네트워크프로그래밍

팀 프로젝트 과제 (2018학년도 1학기)

과 목 명 : 네트워크프로그래밍

교 수 명 : 정 의훈

학 생 명 : 2015156048 황다솔

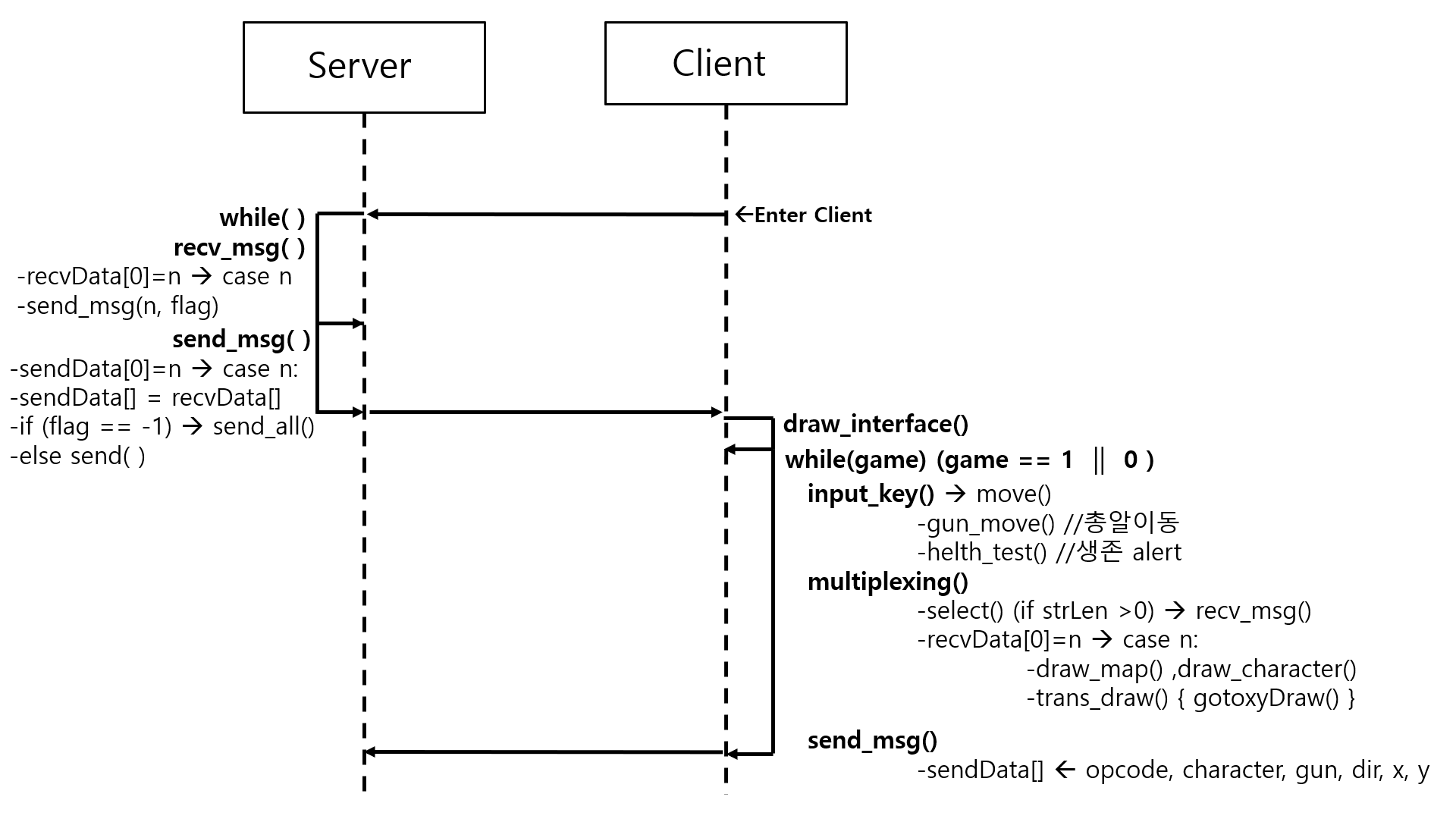
2016152043 안덕규

2016152025 양수연

개요

1. 시퀀스 다이어그램
2. recv\_msg
3. send\_msg
4. 실행 화면

시퀀스 다이어그램



rECV\_msg()

다음에 오는 설명은 클라이언트의 recv\_msg()를 토대로 작성하였다. recv\_msg는 서버의 send\_msg와 상응하여 서버의 send\_msg()는 생략한다.

