네트워크 게임 프로그래밍

Term Project

2019180020 설찬형

2019180004 구윤성

**1. 게임 기획**

구윤성 학우가 윈도우 프로그래밍에서 c++로 만든 프로젝트, 쿠키런 모작을

이용할 예정입니다.

**2. 게임 설명**

쿠키런 모작으로, 쿠키가 달리면서 코인을 먹을 시에 점수를 얻도록 설정하였습니다. 이를

네트워크 통신을 이용해 2인이 게임에 참여할 수 있도록 바꿀 예정입니다. 실행 방법은

스페이스바를 이용해 점프를 하는 방식입니다. 시작할 때, 5개의 목숨으로 시작을 하며,

낙사의 경우에는 목숨이 다 사라집니다. 오브젝트는 4가지가 있으며, 코인을 먹을 시에는

게임이 끝나고 코인이 점수로 환산되며, 물약을 먹을 땐 일정 시간 동안 캐릭터의

크기를 키웁니다. 바나나는 일정 시간 동안 캐릭터의 속도를 느리게 해주며, 적의 경우,

적의 상단을 점프를 이용해 밟게 될 경우, 적에게 피해를 주며 적이 사라지고, 측면에

피격될 경우, 목숨을 하나씩 잃습니다. 이를 네트워크 통신을 이용해 상대의 화면을 보면서

상대의 목숨과 코인을 확인하며 자신의 플레이를 안정적이거나 모험적으로 플레이를

할 수 있도록 하고자 합니다.



현재 인게임 스크린샷 2인 플레이 시 인게임 스크린샷

**3. High-Level Design**

**Server**

**도표, 스케치, 기술 도면, 평면도이(가) 표시된 사진

자동 생성된 설명**

**Client**

**도표, 기술 도면, 평면도, 텍스트이(가) 표시된 사진

자동 생성된 설명**

**4. Low-Level Design**

**서버**

1. 구조체

struct CLIENT:

SOCKET c\_socket 클라이언트 소켓

float x x, y 좌표

float y

struct PACKET:

int size 패킷 크기

int type 패킷 타입 (접속 / 종료 / 위치)

DATA data 데이터 부분 (x, y)

1. 전역 변수

#define SERVERPORT

서버포트

#define BUFSIZE

버퍼 사이즈

SOCKET server\_socket

서버 local 소켓

array <CLIENT, 2> Clients

서버에서 관리하는 클라이언트들의 정보를 담는 공유 데이터

Bool IsReady

2명의 클라이언트 접속을 확인하는 변수

True : 게임 준비

False : 계속 새 클라이언트를 받음

Bool IsRunning

게임 종료 여부를 확인하는 변수

True : 게임 중

False : 게임 종료

1. 스레드 함수

DWORD WINAPI RecvThread(LPVOID arg)

지역 변수:

- SOCKET client\_socket recv를 받을 클라이언트의 소켓

- sockaddr\_in clientaddr 클라이언트 주소 구조체

- char addr[INET\_ADDSTRLEN] 클라이언트 주소 문자열

- PACKET packet recv에 사용할 버퍼

- int retval recv함수의 반환 값

1. while(true)문을 반복하면서 recv함수를 호출한다
2. retval의 값에 따라 접속을 종료할 지 패킷을 수신할 지 결정한다.
3. retval가 true이면, packet에 담긴 정보를 바탕으로 공유데이터를 변경한다.
4. 이 때, 다른 함수들이 공유데이터에 접근하므로 lock을 걸어준다.
5. 변경이 완료되면 lock을 해제한다
6. 기타 함수

bool InitServer()

지역변수:

WSADATA was winsock 구조체

sockaddr\_in serveraddr 서버 주소 구조체

int server\_retval bind / listen 반환 값

void InitGame()

게임 초기화 함수

void EndGame()

게임 종료 함수

void CheckData()

클라이언트 데이터를 확인하고 충돌처리를 담당하는 함수

void SendData(int sendtype)

지역변수:

PACKET packet 보낼 데이터 패킷

1. 새로운 패킷을 하나 생성하고 클라이언트데이터를 읽어 패킷에 담는다.
2. 이 때, 패킷의 type은 인자 sendtype으로 보낸다.
3. 모든 클라이언트에게 변경된 데이터를 보낸다

**클라이언트**

1. 전역 변수

#define SERVERPORT

서버포트

#define BUFSIZE

버퍼 사이즈

SOCKET client\_socket

클라이언트 local 소켓

HANDLE hRecvThread

Recv스레드 핸들

Bool IsReady

2명의 클라이언트 접속을 확인하는 변수

True : 게임 준비

False : 계속 새 클라이언트를 받음

Bool IsRunning

게임 종료 여부를 확인하는 변수

True : 게임 중

False : 게임 종료

1. 스레드 함수

DWORD WINAPI RecvThread(LPVOID arg)

지역 변수:

- SOCKET client\_socket recv를 받을 클라이언트의 소켓

- PACKET packet recv에 사용할 버퍼

- int retval recv함수의 반환 값

1. while(true)문을 반복하면서 recv함수를 호출한다
2. retval의 값에 따라 접속을 종료할 지 패킷을 수신할 지 결정한다.
3. retval가 true이면, packet에 담긴 정보를 바탕으로 데이터를 변경한다.
4. 이 때, 변경을 완료할 때까지 변경할 데이터에 lock을 걸어둔다
5. 기타 함수

bool InitClient()

지역변수:

WSADATA was winsock 구조체

sockaddr\_in serveraddr 서버 주소 구조체

int retval bind / listen 반환 값

void SendData(int sendtype)

지역변수:

PACKET packet 보낼 데이터 패킷

1. 새로운 패킷을 하나 생성하고 데이터를 읽어 패킷에 담는다.
2. 이 때, 패킷의 type은 인자 sendtype으로 서버로 패킷을 보낸다.

