

2014
PORTFOLIO
Game Project

Java. TCP. Swing.
버블버블 게임 프로젝트

CONTENTS 목차

A. 기획안

B. 미리 보기

C. 플로 차트

D. 로직 분석



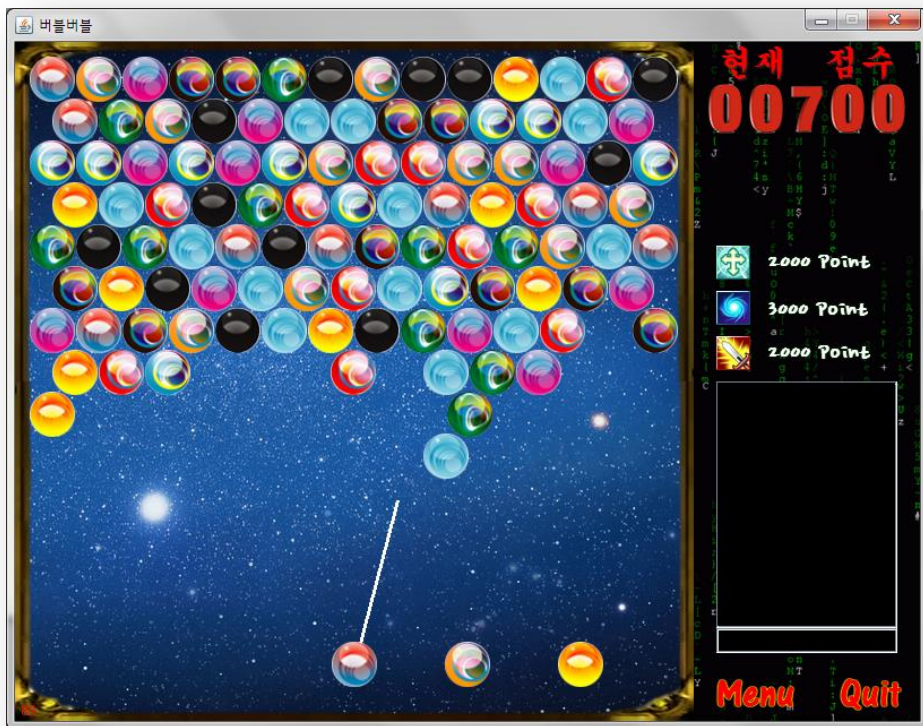
프로젝트 내용	
프로젝트 명	버블버블(구슬쏘기) 게임
프로젝트 성격	Java & TCP & Swing 기반의 네트워크 게임
기획 의도	버블버블 게임과 네트워크를 통해 온라인 플레이가 가능한 게임
개발 기간	2014.01.08 ~ 2014.01.31
주요 기술	JAVA / TCP(ServerSocket) / AWT(Swing)
프로젝트 개요	버블버블(구슬쏘기) 1:1대전 및 채팅이 가능한 네트워크 게임
팀구성 인원	1명
본인 역할	게임 구상, 디자인(이미지), 로직구현 등 전부 혼자 개발

