네트워크 게임프로그래밍 텀프로젝트

박신우 2021182014

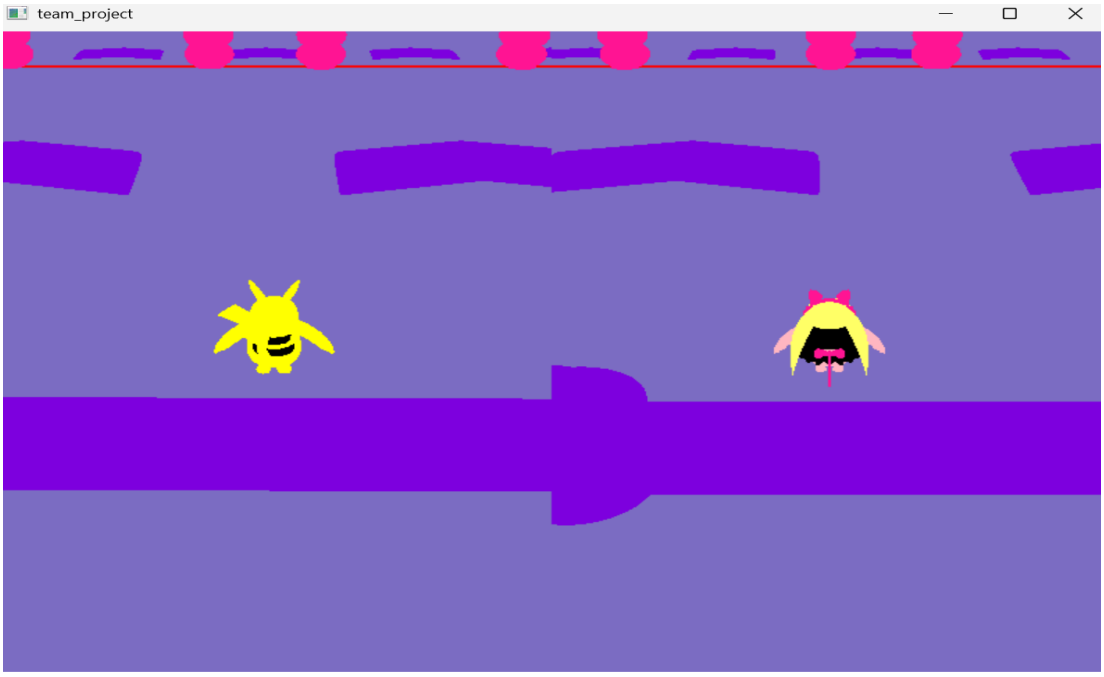
배주환 2021184015

안윤진 2023182021

목차

1.애플리케이션 기획

1.1 게임 소개



이 게임은 컴퓨터 그래픽스 과목에서 **배주환**, 조윤솔 학생이 작업한 게임으로 **'Fall Guys'** 게임을 모작하여 제작한 3D 장애물 경주 게임이다.

컴퓨터 그래픽스 과목에서 배운 **객체의 회전과 이동 변환** 개념을 중심으로 구현하기에 적합하다고 생각하여 이 게임을 만들게 되었다.

플레이어는 **W, A, S, D 키로 이동**, **스페이스바로 점프**할 수 있으며, 다양한 **회전·이동 장애물**을 피하면서 **골인 지점에 가장 먼저 도달하는 것**이 목표이다. 여러 플레이어가 동시에 경쟁하는 구조를 통해 **물리적 상호작용과 실시간 그래픽 표현**을 경험할 수 있다.

1.2 추가 기획 사항

- 뷰포트 분리, 캐릭터 하나만 플레이로 변경

현재 게임이 서버 없이 멀티플레이가 가능하도록 뷰포트가 2개로 분리되어 있어, 서버를 활용한 멀티플레이를 위해 뷰포트를 하나로 바꾸고 캐릭터 하나만 컨트롤 하도록 변경

