
네트워크 게임 프로그래밍

추진 계획서



학번	이름
2020180051	임성규 (PM)
2020182050	박찬호
2018184019	윤주성

[목 차]

1. 애플리케이션 기획	3
1.0 게임 개요	3
1.1 게임 소개	3
1.2 게임 흐름	4
2. 개발환경	5
3. High-Level Design	6
3.1 다이어그램	6
3.2 서버 구현 내용	7
3.3 클라이언트 구현 내용	8
4. Low-Level Design	9
4.1 데이터를 전송할 때 사용할 패킷	9
4.2 서버	11
4.3 클라이언트	12
4.4 스레드 동기화	12
5. 팀원 역할 분담	13
5.1 서버	13
5.2 클라이언트	13
6. 개발 일정	14

1. 애플리케이션 기획

1.0 개요

윤주성 학생과 임성규 학생이 2022년 2학기 컴퓨터그래픽스 과목을 수강했을 당시, 기말 텀프로젝트로 제작한 1인 레이싱 게임입니다.

1.1 게임 소개

① 게임 이름 : RUN

② 게임 장르 : 2D 3인 레이싱 게임

③ 게임 컨셉: 장애물을 피해 다른 플레이어보다 먼저 결승점에 도착한다.

④ 게임 규칙

- 3명의 플레이어로 구성된다.

- 플레이어는 맵의 구멍을 피해 결승선까지 달린다.

- 플레이어는 상하좌우로 움직일 수 있다.

- 옆면과 충돌했을 경우 맵이 회전한다.

- 구멍에 떨어지거나 맵이 회전할 경우, 지연시간이 발생한다.

- 결승선에 동시에 도달할 경우 공동우승하는 것으로 판정한다.

- 모든 플레이어가 결승점에 도달했을 경우, 게임이 종료된다.

⑤ 조작 방법

- 이동 : 방향키

- 점프 : 스페이스바

1.2 게임 흐름

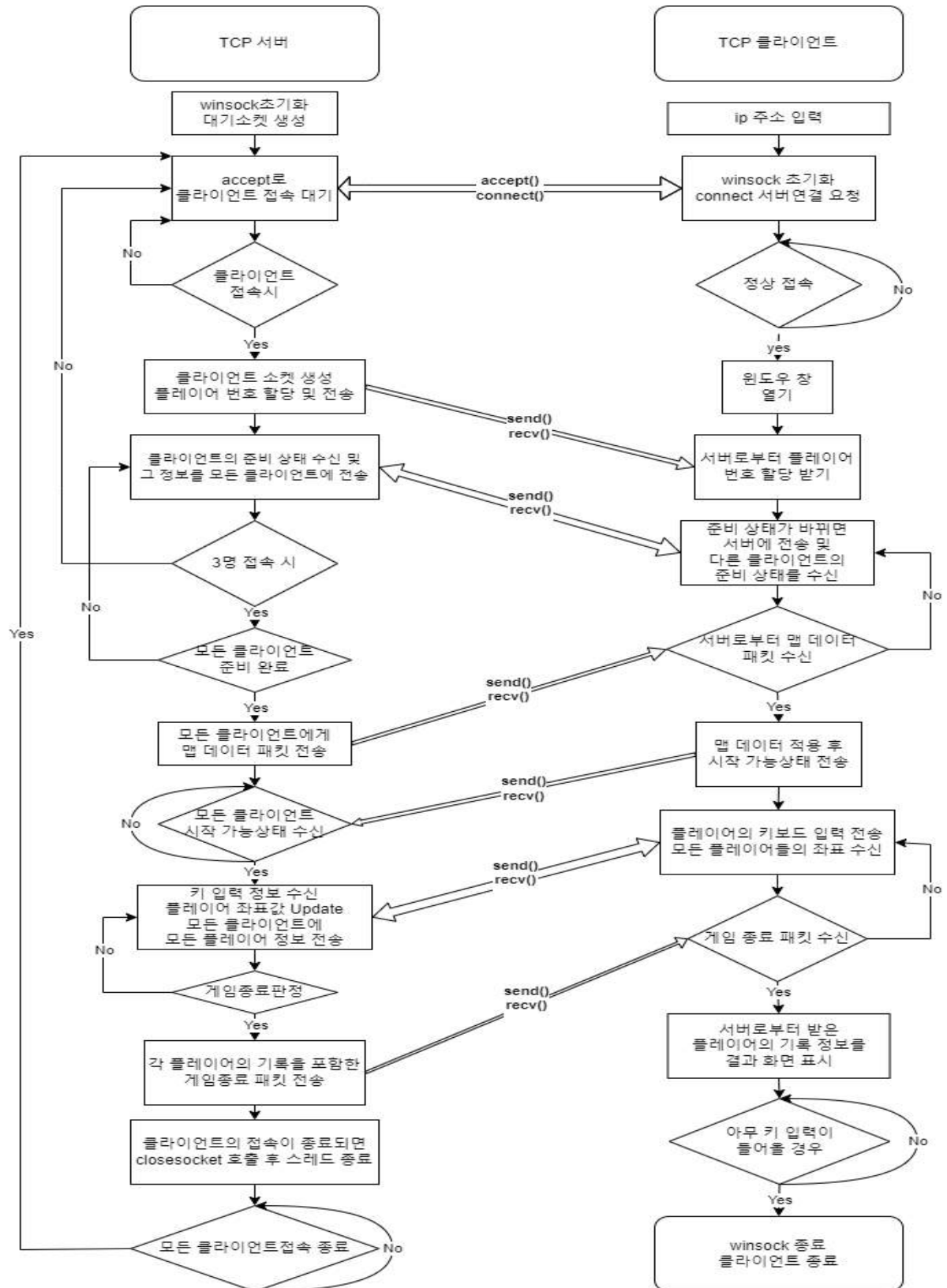
화면	설명
콘솔 화면	<ul style="list-style-type: none"> - 콘솔창에 서버의 IP 주소를 입력한다. - 서버 IP 주소를 입력하고 로비화면으로 넘어간다.
로비 화면	<ul style="list-style-type: none"> - 서버에 접속한 순서대로 플레이어 번호가 할당되고, 본인의 번호가 강조되고 레디버튼을 누르면 번호 옆에 레디상태 출력한다. - 3명의 클라이언트가 접속하고 모든 클라이언트가 READY 버튼을 누르면 게임을 시작한다.
게임 플레이 화면	 <ul style="list-style-type: none"> - 장애물을 피해 다른 플레이어보다 먼저 결승점에 도착한다.
결과 화면	<ul style="list-style-type: none"> - 모든 플레이어가 결승점에 도달했을 때 결과화면을 출력한다. - 플레이어의 번호와 기록을 화면에 출력하고, 본인의 플레이어 번호를 하이라이트 처리한다.