기획안

미리보기

플로차트

로직분석



게임 메인 화면



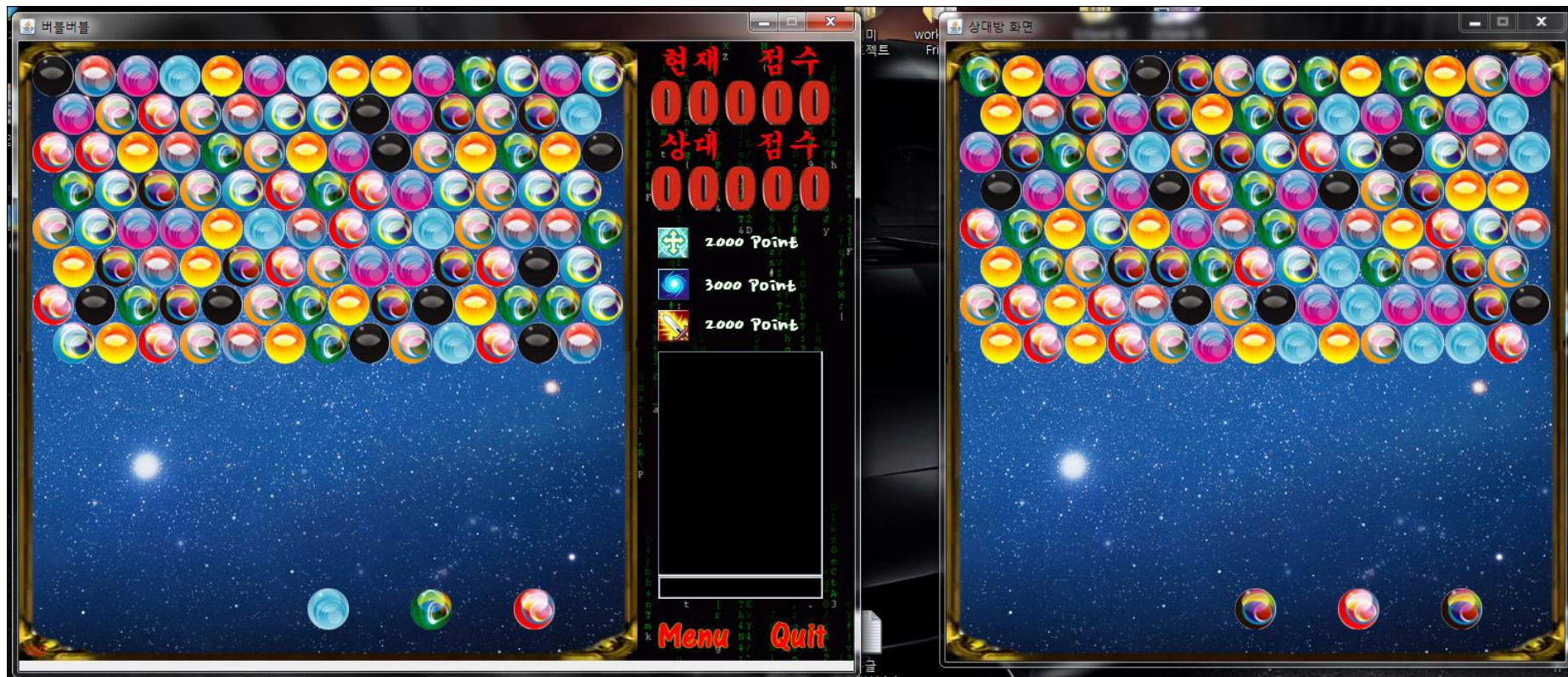
네트워크 접속 화면

기획안

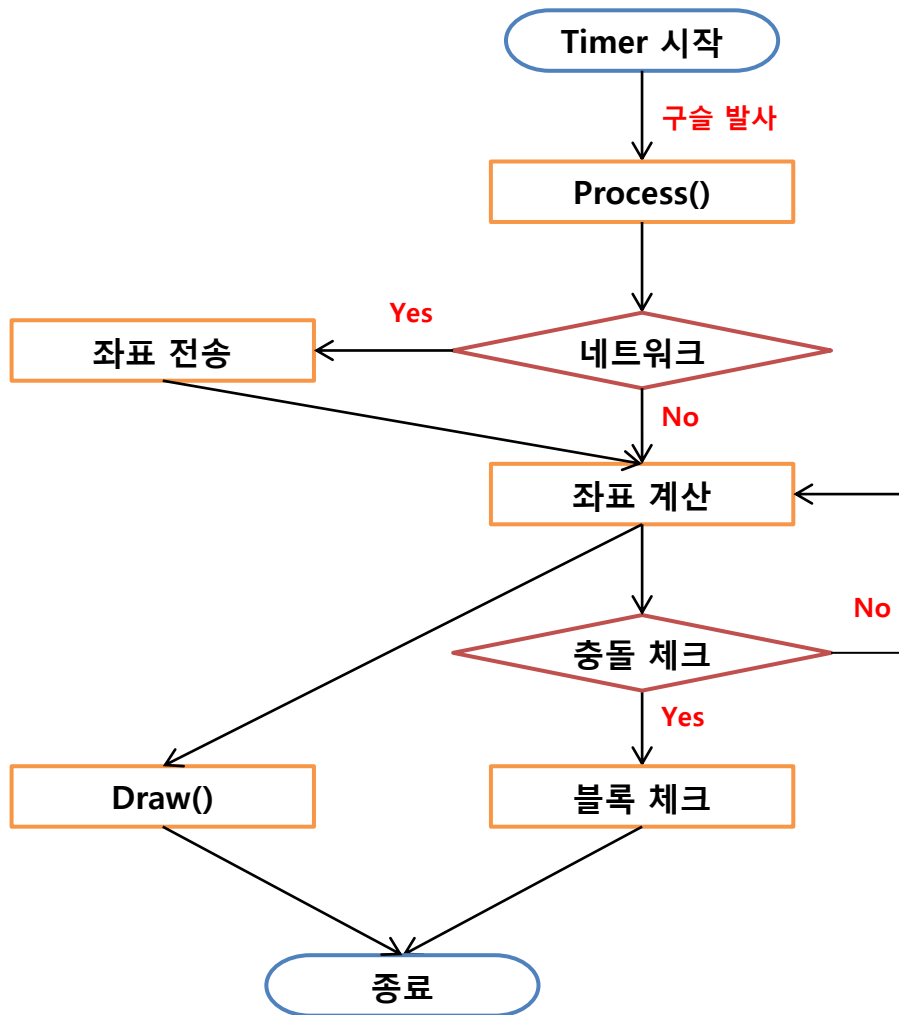
미리보기

플로차트

로직분석



네트워크 접속 화면



- ❖ Timer : 매 초당 200회 구동.
- ❖ Process() : 계산 메소드 시발점.
- ❖ 네트워크 : boolean 값을 이용하여
네트워크 연결 체크.
- ❖ 좌표 전송 : 구슬 발사좌표 전송.
- ❖ 좌표 계산 : 구슬 발사좌표를 기준으로
이동좌표를 계산.
- ❖ 충돌 체크 : 구슬과 블록의 충돌 체크.
- ❖ 블록 체크 : 블록이 3칸 이상 겹쳤는지 체크.
- ❖ Draw() : JPanel에 그림 그리기 메소드.



※ 볼과 블록의 구성

1. 볼의 구성

볼은 3개의 객체로 구성됩니다. 첫 번째 볼 객체는 앞으로 발사될 볼의 객체이며 나머지 두 개의 볼 객체는 다음에 발사될 볼의 객체입니다.

2. 블록의 구성

블록은 [13][13] 객체 배열로 만들어졌습니다. 각 블록의 객체 안에는 draw_block 이라는 boolean 값이 존재하며 true 일 경우 화면상에 그려지게 되며 블록이 존재한다는 뜻입니다. 충돌 체크 로직에서는 블록의 draw_block 값을 체크 후 볼과 블록이 닿았는지 확인하게 됩니다.

//볼의 생성

```
Bubble_Ball class_ball[]=new Bubble_Ball[3];
class_ball[0]=new Bubble_Ball((W-200)/2-ball_round/2, H-50-ball_round/2, ball_round);
class_ball[1]=new Bubble_Ball((W-200)/2-ball_round/2+100, H-50-ball_round/2, ball_round);
class_ball[2]=new Bubble_Ball((W-200)/2-ball_round/2+200, H-50-ball_round/2, ball_round);
```