case 1: 🡪 대기열 접속중

|  |  |  |
| --- | --- | --- |
| recvData[] | 0 | 1 |
| 값 | 1 | 참여인원 |

1. 대기열 인원 ready\_num 과 recvData[1]을 동기화 해주고, gotoxyDraw()함수를 통해interface부분에 참여인원과 입장 가능한 인원을 시각화 한다.
2. 이때 reay\_timer는 3초의 대기시간으로 새로운 대기자가 등장했을 때 case 1의 for문을 거쳐 대기시간을 3초로 다시 초기화 한다.

case 2: 🡪 게임시작

|  |  |  |  |
| --- | --- | --- | --- |
| recvData[] | 0 | 1 | 2 ~ Map size+2 |
| 값 | 2 | 해당 클라이언트 번호 | 맵 정보 |

1. character변수에 각각 부여된 클라이언트 번호 중 해당 클라이언트 변호를 부여한다.
2. for문을 통해 recvData 배열 속 맵정보를 map(맵정보를 저장하는) 배열에 저장한다.
3. draw\_map(), draw\_character()로 맵과 캐릭터를 그린다.

case 3: 🡪 움직임

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| recvData[] | 0 | 1 | 2 | 3 | 4 |
| 값 | 3 | 해당 클라이언트 번호 | dir | 삭제할 x좌표 | 삭제할 y좌표 |

1. 캐릭터가 있었던 곳의 궤적을 삭제하여 변화된 곳을 그린다.
2. trans\_draw(int num, int dir)을 매개변수로, num은 변화할 대상이 캐릭터인지, 총알인지, 체력인지 판단하여 paint의 값을 결정한다.
3. 총알인 경우
   1. gotoxyDraw()로 콘솔창에 총알을 그린다.
4. UP
   1. UP하려는 대상이 캐릭터이면 원래 있던 곳을 초기화 한 후 새 좌표에 등록한다.

map[recvData[3]][recvData[4]] = 0; 🡪 map배열에서 현 위치 초기화

map[recvData[3]][recvData[4] - 1] = recvData[1]; 🡪 map배열에 해당 클라이언트 번호 입력

* 1. 캐릭터가 아니라면, gotoxyDraw()함수로 해당되는 x, y좌표에 빈 공간을 그린다. recvData[0]==5인 총알인 경우에는 총알을 그리고, recvData[0]==8인 나무 파괴의 경우에 recvData[1]의 값이 1이상이면, 아이템이 존재하므로 4를 더하여 콘솔 창에 표시될 아이템의 인덱스와 매칭시킨다.
  2. recvData[1]에 4를 더하여 map배열에 총알, 아이템 (추가 총알, 체력)의 값을 등록한다.
  3. 이 값은 draw\_map을 통하여 콘솔창에 나타난다.

RIGHT, DOWN, LEFT 모두 UP과 같은 절차를 거친다.

1. RIGHT
2. DOWN
3. LEFT

case 4: 🡪 총알생성

|  |  |  |  |
| --- | --- | --- | --- |
| recvData | 0 | 1 | 2 |
| 값 | 4 | 총알 X좌표 | 총알 Y좌표 |

1. trans\_draw(2,0)의 첫 번째 파라미터 num==2이면, paint=총알의 경우이므로 gotoxyDraw함수로 콘솔창에 총알을 그린다.

case 5: 🡪총알이동

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| recvData[] | 0 | 1 | 2 | 3 |
| 값 | 5 | dir(상하좌우) | 총알 X좌표 | 총알 Y좌표 |

1. trans\_draw(2,recvData[1])의 첫 번째 파라미터 int num==2이면, paint=총알의 경우이므로 recvData[1] 값인 이동한 방향에 따라 콘솔창에 총알을 나타낸다.

case 6:🡪 총알파괴

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| recvData[] | 0 | 1 | 2 | 3 |
| 값 | 6 | dir | 총알 X좌표 | 총알 Y좌표 |

1. trans\_draw(2,recvData[1])의 첫 번째 파라미터 int num==2이면, paint=총알의 경우이므로, 콘솔창에 오른쪽으로 이동한 총알을 그린다.

case 7:🡪 총알 이동 X, 나무만 파괴

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| recvData[] | 0 | 1 | 2 | 3 |
| 값 | 7 | 아이템  0 : 빈칸  1 : 추가총알  2 :체력 | 나무 X좌표 | 나무 Y좌표 |

1. 아이템이 존재한다면 trans\_draw()함수를 통해 map에서 나무를 지우고, 빈 공간을 아이템으로 갱신하여 콘솔에 표시한다.

case 8:🡪 총알 발사, 나무 파괴

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| recvData[] | 0 | 1 | 2 | 3 | 4 |
| 값 | 8 | 아이템  0 : 빈칸  1 : 추가총알  2 :체력 | 총알 X좌표 | 총알 Y좌표 | dir |