2. 개발 환경

- ▶ 운영체제 : windows 11
- ▶ 통신 프로토콜 : TCP/IP 가변길이 전송방식
- ▶ 사용 언어 : C++20
- ▶ 컴파일러 : Visual Studio 2022
- ▶ 라이브러리 : OpenGL, glew, freeglut
- ▶ 형상관리 툴 : github.com/

3. High-Level Design

3.1 다이어그램



3.2 서버 구현 내용

① 서버 실행

- ▶ 서버를 실행하면 winsock 초기화와 대기 소켓을 생성하고 bind 함수와 listen 함수를 통해 지역 IP 주소와 지역 포트 번호를 결정하고, 클라이언트의 접속을 기다리는 대기 소켓 상태로 만든다.
- ▶ 클라이언트가 connect 함수로 서버에 접근하면 accept된 클라이언트를 위한 소켓을 생성하며 그 후 접속한 순서대로 플레이어의 번호를 할당하고 클라이언트에게 전송한다.

② 플레이어 별 스레드를 이용한 데이터 통신

- ▶ 접속한 플레이어 별로 스레드가 할당되어 통신 소켓과 플레이어 번호를 할당받는다.
- ▶ 스레드 내부에서 클라이언트의 준비 상태를 받아오고, 상태가 변경되면 모든 클라이언트에게 변경 사항을 전송한다.
- ▶ 모든 클라이언트가 준비 완료 상태가 되면 모든 클라이언트에게 게임 시작 패킷을 전송한다.
- ▶ 게임 시작이후, 스레드 내부에서는 클라이언트가 보내온 키 입력 정보를 수신하여 플레이어들의 좌표 값을 수정하고, 충돌 검사를 한다.
- ▶ 게임 종료 조건(모든 플레이어가 결승선에 도달)에 맞는지 확인 후 각각의 클라이언트들에게 각 플레이어들의 기록을 포함한 게임 종료 패킷을 전송한다.
- ▶ 클라이언트의 접속이 종료되면 closesocket 함수 호출 후 스레드를 종료한다.
- ▶ 모든 스레드가 종료되면, 새로운 클라이언트의 접속을 대기한다.