- 캐릭터 추가

기존 게임에는 캐릭터 매쉬가 두개뿐이라 3인 플레이를 위해 새로운 캐릭터 추가

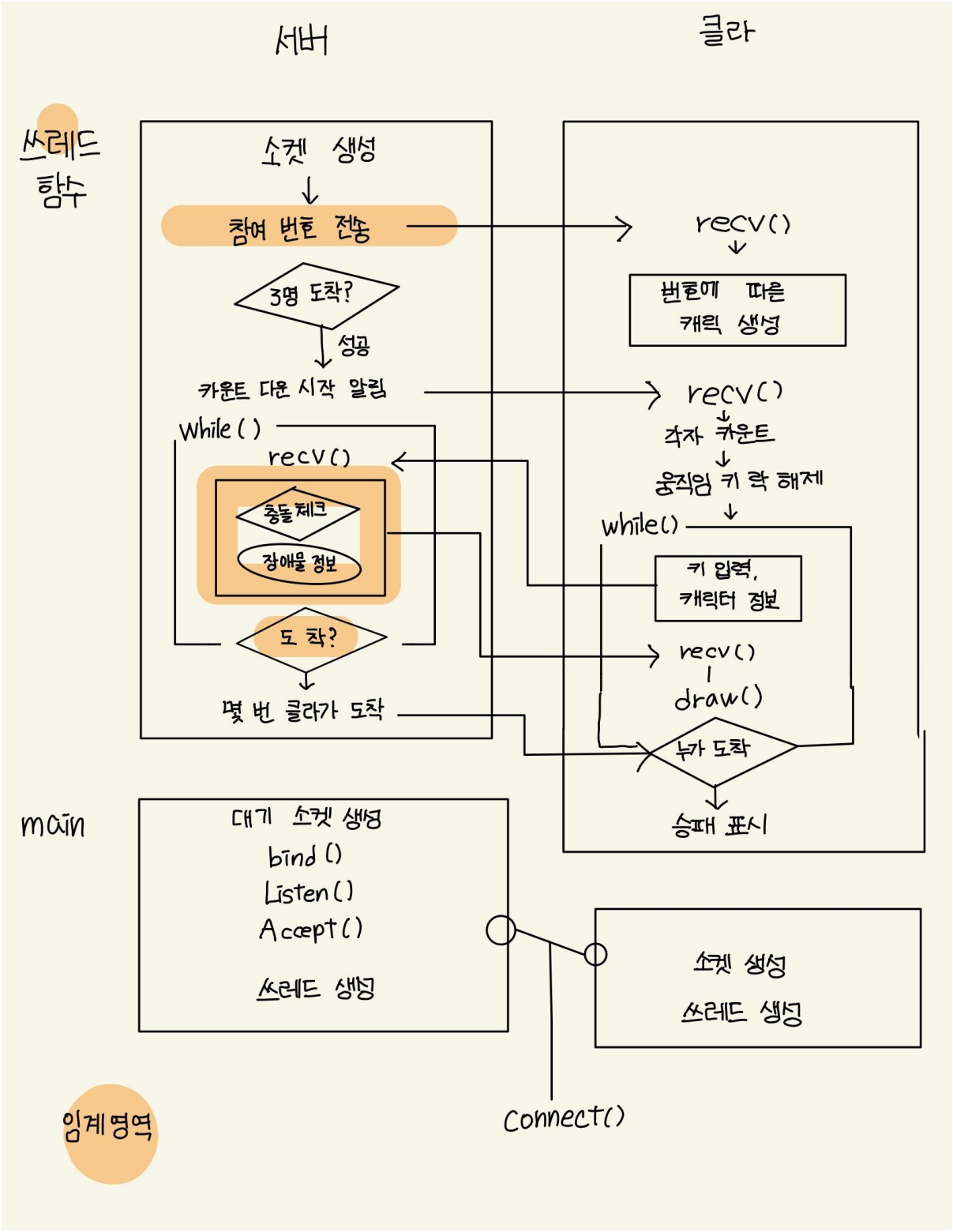
- 캐릭터 번호에 따른 캐릭터 생성

서버에서 캐릭터 번호를 받고 번호에 맞는 캐릭터를 생성

```
void Init CharacterByNum(int characterNum) {
    switch(characterNum) {
        case 1:
            // 캐릭터1에 해당하는 모델, 위치, 속성 초기화
            // ... 캐릭터1 관련 코드
            break;
        case 2:
            // 캐릭터2 초기화
```

```
            // ... 캐릭터2 관련 코드
                break;
        case 3:
            // 캐릭터3 초기화// ... 캐릭터3 관련 코드
            break;
        default:
            // ... 예외 처리
            break;
    }
}
```

2.High-Level Design



2-1. 충돌처리 로직

1. 클라이언트 → 서버 : 이동 요청

- 사용자가 이동 키 입력 시, Character 구조체 정보를 send()

2. 서버 : 이동 요청 수신 및 충돌 처리

- 서버에서 Character 구조체 정보 수신.
- 이동 방향 및 속도에 따라 임시 위치 계산.
- 해당 위치가 맵 내 충돌 객체(벽, 다른 캐릭터 등)와 충돌하는지 검사
- checkCollision 함수 사용

3. 서버 → 클라이언트 : 충돌 처리 결과 전송

- 서버가 Character구조체로 충돌 여부 및 최종 위치를 클라이언트에 send()

4. 클라이언트 : 결과 반영

- 클라이언트는 결과를 받아 캐릭터의 위치 및 상태를 갱신.