1. recvData[0]==8인 경우에 recvData[1]의 값이 0이 아니면 아이템이 존재하는 경우이므로 recvData[1]에 4를 더하여 draw\_map()에서 총알을 그릴 것인지, 아이템을 그릴 것인지 map배열에 저장한다. recvData[1]은 다음과 같은 값으로 draw\_map[()을 통해 콘솔창에 나타난다.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| recvData[1] | 빈 공간 | 벽 ■ | 나무 △ | 캐릭터 ⊙ | 총알 ¤ | 추가총알 ▒ | 체력 ♡ |

case 9: 🡪총알 삭제, 캐릭터 이동

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| recvData[] | 0 | 1 | 2 | 3 |
| 값 | 9 | 체력이 감소할 캐릭터 | X좌표 | Y좌표 |

1. character변수와 비교하여 해당 클라이언트 체력(health)을 감소시키고, trands\_draw\_interface(3)로 콘솔창의 체력을(♥->♡) 갱신한다.

case 10: 🡪총알 캐릭터 충돌

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| recvData[] | 0 | 1 | 2 | 3 |
| 값 | 10 | dir | 총알 X좌표 | 총알 Y좌표 |

1. 발사된 총알의 좌표와 해당 클라이언트의 좌표가 같다면 총알과 캐릭터가 충돌 한 것이므로 해당 클라이언트의 체력을 감소시킨다.

case 11: 🡪 나간 클라이언트 검색

|  |  |
| --- | --- |
| recvData[] | 0 |
| 값 | 11 |

1. 해당 클라이언트는 sendData[1]에 자신의 character값을 넣고 send를 통해 전송한다.

case 12:🡪 생존한 클라이언트

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| recvData[] | 0 | 1 | 2 | 3 | 4 |
| 값 | 12 | User\_count | 캐릭터1 | 캐릭터2 | 캐릭터3 |

1. for문을 통해 현재 생존한 클라이언트를 검색하고, 해당 인덱스에 recvData가 1로 채워져 있다면 클라이언트가 생존해 있는 상황이므로, user\_check배열의 해당 인덱스의 값을 0 🡪 1로 표시해준다.
2. user\_check배열의 값이 0인 인덱스가 있다면, 클라이언트의 공백이 존재하는 것이므로 해당 클라이언트를 user\_delete()함수를 통해 콘솔창에서 삭제한다.

case 13: 🡪 사망

|  |  |  |
| --- | --- | --- |
| recvData[] | 0 | 1 |
| 값 | 13 | dir |

1. 유저가 사망했다면, user\_die()함수를 호출하여 map에서 해당 유저를 지운 후, 콘솔창에서도 삭제한다.

case 14: 🡪 승리

|  |  |
| --- | --- |
| recvData[] | 0 |
| 값 | 14 |

1. 유저가 승리했다면, user\_victory()함수를 호출하여 인터페이스에 띄운다.

Send\_msg()

send\_msg는 서버의 recv\_msg와 상응하여 서버의 recv\_msg()는 생략한다.

case 1 : 🡪 대기 열 접속 시작

1. strLen = 1 을 넣고 send 메시지를 보낸다.
2. 대기열 인원 증가할 때 마다 서버에 ready\_num 을 1 증가해준다.
3. send\_msg 에서 case 1을 실행한다. (대기열 접속)

case 2 : 🡪 대기시간 종료

1. system("cls")을 호출한다.
2. send\_msg 에서 case 2를 실행한다.(게임시작)

case 3 : 🡪 이동

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| sendData[] | 1 | 2 | 3 | 4 |
| 값 | map[X][Y] | dir(상하좌우) | 총알 X좌표 | 총알 Y좌표 |

1. send\_msg 에서 case 3를 실행한다.(움직임)

case 4 : -> 총알 생성

|  |  |  |
| --- | --- | --- |
| sendData[] | 1 | 2 |
| 값 | gun[i][2]  총알의 X 좌표 | gun[i][3]  총알의 Y 좌표 |

1. 대기중인 총알을 검색하기 위해 총알을 검사한다.
2. send\_msg 에서 case 4를 실행한다.(총알 생성)

case 5 : -> 총알 이동

|  |  |  |  |
| --- | --- | --- | --- |
| sendData[] | 1 | 2 | 3 |
| 값 | gun[i][1] 총알 방향 | gun[i][2] 총알 X 좌표 | gun[i][3] 총알 Y 좌표 |

1. send\_msg 에서 case 5를 실행한다. (총알 이동)

case 6 : -> 총알 파괴

|  |  |  |  |
| --- | --- | --- | --- |
| sendData[] | 1 | 2 | 3 |
| 값 | gun[i][1] 총알 방향 | gun[i][2] 총알 X | gun[i][3총알 Y |