3.3 클라이언트 구현 내용

① 콘솔 화면

- ▶ 콘솔 창에 접속할 서버의 IP 주소를 입력한다.
- ▶ winsock 초기화를 하고, connect 함수로 서버에 연결을 요청한다.

정상적으로 connect를 성공하면, 게임 윈도우 창을 열고 로비 화면을 띄운다.

② 로비 화면

- ▶ 서버에 접속해 있는 플레이어에 할당된 번호와 게임 준비 상태가 화면에 출력된다.
- ▶ 준비 버튼이 존재하며, 준비 버튼이 활성화/비활성화 될 때마다 다른 클라이언트와 동기화를 한다.
- ▶ 3명의 플레이어가 모두 READY 버튼을 누르면(게임 준비 완료 상태) 잠시 후 게임이 시작된다.

③ 게임 플레이 화면

- ▶ 랜덤하게 설정된 맵의 데이터를 받아온다.
- ▶ 게임에 입장하면 각 플레이어의 클라이언트는 서버에게서 플레이어들의 좌표 및 상태를 실시간으로 전송 받는다.
- ▶ 각 플레이어의 키보드 입력 정보와 플레이어의 상태 정보는 실시간으로 전송한다.
- ▶ 게임 종료 조건 만족 시 서버의 게임 종료 신호 및 승리한 플레이어 정보를 받아 결과 화면을 출력한다.

④ 결과 화면

- ▶ 서버로부터 플레이어의 기록 정보를 받아 결과 화면을 표시한다.

4. Low-Level Design

4.1 데이터를 전송할 때 사용할 패킷

서버에서 데이터 송신할 때 다음과 같은 클래스에 변수를 담아서 보낸다. 서버는 클라이언트에서의 구분을 위해 클래스 패킷의 길이 및 타입에 데이터를 붙여서 보낸다.

번호	이름	설명
1	<pre>struct SC_LOGIN_PACKET{ short size; char type; char playerid; };</pre>	4바이트 크기만큼 전송한다. 할당된 플레이어의 ID를 전송한다.
2	<pre>struct SC_READY_PACKET{ short size; char type; char playerid; bool ready; };</pre>	5바이트 크기만큼 전송한다. 해당 플레이어의 준비상태를 전송한다.
3	<pre>struct SC_MAP_DATA_PACKET{ short size; char type; float map_size[100][16]; };</pre>	6403바이트 크기만큼 전송한다. 맵 데이터를 전송한다.
4	<pre>struct SC_POSITION_PACKET{ short size char type; float x[3]; float y[3]; float z[3]; int map_index[3]; int bottom_index[3]; bool is_rotating[3]; float now_angle[3]; float bef_mv_x[3];</pre>	102바이트 크기만큼 전송한다. 각 클라이언트에게 모든 플레이어의 좌표를 전송한다.

	float bef_mv_y[3]; };	
5	struct SC_GAME_END_PACKET { short size char type; float end_time[3]; }	15바이트 크기만큼 전송한다. 각 클라이언트에게 모든 플레이어의 기록 정보를 전송한다.

클라이언트에서 데이터 송신할 때 다음과 같은 클래스에 변수를 담아서 보낸다. 클라이언트는 서버에서의 구분을 위해 클래스 패킷의 길이 및 타입에 데이터를 붙여서 보낸다.

번호	이름	설명
1	struct CS_READY { short size char type; };	3바이트 크기만큼 전송한다. 자신의 준비상태를 바꿀 수 있도록 flag를 전송한다.
2	struct CS_MAP_OK { short size char type; };	3바이트 크기만큼 전송한다. 맵 데이터를 정상적으로 수신했음을 알 수 있는 flag를 전송한다.
3	enum class KEY_EVENT : char { KEY_SPACE = 0, KEY_LEFT = 1, KEY_RIGHT = 2 }; struct CS_KEY_EVENT { short size char type; KEY_EVENT key; };	4바이트 크기만큼 전송한다. 서버에 Key_event를 전송한다.