3.Low-Level Design

3-1. 전송을 위한 데이터 구조

플레이어 구조체

```
struct character {
    vec3 position;                  캐릭터의 위치정보를 담음
    vec3 direction;                 캐릭터의 방향정보를 담음
    GLfloat ArmLegSwingAngle;       캐릭터 팔다리 회전 정보
    Bool isCollision;               캐릭터의 충돌 유무
  };
```

장애물 구조체

```
struct MovingObstacle {
    vec3 position;              이동형 장애물의 위치정보를 담음
    vec3 direction;             이동형 장애물의 방향정보를 담음
};
struct RotatingObstacle {
    vec3 position;              회전 장애물의 위치정보를 담음
    GLfloat angel;              회전 장애물 회전정보
};
```

열거형 타입

```
Enum GameState{
```

WAITING_FOR_PLAYERS

IN_GAME

GAME_FINISHED

}

3-2. 스레드 함수

DWROD WINAPI ClientThread(LPVOID arg)

{

// 소켓 생성

// 클라 번호 전송

// 카운트 다운 시작 알림

while() {

// 캐릭터 정보 받기

// 임계영역 입장

// 캐릭터 정보 저장

// 충돌체크

// 캐릭터 정보 , 충돌체크 정보 복사본 만들어두기

// 도착 여부 판단 및 순위 기록

// 임계영역 떠나기

// 충돌체크 처리 결과 및 장애물 정보 보내기

If(도착했다면)

// 몇 번 클라가 먼저 도착했는지 알리기

}

}

3-3. 클라이언트 함수

void InitCharacterByNum(int characterNum)

서버에서 캐릭터 번호를 받고 번호에 맞는 캐릭터를 생성

GameState RenderCountDown() 모든 플레이어가 접속하면 게임시작 카운트다운 출력 함수

{

if(모든 플레이어가 접속 완료 했다면){

//3 ,2 1 랜더

MovingPossible() //캐릭터 움직임 막음 풀림

return IN_GAME

}

else

return WAITING_FOR_PLAYERS

}

void MovingPossible() 카운트 다운이 끝났을 때 캐릭터 움직임 허용 함수

void C2S_Character() 캐릭터에 대한 정보 서버에 전송하는 함수

{

// 캐릭터 정보 구조체 준비

// 서버로 보낼 패킷 생성

// 패킷에 캐릭터 정보 담기

// 서버에 패킷 전송

}

bool recv_character() 캐릭터에 대한 정보를 서버로부터 받는 함수

{

//서버로부터 캐릭터 정보 패킷 수신

//수신 잘 안됐다면 return false

//패킷에서 캐릭터 정보 추출

//캐릭터 정보 캐릭터 구조체에 저장

//캐릭터 정보를 이용해 render_character() 함수 호출

return true

}

bool recv_obstacale () 장애물에 대한 정보를 서버로부터 받는 함수

void RenderCharacter_1(character ch1) 1번 캐릭터에 대한 정보를 받고 그리는 함수

void RenderCharacter_2(character ch2) 2번 캐릭터에 대한 정보를 받고 그리는 함수

void RenderCharacter_3(character ch3) 3번 캐릭터에 대한 정보를 받고 그리는 함수

bool C2S isFinish() 결승점 도착정보 전송함수

{

If(결승점 도착){

// 결승점 도착 정보 패킷 생성

```
        // 서버에 패킷 전송
        // 잘 전송되었으면 true 리턴
        }
}
```

void moveCharacter() 충돌처리를 반영하여 캐릭터를 움직이는 함수

```
{
        If(서버에서 받은 정보에 충돌이 났다면){
                충돌을 반영해서 움직임 처리
        }
        else {
                기존 방식대로 움직임 처리
        ]
}
```

3-4. 서버 함수

bool CheckCollision(캐릭터 구조체) 캐릭터들 충돌 여부 판단 함수.

```
{
    if (캐릭터-캐릭터 충돌 || 캐릭터-장애물 충돌)
        return true;
    else
        return false;
}
```

int S2C_ClientOrder() 플레이어 참가순서 전송 함수

{

//접속 순서 번호 생성

int order = 현재 접속한 클라이언트 수 + 1;