1. send\_msg 에서 case 6을 실행한다. (총알 파괴)

case 7 : -> 나무만 파괴 ( 나무 앞에서 총알을 발사 했을 때)

|  |  |  |
| --- | --- | --- |
| sendData[] | 1 | 2 |
| 값 | 총알 X좌표 | 총알 Y좌표 |

1. switch 문으로 총알의 방향을 알아낸다.
2. send\_msg 에서 case 7를 실행한다. (나무 파괴)

case 8 : -> 나무 파괴 ( 총알 이동 ) ;

|  |  |  |  |
| --- | --- | --- | --- |
| sendData[] | 1 | 2 | 3 |
| 값 | gun[i][1]  총알 방향 | gun[i][2]  총알 X | gun[i][3  총알 Y |

1. send\_msg 에서 case 8를 실행한다.(총알 발사, 나무 파괴)

case 9 : -> 제자리 총알 발사

|  |  |  |  |
| --- | --- | --- | --- |
| sendData[] | 1 | 2 | 3 |
| 값 | Map[x][y] 체력을 감소시킬 캐릭터 | x | y |

1. send\_msg 에서 case 9를 실행한다. (총알 삭제, 캐릭터 이동)

case 10 : -> 총알 캐릭터 충돌

|  |  |  |  |
| --- | --- | --- | --- |
| sendData[] | 1 | 2 | 3 |
| 값 | gun[i][1] 총알 방향 | gun[i][2] 총알 X | gun[i][3] 총알 Y |

1. send\_msg 에서 case 10을 실행한다.(총알 캐릭터 충돌)

case 11 : -> 생존 신고

|  |  |
| --- | --- |
| sendData[] | 1 |
| 값 | Character 생존해 있는 캐릭터 |

1. user\_count 와 ready\_num(참여인원)을 비교하여 수가 같다면 case 12를 실행한다.
2. 생존한 클라이언트를 확인한다.
3. ① user\_count 와 ready\_num(참여인원)을 비교하여 참여인원이 1명이라면
4. Case 14문을 실행하여 승리하였다는 메세지를 띄운다.

case 13: -> 사망

|  |  |
| --- | --- |
| sendData[] | 1 |
| 값 | Character사망한 캐릭터 |

1. send\_msg 에서 case 13을 실행한다.

실행 화면

<1> 대기열

1. void title() 함수를 이용하여 대기열 화면을 출력한다.

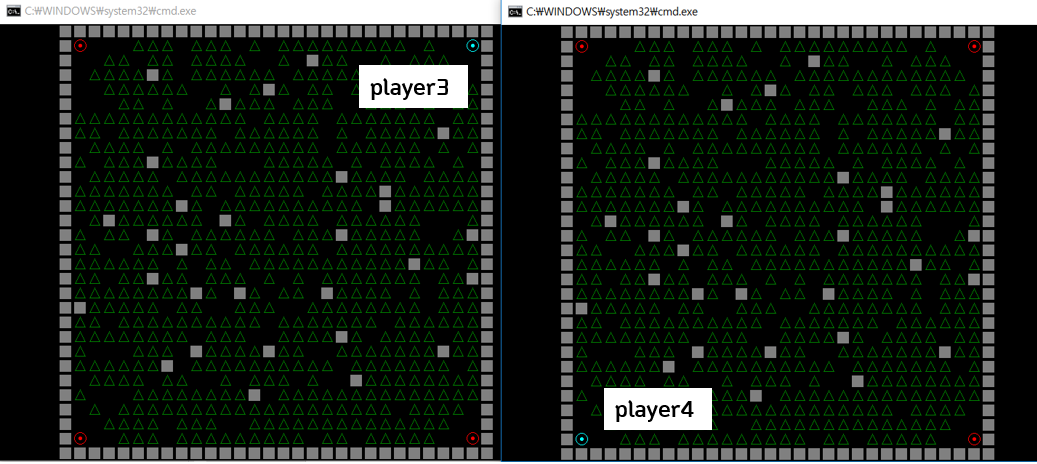
2. multiplexing()을 통해 새로 들어오는 클라이언트를 확인한다.

