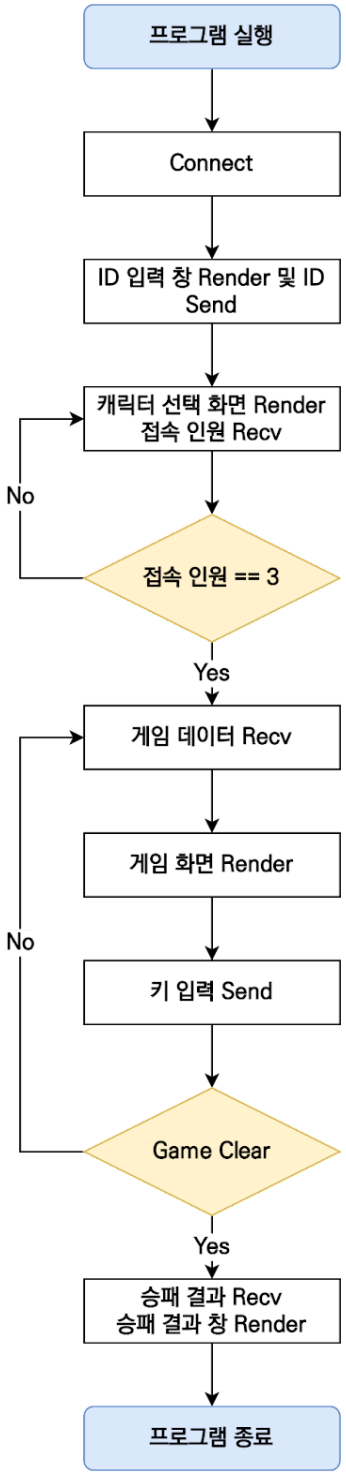
- 코인 획득: 맵에 배치된 코인을 획득하면 1점을 얻는다.
- 몬스터 처치: 맵에 배치된 몬스터를 밟아 처치하면 2점을 얻는다.
- 열쇠 획득: 열쇠를 최초로 발견해 획득하면 5점을 얻는다.

8. **종료 조건**

- 게임 클리어
  - 맵에 발생한 열쇠를 획득해 포탈로 탈출하면 게임을 종료한다.
- 게임 오버
  - 모든 플레이어의 체력이 0이 되면 게임을 종료한다.
  - 플레이 타임 2분이 초과하면 게임을 종료한다.

II. High-Level 디자인

1. 클라이언트



① 프로그램 실행 후 서버에 접속한다.

② ID를 입력하면 서버에 ID 정보를 전송한다.

③ 캐릭터 선택 화면에서 플레이어 캐릭터 선택 정보와 접속 인원을 받는다.

④ 접속 인원이 3명이 되면 게임을 시작한다.