//블록의 생성

```
Bubble_Block [][]class_block=new Bubble_Block[14][14];
for (int i = 0; i < 14; i++) {
    for (int j = 0; j < 14; j++) {
        if (i<game_level) {
            if (i%2==0) {//i=짝수
                class_block[i][j]=new Bubble_Block (41*j+13, 37*i+13, true, ran.nextInt(10)+1);
            }else {//i=홀수
                class_block[i][j]=new Bubble_Block (41*j+33, 37*i+13, true, ran.nextInt(10)+1);
            }
            if (j==13) class_block[i][j].draw_block=false;
        }
        else {
            if (i%2==0) {//i=짝수
                class_block[i][j]=new Bubble_Block (41*j+13, 37*i+13, false, ran.nextInt(10)+1);
            }else {//i=홀수
                class_block[i][j]=new Bubble_Block (41*j+33, 37*i+13, false, ran.nextInt(10)+1);
            }
            if (j==13) class_block[i][j].draw_block=false;
        }
    }
}
```




※ 충돌 체크 로직 - 1

볼의 위치 찾기

현재 화면에는 13x13의 블록이 존재합니다. 눈에 보이지 않는 블록도 사실은 화면 뒤에 존재하기 때문에 현재 볼의 위치를 찾는 로직이 필요합니다. 만약 볼이 [5][9] 블록의 위치에 존재한다면 [5][9] 블록의 전후좌우 블록과 볼이 겹쳤는지 체크하게 되는 로직이 시작됩니다. 이 로직은 다음 장에서 설명해 드리도록 하겠습니다.

(충돌 로직은 위치 찾기 for문 2개와 충돌체크 for문 2개가 함께 이루어진 4중 for문으로 구성됩니다. 하지만 for문 2개를 붙여놓았을 뿐, 절대 4중 for문은 아닙니다.)

```
for (int i = 13; i >= 0; i--) {  
    //볼의 좌표를 기준으로 위아래까지 합쳐 총 3개의 블록만 해당된다.  
    if (class_ball_copy.y > class_block[i][0].y-40 &&  
        class_ball_copy.y < class_block[i][0].y+45) {}  
    else continue;  
  
    for (int j = 0; j < 14; j++) {  
        //볼의 좌표를 기준으로 좌우까지 합쳐 총 3개의 블록만 해당된다.  
        if (class_ball_copy.x > class_block[i][j].x-40 &&  
            class_ball_copy.x < class_block[i][j].x+45) {}  
        else continue;  
  
        //2중for문으로 볼과 위치를 비교할 블록 7개를 찾았다.  
        //이제 아래 2중for문은 볼이 어떤 블록과 맞닿았는지 찾는 로직이다.  
        //그리고 확인하려는 블록이 꺼진 상태라면 패스한다.  
        if (!class_block[i][j].draw_block) continue;  
  
        .  
        .  
        .  
    }  
}
```




※ 충돌 체크 로직 - 2

볼과 블록의 충돌 체크

현재 어떤 블록의 위치에 볼이 있는지 찾았다면 해당 블록의 상하좌우에 있는 블록들과 볼이 닿았는지 체크합니다. 블록은 상단에 2개, 좌우에 각각 1개씩 2개, 하단에 2개로 총 6개의 블록으로 둘러싸여 있습니다. 각각의 블록에 있는 draw_block boolean값이 true라면 본격적으로 볼과 충돌체크를 시작합니다. 볼과 블록은 생성 시 원의 둘레 좌표 정보를 담은 해쉬맵이 있습니다. 이 해쉬맵을 이용한 볼과 블록의 충돌체크를 하게 되는 로직이 바로 오른편 로직입니다.

```
for (int ii = 2; ii < 40; ii+=2) {
    if (ii==20) continue; //상하좌우 꼭지점은 체크 안한다.
    bx1=class_ball_copy.x+ii;
    by1=class_ball_copy.y+20-class_ball_copy.hm.get(ii);
    bx2=class_ball_copy.x+ii;
    by2=class_ball_copy.y+20+class_ball_copy.hm.get(ii);
    for (int jj = 2; jj < 40; jj+=2) {
        if (jj==20) continue; //상하좌우 꼭지점은 체크 안한다.
        bx3=class_block[i][jj].x+jj;
        by3=class_block[i][jj].y+20-class_ball_copy.hm.get(jj);
        bx4=class_block[i][jj].x+jj;
        by4=class_block[i][jj].y+20+class_ball_copy.hm.get(jj);
        if (bx1>=bx4-2 && bx1<=bx4+2 && by1>=class_block[i][jj].y+20 && by1<=by4) {
            row=i; column=j;
            if (jj<20) {
                tic_location=1; shot_ball=-1; break exit_ball_tic;
            }
            else {
                tic_location=2; shot_ball=-1; break exit_ball_tic;
            }
        }
        if (bx2>=bx3-2 && bx2<=bx3+2 && by2<=class_block[i][jj].y+20 && by2>=by3) {
            row=i; column=j;
            if (jj<20) {
                tic_location=3; shot_ball=-1; break exit_ball_tic;
            }
            else {
                tic_location=4; shot_ball=-1; break exit_ball_tic;
            }
        }
    }
}
```