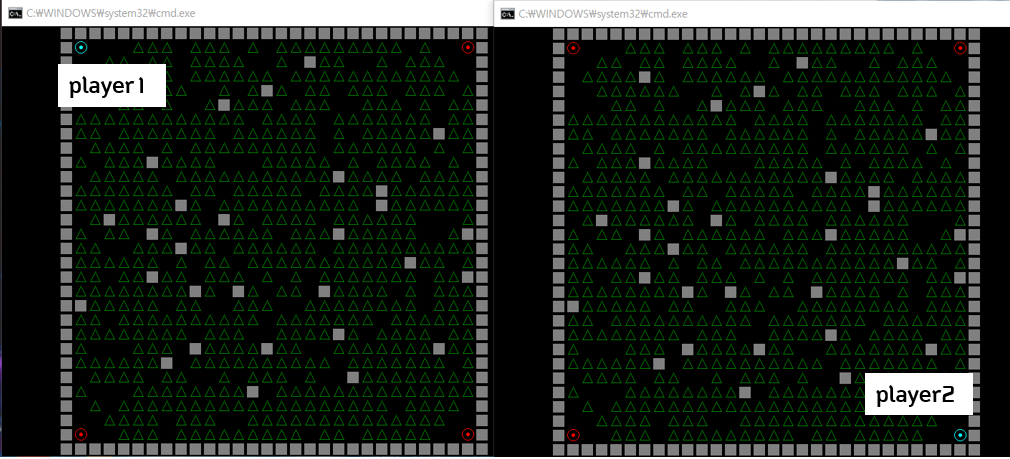
<2> 게임 첫 화면 – 게임 시작

1. void draw\_map()을 이용하여 맵을 출력한다.

2. probability() 함수를 이용하여 맵을 랜덤 설정한다.

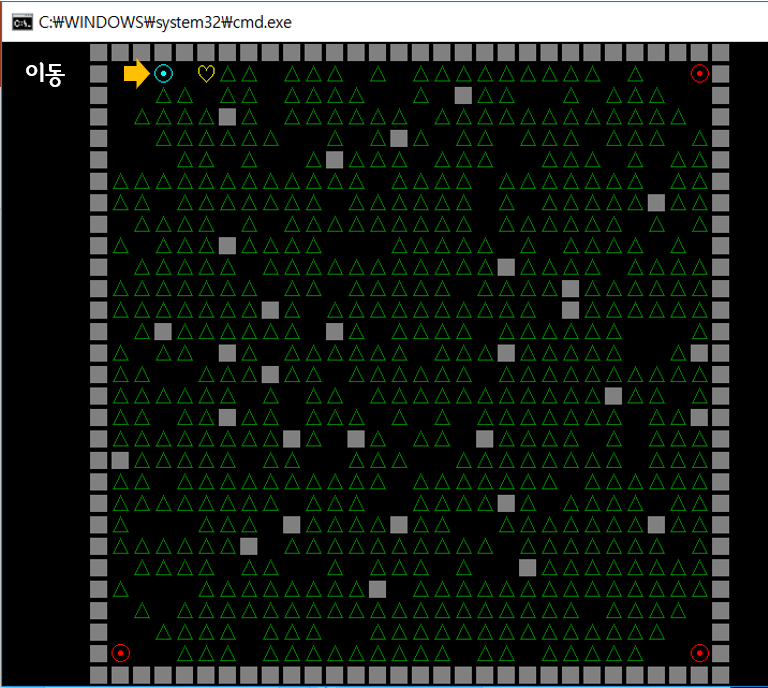
3. draw\_interface() 함수를 이용하여 플레이어의 상태를 표현한다.

<3> player 등장



1. void draw\_character()를 이용하여 player들을 출력한다.

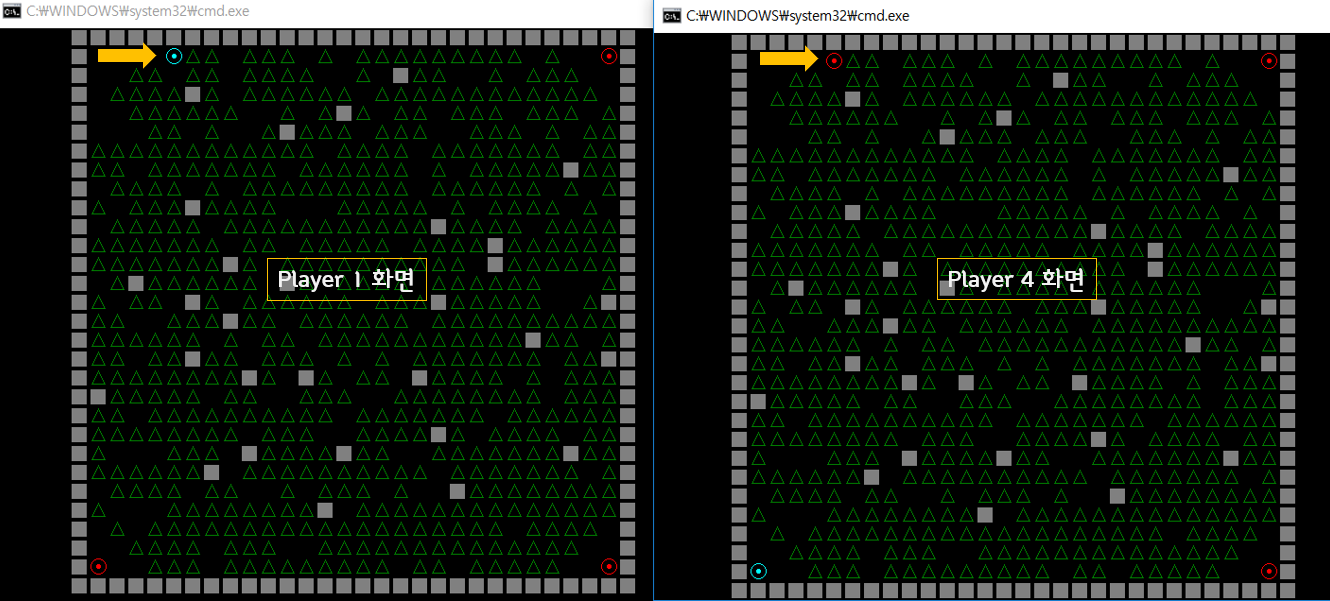
2. 적을 그리는 부분과 자신의 캐릭터를 그리는 부분을 나누었다.