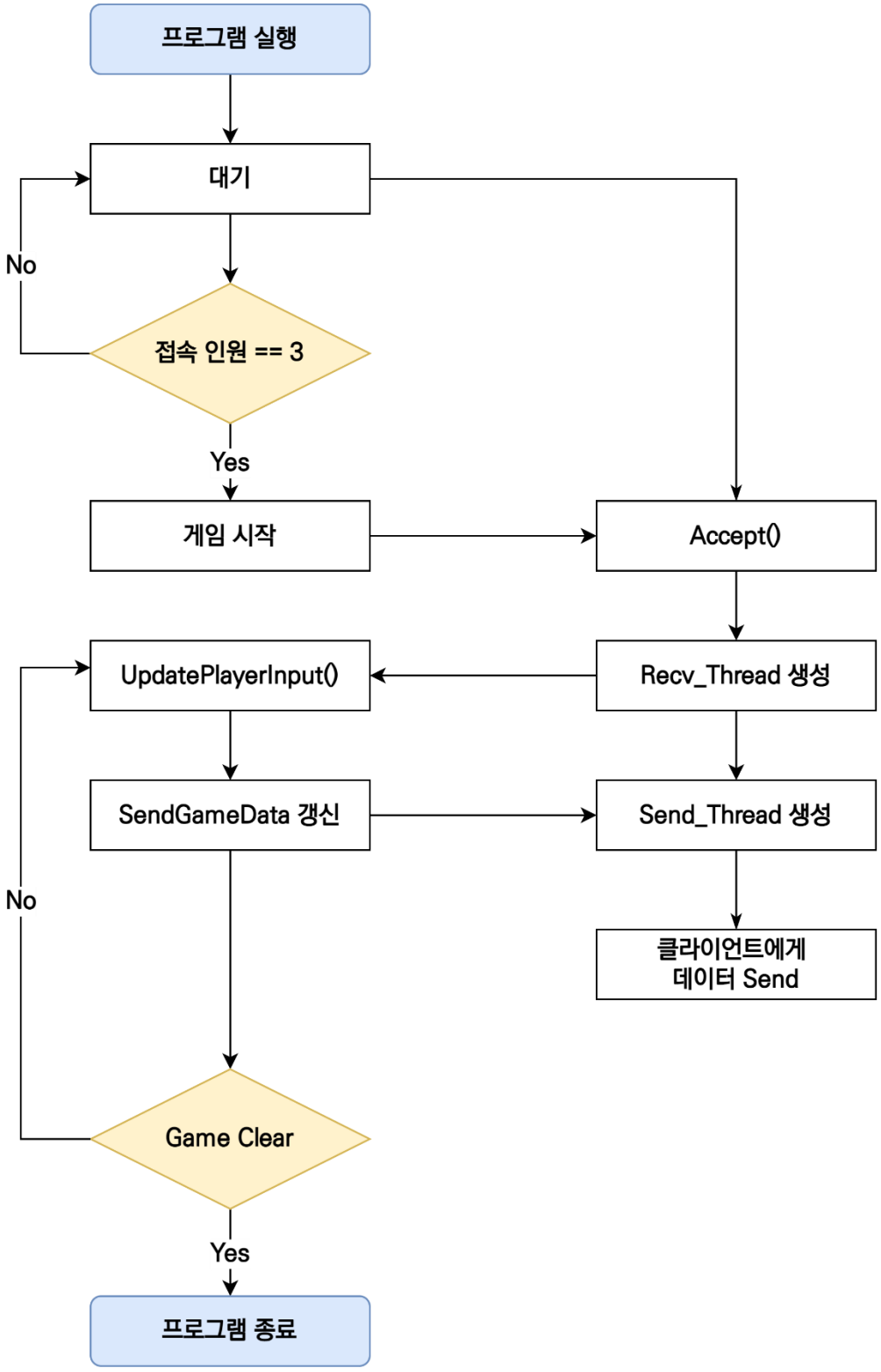
⑤ 게임 진행 중에는 서버로부터 게임 데이터를 받아 게임 화면을 렌더링한다.

⑥ 플레이어의 키 입력을 서버로 보낸다.

⑦ 게임 클리어 혹은 게임 오버 시 승패 결과를 받아 결과 창을 렌더링한다.

⑧ 프로그램을 종료한다.

2. 서버


프로그램 실행
대기
No
접속 인원 == 3
Yes
게임 시작
Accept()
UpdatePlayerInput()
Recv_Thread 생성
SendGameData 갱신
Send_Thread 생성
No
클라이언트에게
데이터 Send
Game Clear
Yes
프로그램 종료

~~① 프로그램 실행 후 접속 인원이 3명이 될 때까지 대기를 한다.~~

① 프로그램 실행 후 게임 시작 전에는 클라이언트로부터 ID 정보를 받아 접속 인원이 3명이 될 때까지 대기한다.

② 게임이 시작되면 Send_Thread와 Recv_Thread를 생성한다.

③ 초기 데이터를 보낸 후 event를 시작한다. 접근 순서가 중요해 이벤트를 사용해 동기화를 진행한다.

④ 멀티스레드를 사용해 Recv_Thread는 클라이언트의 입력을 받아 게임 데이터를 갱신하는 역할을 한다. 이벤트 신호 상태를 확인하여 신호 상태일 때 게임 데이터를 갱신한다. ~~클라이언트의 입력을 Recv_Thread로 받아 게임 데이터를 갱신한다.~~

⑤ 점수에 직접적인 영향을 주는 코인, 몬스터, 열쇠에 충돌이 있을 때 이벤트 신호를 비신호 상태로 바꾼다.