**스레드 동기화 과정**

서버

1. 1번 클라이언트 스레드에서 상태 메시지를 수신 후 공유 버퍼에 쓴다.
2. 서버 메인 스레드에서 클라이언트 A에게서 온 상태 메시지를 처리하고 게임 로직을 수행한다
3. 2번 클라이언트 스레드에서 상태 메시지를 수신 후 공유 버퍼에 쓴다.
4. 3번 스레드에서 클라이언트 B에게서 온 상태 메시지를 처리하고 게임 로직을 수행한다
5. 1~4의 내용을 반복하며 4번 단계에서 게임오버를 판단해 승패를 결정한다.
6. 위 순서는 이벤트를 통해 제어한다.

클라이언트

1. recv 스레드에서 서버에게서 온 상태 메시지를 공유버퍼에 쓴다.
2. 메인 스레드에서 상태 메시지를 처리하고 게임로직을 수행한다.
3. 1~2번을 반복하며 1번 단계에서 게임오버 메시지를 수신하면 메시지를 공유버퍼에 쓰고 수신 스레드를 종료하며, 이후 메인스레드에서 게임오버 메시지를 처리한다.
4. 1~2번 과정의 순서는 이벤트를 통해 제어한다.

**5. 팀원별 역할 분담**

구윤성: 서버 메인 스레드 설계, 서버 상태 메시지 수신 스레드 설계

설찬형: 클라이언트 메인 스레드 설계, 클라이언트 상태 메시지 수신 스레드 설계,

서버 패킷 처리 설계

**6. 개발 환경**

언어: C++

개발 도구: Visual Studio 2022

운영 체제: Windows

**7. 개발 일정**

설찬형 (클라이언트 메인 스레드 설계, 클라이언트 상태 메시지 수신 스레드 설계,   
서버 패킷 설계)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | 10/31 | 11/1 | 2 |
| 3 | 4 | 5  Player class  구현  이동과 쿠키  상태 등 Input 명령 | 6  Player class  구현  1p, 2p를  구분하는  함수 구현 | 7  Player class  구현  Player Update  함수 구현 | 8 | 9  Player  상태 메시지  구현 |
| 10  1주차 피드백 및 부족한 진도 보강 | 11 | 12 | 13  충돌 체크관련  함수 구현 | 14  충돌 체크  상태 메시지  구현 | 15 | 16 |
| 17  2주차 피드백  및 부족한  진도 보강 | 18 | 19 | 20  상태 메시지  수신 스레드  구현 | 21 | 22  게임 재시작 관련 구현 설계 | 23 |
| 24  3주차 피드백 및 부족한 진도 보강 | 25 | 26  서버 패킷 설계  (상태 메시지 송수신 처리) | 27  서버 패킷 처리 관련 로직 구현 | 28 | 29  서버 패킷 처리 관련 로직 구현 | 30  서버 패킷 처리 관련 로직 구현 |
| 12/1  4주차 피드백 및 부족한 진도 보강 | 2 | 3  랭킹 관련 함수 구현 | 4  랭킹 관련  함수 구현 | 5 | 6  코드 최적화 | 7  코드 리팩토링 |
| 8  5주차 피드백  및 부족한 진도 보강 | 9  코드 최적화  및 리팩토링 | 10  디버깅 및  버그수정 | 11  디버깅 및  버그수정 | 12  디버깅 및  버그수정 | 13 |  |

구윤성 (서버 메인 스레드 설계, 서버 상태 메시지 수신 스레드 설계)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | 10/31 | 11/1 | 2 |
| 3 | 4 | 5  InitServer  생성 | 6  클라이언트 2인용으로  재구성 | 7  클라이언트 2인용으로  재구성 | 8 | 9 |
| 10  1주차 피드백 및 부족한 진도 보강 | 11  서버  Main스레드  생성 | 12 | 13  서버 패킷 수신 구현 | 14  클라이언트  처리 담당  스레드 생성 | 15 | 16 |
| 17  2주차 피드백  및 부족한  진도 보강 | 18 | 19  서버 패킷 송신 구현  (1p, 2p) | 20 | 21 | 22  서버 패킷 recv 처리 구현 | 23 |
| 24  3주차 피드백 및 부족한 진도 보강 | 25 | 26  상태 메시지  수신 구현  RecvThread | 27 | 28  상태 메시지  수신 구현  RecvThread | 29 | 30  코드 정리 및 점검 |
| 12/1  4주차 피드백 및 부족한 진도 보강 | 2 | 3  게임 종료 관련 함수  구현 | 4 | 5  랭킹 관련  함수 구현 | 6  코드 최적화 | 7  코드 리팩토링 |
| 8  5주차 피드백  및 부족한 진도 보강 | 9  코드 최적화  및 리팩토링 | 10  디버깅 및  버그수정 | 11  디버깅 및  버그수정 | 12  디버깅 및  버그수정 | 13 |  |