<4-1> 이동 

1. input\_key()에서 키입력이 들어오면

2. move(int pos, int dir)에서 이동할 위치의 맵 정보를 알아오고

3. trans\_draw(int num, int dir)에서 변화된 곳을 그린다.



<4-2> 다른 클라이언트에서의 이동 화면

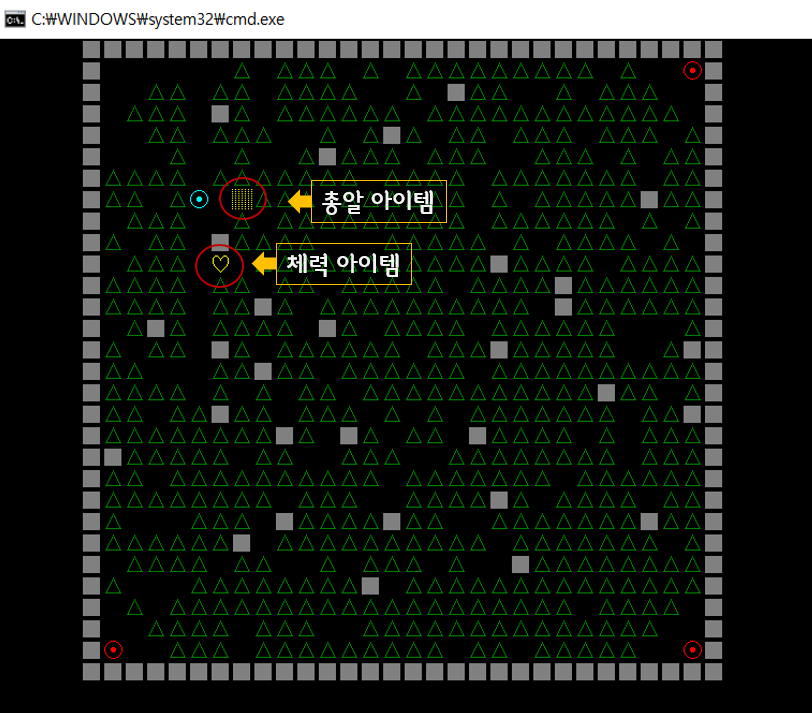
<5> 공격 – 총알 이동



1. gun\_move() 으로 총알 이동을 출력한다. 방향키에 따라 방향을 달리한다.

2. trans\_draw\_interface(int num)에서 인터페이스 수정 내용을 출력한다.

<6> 아이템



1. 총알이 나무에 닿았을 때 probability()함수에서 아이템이 나올지 확정 짓고.

2. sendData[1]에 1번 값을 넣고 클라이언트에게 정보를 전한다.