⑥ Send_Thread는 클라이언트에게 게임 데이터를 전송하는 역할을 한다. 1/30초마다 갱신된 게임 데이터를 보낸다.

~~⑦ 1/30초마다 Send_Thread로 게임 데이터를 클라이언트에게 보낸다.~~

~~⑧ 플레이어의 키보드 입력으로 게임 데이터를 갱신하고, Send_Thread에 게임 데이터를 담아 클라이언트에게 전송한다.~~

⑨ 승패 결과가 정해지면, 결과를 전송하고 프로그램 종료한다.

Ⅲ. Low-Level 디자인

1. 클라이언트

역할	변수, 함수

클라->서버 전송 구조체	struct SendPlayerData{
	USHORT uID;
	USHORT uCharNum;
	KeyInput Input;
	}

	SendPlayerData PlayerData;

서버->클라 전송 구조체	struct SendGameData{
	PlayerMgr players[];
	clock_t ServerTime;
	bool bWin;
	bool bIsPlaying;
	
	vector<Monster> monsters;
	vector<Coin> coins;
	}

	SendGameData ServerData;

키보드 입력 유무 판단 구조체	struct KeyInput{
	bool bRight;
	bool bLeft;
	bool bSpace;
	}

플레이어 정보를 전송 구조체에 업데이트	void UpdateSendData()
서버와 통신할 클라이언트의 소켓정보	SOCKET sock
데이터 통신시 사용 소켓 주소 구조체	SOCKETADDR_IN serveraddr

서버의 아이피를 설정하는 변수	`#define SERVERIP`
서버 포트번호 설정에 사용하는 변수	`#define SERVERPORT`
서버에 게임 시작 유무 판단 변수	`bool IsPlayingGame`
총 인원수를 설정하는 변수	`#define TOTALCLIENT`
사망시 선택한 플레이어의 시점으로 카메라 전환하는 함수	`void ChangeCamera()`
현재 카메라가 따라가는 플레이어 (플레이어 멤버변수)	`USHORT Player::uLookAt`
점수 관리하는 변수	`USHORT Player::uScore`

2. 서버

역할	변수, 함수
접속 인원을 저장하는 변수	USHORT uClientNum

모든 플레이어를 관리하는 구조체	struct PlayerMgr{
	DWORD portnum;
	Player player;
	}

	PlayerMgr Players[]
키 입력을 받는 스레드	DWORD WINAPI Recv_Thread(LPVOID arg)
게임 데이터를 보내는 스레드	DWORD WINAPI Send_Thread(LPVOID arg)

클라이언트에게 받는 구조체	struct RecvPlayerData {
	USHORT uID;
	USHORT uCharNum;
	KeyInput Input;
	}

서버 시간 기록함수	void RecordTime()
서버 시간 갱신	void UpdateTime()
이벤트 스레드 핸들 변수	HANDLE hEventHandle;
서버 생성 시간을 저장하는 변수	clock_t serverStartTime
현재 서버 시간을 저장하는 변수	clock_t serverCurTime
서버 시간의 변화량을 저장하는 변수	clock_t serverDeltaTime
코인과 플레이어 간 충돌 여부를 저장하는 변수	bool bCollisionCoin

몬스터와 플레이어 간 충돌 여부를 저장하는 변수	`bool bCollisionMonster`
열쇠와 플레이어 간 충돌 여부를 저장하는 변수	`bool bCollisionKey`
포탈과 플레이어 간 충돌 여부를 저장하는 변수	`bool bCollisionPotal`
플레이어 위치 정보 갱신 필요 여부를 저장하는 변수	`bool bIsPosUpdate`
게임 실행 여부를 나타내는 변수	`bool bIsPlaying;`
클라이언트와 통신할 서버의 소켓 정보	`SOCKET sock`
서버 소켓 구조체	`struct sockaddr_in serveraddr`
클라이언트 소켓 구조체	`struct sockaddr_in clientaddr`
서버 포트번호 설정에 사용하는 변수	`#define SERVERPORT`
윈속 초기화 시 사용하는 변수	`WSADATA wsa`
서버를 생성하여 플레이어의 초기 위치를 선정하고 서버 데이터를 초기화하는 함수	`void InitServer()`
접속 인원 조건이 모두 충족되었는지 확인하는 함수	`bool AllReady()`
플레이어 몬스터 간 충돌 체크 함수	`bool IsCollidedMonster(Player&, Monster)`
플레이어 코인 간 충돌 체크 함수	`bool IsCollidedCoin(Player&, Coin)`