클라이언트에게 순서 번호 전송

참가한 수 만큼 return

}

bool S2C_isPlayerReady() 플레이어 모두 접속 완료됐는지 전송 함수

bool S2C_obstacle() 장애물 회전 , 이동정보 전송 함수

{

각 클라이언트에게 장애물의 정보 구조체 전송

전송 잘 되었다면 true 아니면 false return

}

void S2C_IsFinish() 캐릭터가 도착 시 전송하는 함수

{

If(Check_IsFinish())

게임이 끝났다고 모든 클라이언트에게 전송

Else

게임 속행

}

bool Check_IsFinish(){

If(어떤 캐릭터가 도착했다면)

true

else

false

}

bool recv_character() 캐릭터에 대한 정보 클라이언트로부터 받는 함수

{

각각의 캐릭터들의 구조체를 받음

ch1. isCollision = CheckCollision(ch1) ;

ch2. isCollision = CheckCollision(ch2);

ch3. isCollision = CheckCollision(ch3);

//캐릭터 구조체 정보를 각 클라이언트로 보냄

for(캐릭터들 정보 다 보낼 때까지){

S2C_Character()

}

전송 잘 되었다면 true 아니면 false return

}

bool S2C_Character(캐릭터 구조체) 플레이어 정보 클라이언트로 전송 함수

{

각 클라이언트에게 캐릭터 정보 전송

전송 잘 되었다면 true 아니면 false return

}

4.개발 환경

컴파일러 : visual studio 최신버전

통신 프로토콜: TCP/IP

라이브러리: OpenGL

사용언어 : C/C++

버전관리툴 : GitHub

5.개발 일정

	일	월	화	수	목	금	토
1주차	10/26	10/27	10/28	10/29	10/30	10/31	11/1
박신우	시험기간					기획서 최종 수정	뷰포트 병합
배주환							
안윤진							
2주차	11/2	11/3	11/4	11/5	11/6	11/7	11/8
박신우	캐릭터 구조체 작성			InitCharacterByNum()		C2S_Character()	
배주환	3번째 캐릭터 추가			기존 프로젝트에서 서버, 클라 코드 분리		기존 프로젝트에서 서버, 클라 코드 분리	서버 연결
안윤진		캐릭터 리팩토링		장애물 리팩토링			코드 점검 및 오류수정

3주차	11/9	11/10	11/11	11/12	11/13	11/14	11/15
박신우	server_recv_character()		S2C_Character()			Client_recv_character()	
배주환	ClientThread()	S2C_Obstacle()		S2C_Obstacle()		S2C_Obstacle()	
안윤진	코드 점검 및 오류수정		S2C Client Order		void S2C_isPlayerReady()		테스트및 오류수정
4주차	11/16	11/17	11/18	11/19	11/20	11/21	11/22
박신우	Move character()		캐릭터 움직임 서버로 확인	캐릭터, 장애물 충돌처리			회전 장애물 충돌처리
배주환		S2C_Obstacle()		S2C_Obstacle()		S2C_Obstacle()	
안윤진	테스트 및 오류수정	카운트다운 에셋 모델링			void RenderCountDown()		void MovingPossible()
5주차	11/23	11/24	11/25	11/26	11/27	11/28	11/29
박신우	캐릭터, 캐릭터 충돌처리		충돌처리를 반영하여 캐릭터 움직임 처리			실습시간 확인	
배주환	recv_Obstacle()			장애물 checkcollision() 확인			
안윤진		void RenderCharacter_1,2,3(characterch()					도착 정보 보내기&받기
6주차	11/30	12/1	12/2	12/3	12/4	12/5	
박신우	안정성 테스트 및 오류수정					최종 제출	
배주환	추가 구현사항	최종검수			최종 검수		
안윤진	도착 정보 보내기 & 받기	추가 구현사항 및 오류 수정					

6.팀원 역할 분담

안윤진: 캐릭터, 장애물 리팩토링, S2C Client Order , void S2C_isPlayerReady() , void RenderCountDown(), void MovingPossible(), void RenderCharacter_1,2,3(character ch), 도착정보 보내기, 받기

배주환: 클라, 서버 코드 분리, 3번째 캐릭터 만들기, S2CObstacle , 장애물 정보 받기, ClientThraed()

박신우: 뷰포트 병합 InitCharacterByNum, 충돌처리, moveCharacter , 캐릭터 정보 보내기, 받기