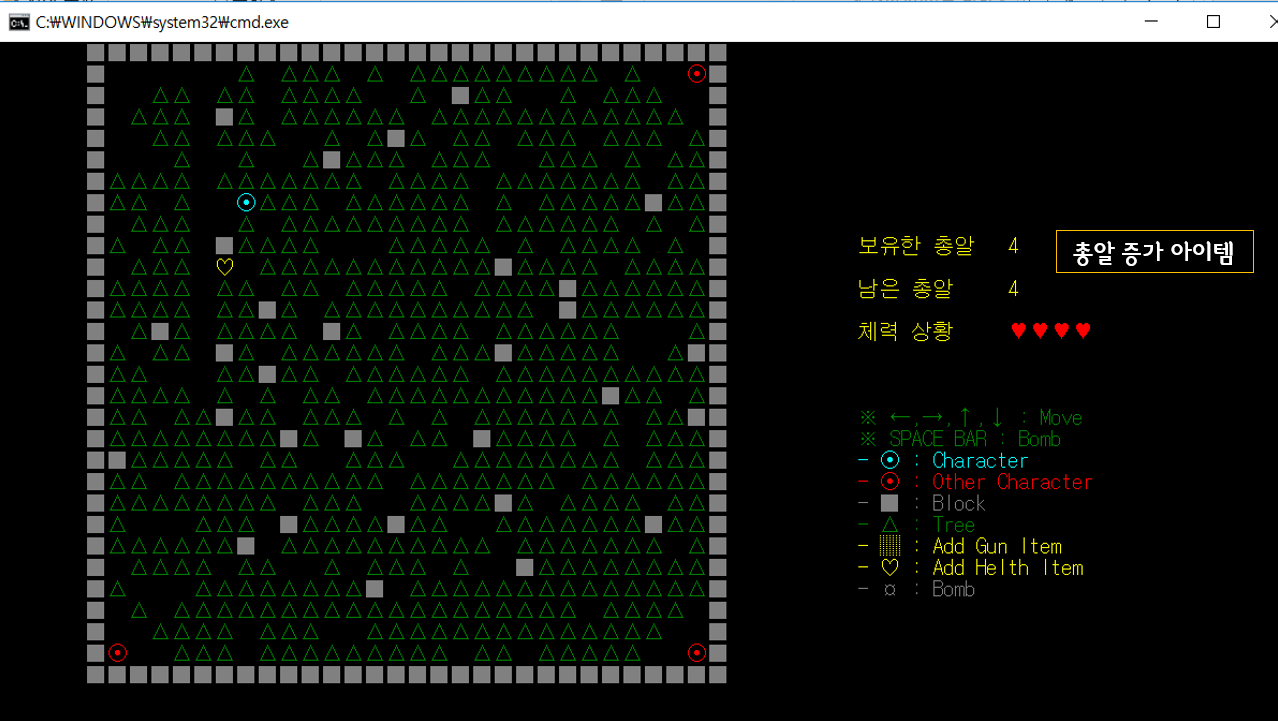
3. trans\_draw(int num, int dir)에서 아이템이 나오면 출력한다.

<7-1> 아이템 획득 – 체력

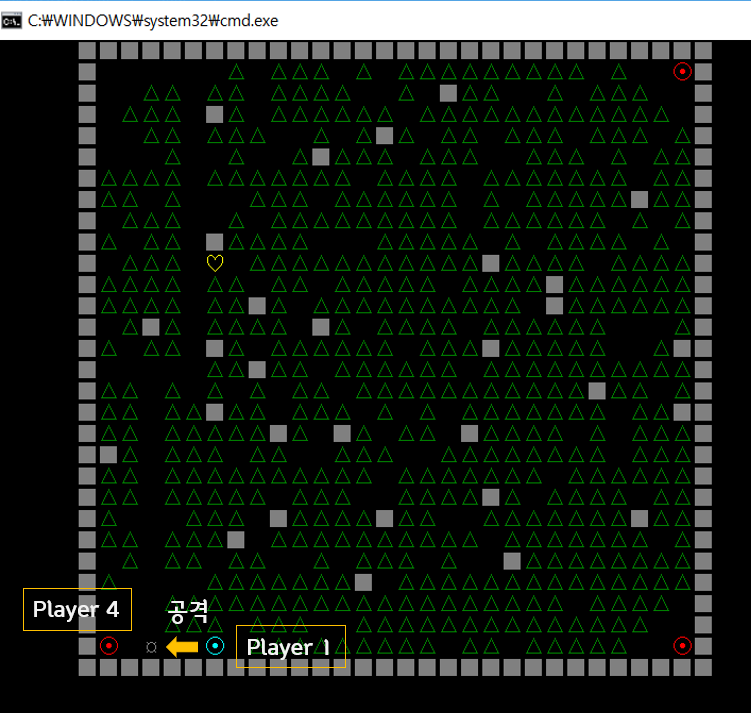


1. trans\_draw\_interface(int num)에서 helth\_nowM 상태면 체력을 깎고 helth 상태면 체력을 추가 하였다. (체력은 최대 5개로 한정하였다.)