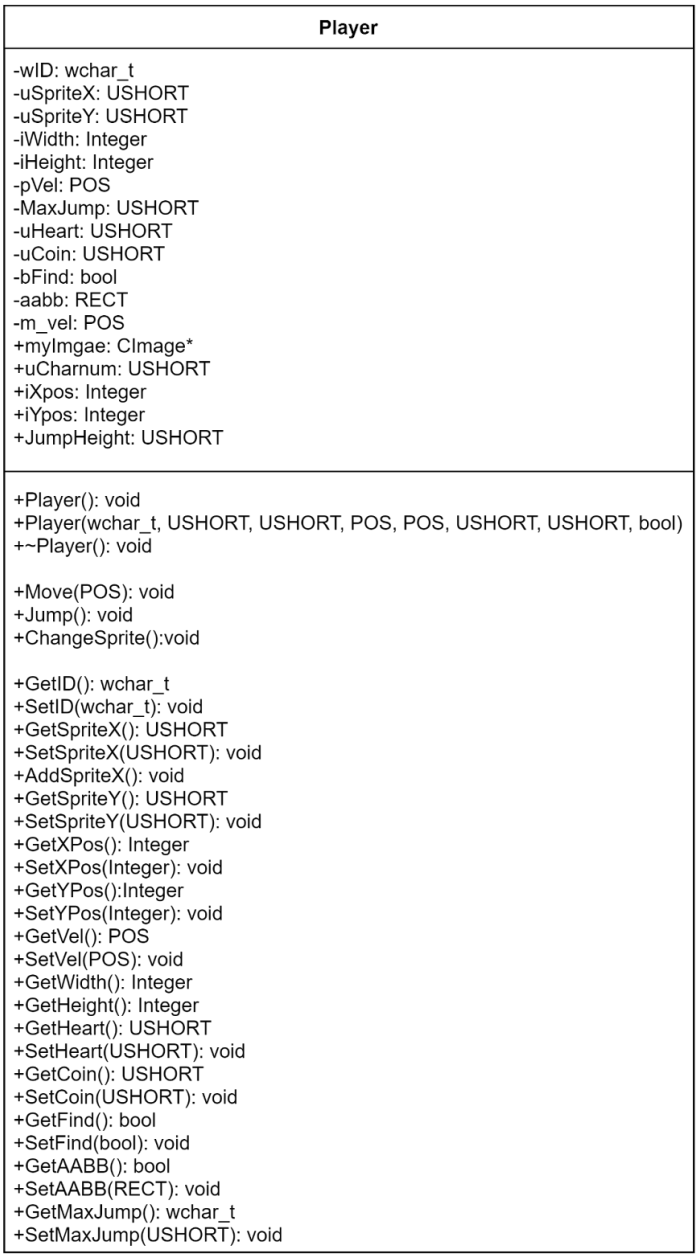
플레이어 열쇠 간 충돌 체크 함수	`bool IsCollidedKey(Player&, Key)`
플레이어 포탈 간 충돌 체크 함수	`bool IsCollidedPortal(Player&, Potal)`
플레이어 몬스터 간 충돌 처리 함수	`void CollideMonster()`
플레이어 코인 간 충돌 처리 함수	`void CollideCoin()`
플레이어 열쇠 간 충돌 처리 함수	`void CollideKey()`
플레이어 포탈 간 충돌 처리 함수	`void CollidePortal()`
플레이어의 위치정보 갱신 함수	`void UpdatePlayerLocation()`
플레이어 키보드 입력 처리 함수	`void UpdatePlayerInput()`
게임 종료 판별 함수	`void CheckGameOver()`

플레이어, 코인, 몬스터, 플랫폼 생성 및 초기화 함수	`void InitPlayer()`
	`void InitCoin()`
	`void InitMonster()`
	`void InitPlatform()`