4.2 서버

- 송신 함수

번호	이름	설명
1	<code>void send_sc_login_packet()</code>	서버에 접속한 클라이언트에게 접속한 순서대로 플레이어 번호를 할당하는 함수
2	<code>void send_sc_ready_packet(char client_id)</code>	로비에 접속해있는 각 클라이언트의 준비 상태를 전송하는 함수
3	<code>void send_sc_map_data_packet()</code>	모든 클라이언트에게 맵 데이터를 전송하는 함수
4	<code>void send_sc_position_packet()</code>	모든 클라이언트에게 모든 플레이어의 좌표 정보를 전송하는 함수
5	<code>void send_sc_game_end_packet()</code>	게임이 종료된 후, 모든 플레이어의 기록 정보를 모든 클라이언트에게 전송하는 함수

- 수신 함수

번호	이름	설명
1	<code>void RecvThread(SOCKET s)</code>	클라이언트별로 Recv 스레드를 생성하고, 모든 Recv 데이터를 처리하는 함수

- 기록 관리

class Timer		
멤버 변수	<code>std::chrono::steady_clock::time_point start_time</code>	클래스 생성될 때 시간 저장
	<code>std::chrono::steady_clock::time_point end_time[3]</code>	set_end_now 함수가 호출됐을 때 해당하는 인덱스에 시간 저장
멤버 함수	<code>void set_end_now(int client_id)</code>	end_time[client_id]에 현재 시간 저장
	<code>float get_record(int client_id)</code>	end_time[client_id] - start_time을 리턴한다.

4.3 클라이언트

- 송신 함수

번호	이름	설명
1	void send_cs_ready_packet()	자신의 준비 상태 변경을 서버에 전송하는 함수
2	void send_cs_map_ok_packet()	클라이언트가 맵 데이터를 정상적으로 수신했음을 서버에 전송하는 함수
3	void send_cs_key_event_packet (KEY_EVENT key)	자신의 Key Event(←,→,space)를 서버에 전송하는 함수

- 수신 함수

번호	이름	설명
1	void RecvThread (SOCKET s, CMap* map, std::mutex& m)	서버에서 데이터를 받았을 때 모든 데이터를 처리하고, 클라이언트를 업데이트 해주는 함수

4.4 스레드 동기화

번호	이름	설명
1	void RecvThread (SOCKET s)	서버에서 클라이언트 별로 스레드를 할당하고, 클라이언트가 보내는 데이터를 받아서 처리한다. std::mutex 객체를 사용하여 공유 자원을 보호한다.
2	void RecvThread (SOCKET s, CMap* map, std::mutex& m)	서버에서 보내는 모든 데이터를 처리하고, 게임을 업데이트 한다. 인자로 받은 std::mutex 객체를 사용하여 공유 자원을 보호한다.

5. 팀원 역할 분담

5.1 서버

서버 구현 내용	담당자		
	임성규	박찬호	윤주성
기본적인 네트워크 환경 구축	o		
프로토콜 정의	o		
RecvThread	o		
send_sc_login_packet			o
send_sc_ready_packet		o	
send_sc_map_data_packet		o	
send_sc_position_packet		o	
send_sc_game_end_packet		o	
Class Timer 구현		o	
Update 구현	o		

5.2 클라이언트

클라이언트 구현 내용	담당자		
	임성규	박찬호	윤주성
게임 리소스 모델링 및 디자인			o
클라이언트 렌더링 구현	o		
RecvThread	o		
send_cs_ready_packet			o
send_cs_map_ok_packet			o
send_cs_key_event_packet			o

6. 개발 일정

11월

	일	월	화	수	목	금	토
			31	1	2	3	4
임성규			네트워크 환경구축		네트워크 환경구축		
박찬호				send_sc_ready_packet 구현			
윤주성			send_sc_login_packet 구현	게임 리소스 모델링 및 디자인			
	5	6	7	8	9	10	11
임성규			프로토콜 정의	Server RecvThread 구현	Server RecvThread 구현		
박찬호		send_sc_map_data_packet 구현		send_sc_position_packet 구현			
윤주성			send_cs_ready_packet 구현	send_cs_ready_packet 구현			
	12	13	14	15	16	17	18
임성규			Server Update 구현		Client 렌더링 구현		
박찬호		send_sc_game_end_packet 구현		Timer::set_end_now 구현			
윤주성			send_cs_map_ok_packet 구현	send_cs_map_ok_packet 구현			
	19	20	21	22	23	24	25
임성규			Client RecvThread 구현		Client RecvThread 구현		
박찬호		Timer::get_record 구현					
윤주성			send_cs_key_event_packet 구현	send_cs_key_event_packet 구현			
	26	27	28	29	30		
임성규		디버깅	디버깅	디버깅	디버깅		
박찬호		디버깅	디버깅	디버깅	디버깅		
윤주성		디버깅	디버깅	디버깅	디버깅		

12월

	일	월	화	수	목	금	토
						1	2
임성규							
박찬호							
윤주성							
	4	5	6	7	8		
임성규		디버깅	디버깅	디버깅	최종 점수인		
박찬호		디버깅	디버깅	디버깅			
윤주성		디버깅	디버깅	디버깅			