<7-2> 아이템 획득 – 총알



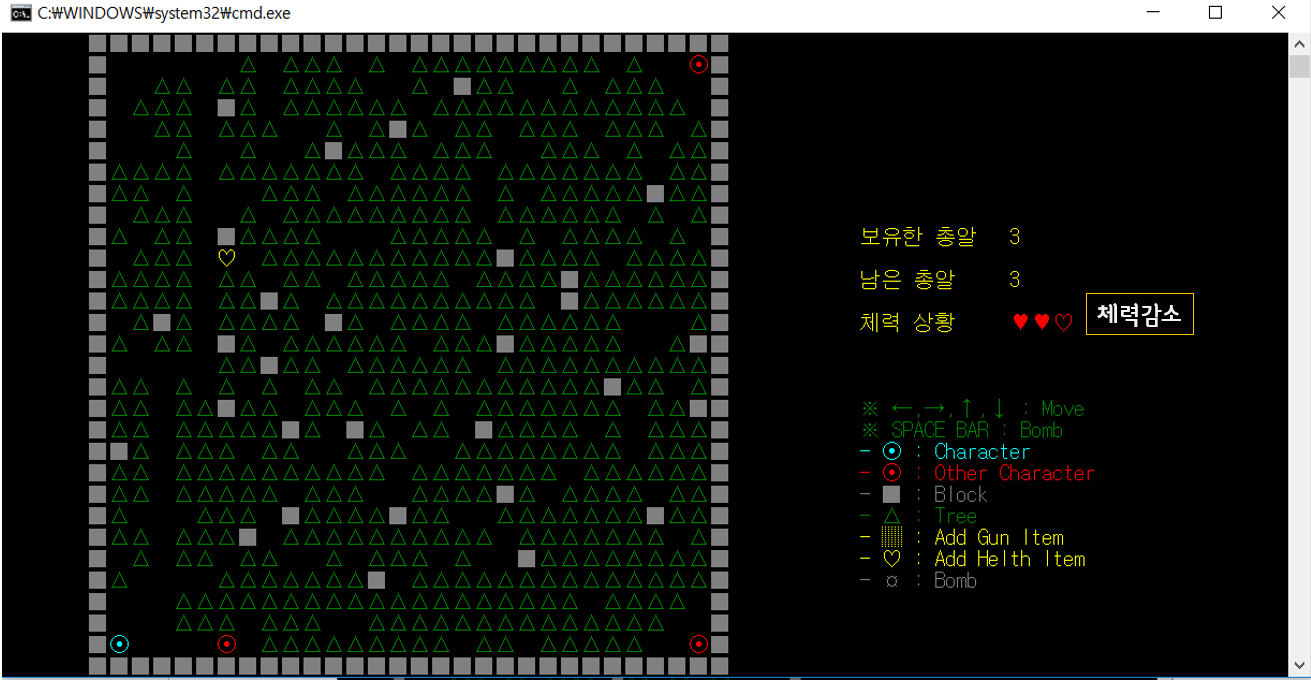
1. trans\_draw\_interface(int num)에서 총알 아이템을 먹으면 총알의 최대 수 와 보유한 총알의 수를 증가하게 하였다.

<8> 공격 

1.gun\_collision(int i, int pos) 총알 충돌 확인 함수에서 캐릭터와 총알이

충돌을 하면 send\_msg(10)를 호출 한다.

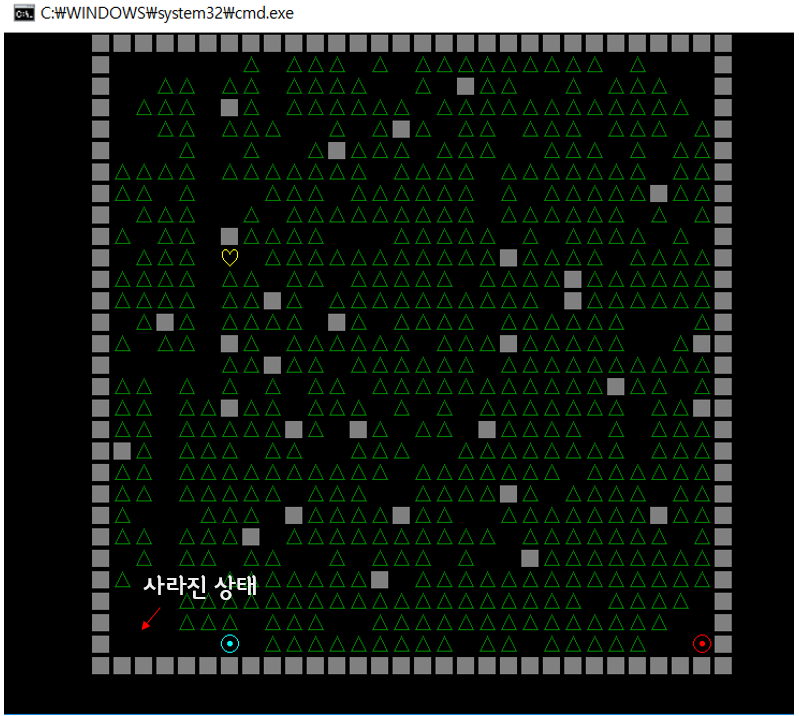
<9> 체력 감소



1. helth\_test()통해 체력 검사를 한다.

2. helth 변수가 0보다 작거나 같아진다면 사망 상태로 분류한다.

<10> 사망 상태 – 상대방을 죽임.



1. user\_die(int num)에서 클라이언트 사망 상태를 알 수 있다.

<11> 사망 – 자신의 캐릭터 사망



1. void helth\_test()에서 helth 변수가 0보다 작거나 같아진다면 사망 상태로 분류 한다. GAME OVER를 출력한다.

<12> 승리



1. 유저가 승리한다면 void user\_victory() 함수에서 VICTORY를 인터페이스에 호출 한다.