클라이언트로부터 플레이어가 캐릭터를 선택했다는 상태 전달받는 함수	`bool IsReady(Player&)`
플레이어별 점수에 따른 승패 판별 함수	`bool IsPlayerWinner(USHORT)`
~~클라이언트와 데이터 통신을 위한 스레드 함수~~	~~`DWORD WINAPI ProcessClient(LPVOID arg)`~~
~~클라이언트와 데이터 통신을 위한 스레드 핸들 변수~~	~~`HANDLE hClientThread`~~

Ⅳ. 기존 프로그램 내부 클래스 구조와 함수

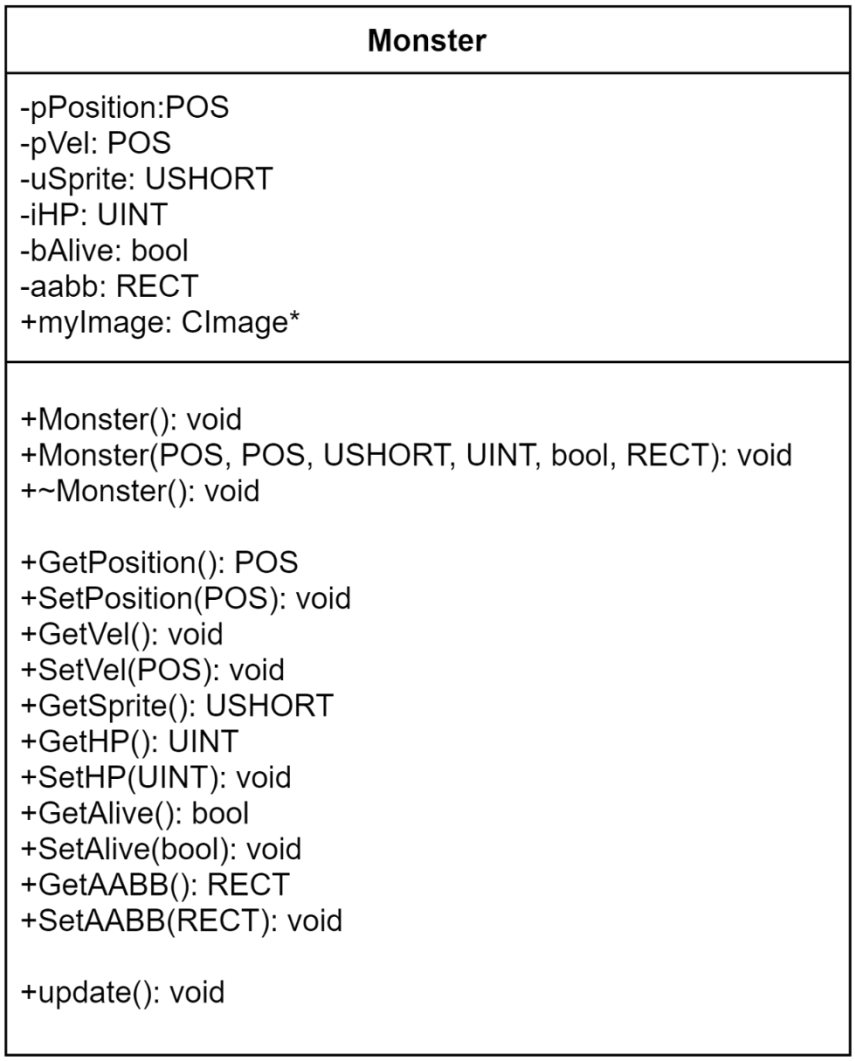
1. Player



함수	역할
void Player::Move(POS force)	키 입력에 따라 플레이어를 이동한다.
void Player::Jump()	키 입력에 따라 플레이어를 점프시킨다.

void Player::ChangeSprite()	캐릭터의 스프라이트를 전환한다.

2. Monster



3. Background

Background
-width: Integer -height: Integer +canvas_width: Integer +canvas_height: Integer +window_left: Integer +window_bottom: Integer +Imgae: CImage*
+Background(): void +~Background(): void +Update(): void +SetWindow(Integer, Integer): void +SetWidth(Integer): void +SetHeight(Integer): void

4. 기타

<<utility>> **POS**
+x: float +y: float
+POS(): void +POS(float, float): void +operator=(const POS): POS

class Player	플레이어의 위치, 스프라이트번호, 속력, 점프위치, 목숨, 코인갯수를 저장하는 구조체
class Monster	몬스터의 위치, 스프라이트 번호, 목숨을 저장하는 구조체
class Background	문의 위치와 개방유무와 배경크기정보를 저장하는 구조체

V. 팀원별 역할분담

우정연	서버	✓ 플레이어와 몬스터, 코인, 열쇠 사이의 충돌을 체크하고 그에 따른 오브젝트의 정보를 업데이트한다. (점수 증가, 오브젝트 삭제 등) ✓ 게임 시작 시 서버 시간을 기록하고, 클라이언트로 전송한다. ✓ 플레이 시간을 계산해 클라이언트로 전송한다. ✓ 서버 시간을 통해 게임 종료 여부를 판단하고, 클라이언트로 전송한다. ✓ 클라이언트에서 캐릭터 선택 여부 정보를 받아 캐릭터 중복 선택을 막고, 접속 인원 수를 판단해 게임 시작 여부를 클라이언트로 전송한다. ✓ 클라이언트에게 정보를 전송하는 Send_Thread 를 작성한다. ✓ 서버 동기화
	클라	✓ 몬스터 클래스를 구현한다. ✓ 충돌처리를 위한 구조체와 함수를 구현한다. ✓ 서버로부터 플레이 시간 정보를 받아 화면에 출력한다. ✓ 게임 플레이 전 캐릭터 선택 기능을 구현한다.
이세민	서버	✓ 클라이언트에게 ID 정보를 받아 기록한다. ✓ 클라이언트의 키보드 입력을 받아 플레이어의 이동을 구현한다. ✓ 플레이 시간 업데이트를 구현하고, 플레이 시간을 클라이언트로 전송한다. ✓ 플레이 시간 및 승패, 게임 오버 여부로 게임 종료를 판단하고 승패 결과를 클라이언트로 전송한다. ✓ 1/30 초마다 클라이언트에게 게임 데이터 전송을 구현한다.

		✓ 서버 동기화
	클라	✓ 플레이어 클래스를 구현한다. ✓ 플레이어별 ID 입력 화면을 구현하고 ID 정보를 서버에 전송한다. ✓ 클라이언트의 키보드 입력 정보를 서버에게 전송한다. ✓ 게임 종료 판단을 서버에게 받고, 스코어에 따른 승패 여부를 화면에 출력한다. ✓ 리소스 제작 • ID 입력 화면 리소스 • 승패 결과 화면 리소스
정가온	서버	✓ 발판, 코인, 몬스터, 캐릭터 정보를 초기화하고 클라이언트로 전송한다. ✓ 클라이언트에게 받은 정보 수신한다. ✓ 서버 동기화 ✓ 클라이언트에게 받은 정보 수신하는 Recv_Thread 를 구현한다. 서버에서 갱신된 데이터들을 클라이언트에게 전송한다. • 몬스터 위치와 생존 유무 • 코인 획득 유무 • 코인 수에 따른 열쇠 렌더링 유무 • 키입력에 따른 플레이어 위치
	클라	✓ 코인 클래스를 구현한다. ✓ 플레이어의 위치에 따라 카메라 스크롤링을 구현한다. ✓ 플레이어 사망시 생존한 플레이어 시점으로 카메라 전환 기능 추가한다.

		✓ 서버에서 받은 초기정보를 기반으로 렌더링을 진행한다. • 코인 • 발판 • 플레이어 ✓ 서버에서 받은 코인정보를 기반으로 렌더링 구현한다.

Ⅵ. 개발 환경

- Visual Studio 2022
- Windows API
- GitHub

우정연	11th Gen Intel(R) Core(TM) i7-11800H / 32.0GB/ Windows 11
이세민	AMD Ryzen 7 5800H with Radeon Graphics / 16.0GB / Windows 11
정가온	AMD Ryzen 7 5700U with Radeon Graphics / 16.0GB / Windows 11

Ⅶ. 개발 일정

1. 2019180025 우정연

11월						
일	월	화	수	목	금	토
		1	2	3	4	5
				기획서 1차 제출		몬스터 클래스 수정
6	7	8	9	10	11	12
캐릭터 선택창 리소스 제작		기획서 2차 제출		기획서 3차 제출	충돌처리를 위한 구조체 생성	기획서 3.5차 제출
13	14	15	16	17	18	19
			15일까지 각 팀원들 구현한 내용들 검토 및 Merge	IsReady()		AllReady()
20	21	22	23	24	25	26
RecordTime()	SendThread	UpdateTime() 서버 동기화	22일까지 각 팀원들 구현한 내용들 검토 및		IsCollidedMonster(), CollideMonster()	IsCollidedCoin(), CollideCoin()

			Merge			
27	28	29	30			
IsCollidedKey(), CollideKey()	IsCollidedPortal(),CollidePortal()	CheckGameOver()				
12월						
일	월	화	수	목	금	토
				1	2	3
				문제 분석 및 디버깅		
4	5	6	7	8		
최종 디버깅				제출일		

2. 2020182032 이세민

11월						
일	월	화	수	목	금	토
		1	2	3	4	5
				기획서 1차 제출	플레이어 클래스 정리, UpdatePlayerInput() 키 이벤트	UpdatePlayerLocation() 이동
6	7	8	9	10	11	12
	ID 입력 화면 리소스 제작	기획서 2차 제출	SendPlayerData 구조체 구현	~~UpdateSendData() 구현~~	KeyInput 구조체 통신 구현	기획서 3.5차 제출
13	14	15	16	17	18	19
	~~ID 정보를 담은 SendPlayerDdata 송수신 구현~~ UpdateSendData() 구현	ID 정보를 담은 SendPlayerDdata 송수신 구현	15일까지 각 팀원들 구현한 내용들 검토, merge 전송 시간 계산 함수 구현		승패 결과 화면 리소스 제작, UpdateTime() 구현	각종 테스트 및 오류 수정
20	21	22	23	24	25	26

	타임 오버에 따른 CheckGameOver() 구현	클라이언트에게 bIsPlaying 전송, 서버 동기화	22일까지 각 팀원들 구현한 내용들 검토 및 Merge		IsPlayerWinner() 승패 판별 함수 구현 및 송수신	승패 결과 화면 렌더링 구현
27	28	29	30			
	열쇠 획득 CheckGameOver() 구현		모든 플레이어 게임 오버 시 CheckGameOver()			

12월						
일	월	화	수	목	금	토
				1	2	3
				문제 분석 및 디버깅		
4	5	6	7	8		
최종 디버깅				제출일		

3. 2020180034 정가온

11월						
일	월	화	수	목	금	토
		1	2	3	4	5
				기획서 1차 제출	코인클래스 구현 class Coin{}	플랫폼 클래스 구현 platform{}
6	7	8	9	10	11	12
		ScrollingCamera(); 기획서 2차 제출		기획서 3차 제출		기획서 3.5차 제출
13	14	15	16	17	18	19
	InitPlatform() 구현		15일까지 각 팀원들 구현한 내용들 검토 및 merge.	PlayerMgr 구조체 구현, InitPlayer() 구현	initMonster()구현 InitCoin() 구현	initServer() 서버를 생성하여 플레이어의 초기 위치를 선정하고 서버 데이터를 초기화
20	21	22	23	24	25	26
클라이언트	~~InitPlatfor~~	클라이언트	22일까지	수신받은 데		

에게 SendGame Data 정보전송	~~m() 구현~~ Recv_Thread 구현	~~에게~~ ~~SendGame Data~~ 정보전송 서버 동기화	각 팀원들 구현한 내용들 검토 및 Merge	이터를 토대로 클라이언트 렌더링 수정		
27	28	29	30			
	ChangeCamera() 플레이어 사망시 카메라 전환 구현		29일까지 각 팀원들 구현한 내용들 검토 및 merge.			
12월						
일	월	화	수	목	금	토
				1	2	3
				문제 분석 및 디버깅		
4	5	6	7	8		
최종 디버깅				제출일		