※ 볼 좌표 계산 로직

볼 좌표 계산

발사될 볼의 위치(마우스 클릭위치)와 볼의 최초 위치의 x,y 값의 차이를 비율로 환산합니다. (1:0.xxxxxx) double 소수점 6째자리까지 만들어 move_X, move_Y라는 변수에 담습니다. 연산 시 move_XX, move_YY 변수에 이 값을 더하고 수치가 1이 넘으면 화면에서 1픽셀을 이동시킵니다.

사인 코사인을 이용한 삼각함수 빗변공식을 이용하여 로직을 작성하게 되면 볼의 움직임이 살짝 떨리는 현상이 발생하여 비율을 이용한 로직 구성을 하게 되었습니다.

```
move_XX+=move_X;
move_YY+=move_Y;
if (move_Right) {
    if (move_XX>=1) {class_ball[0].x+=speed_ball;move_XX-=1;}
    if (move_Down) {
        if (move_YY>=1) {class_ball[0].y+=speed_ball;move_YY-=1;}
    }
    else {
        if (move_YY>=1) {class_ball[0].y-=speed_ball;move_YY-=1;}
    }
    if (class_ball[0].x>=W-class_ball[0].ball_round-213) {
        move_Right=false;
    }
}
else {
    if (move_XX>=1) {class_ball[0].x-=speed_ball;move_XX-=1;}
    if (move_Down) {
        if (move_YY>=1) {class_ball[0].y+=speed_ball;move_YY-=1;}
    }
    else {
        if (move_YY>=1) {class_ball[0].y-=speed_ball;move_YY-=1;}
    }
    if (class_ball[0].x<=13) {
        move_Right=true;
    }
}
```



※ 블록체크

블록체크 로직

블은 블록과 충돌 시 충돌위치에 존재하던 보이지 않는 블록의 위치에 고정하게 됩니다. 이 후 새로운 블록과 상하좌우 6개의 블록이 같은 모양인지 체크하는 로직입니다. 먼저 조건문을 통해 체크하고자 하는 블록의 draw_block 값이 true인지 체크합니다. false라면 블록이 없으므로 같은 모양인지 체크할 이유가 없기 때문입니다. 이후 블록의 block_num 값을 비교 후 값이 같다면 재귀함수를 통해 다시 한 번 더 체크하게 됩니다. 이렇게 체크한 값이 3개 이상일 경우 블록은 삭제됩니다. 또한, 13x13 배열의 블록은 짝수 행과 홀수 행의 위치가 다릅니다. 따라서 오른쪽의 6개의 조건문은 짝수 행만 체크할 수 있고 홀수 행 체크 로직은 이하 생략된 부분에 또 존재합니다.

```
for (int f=0; f<list_check_copy.size(); f++) { //블럭 체크 시작
    int n=list_check_copy.get(f).i;
    int m=list_check_copy.get(f).j;
    if (class_block[n][m].check_block==false) { //블럭의 check_block이 false면 체크를 수행
        한다.
        class_block[n][m].check_block=true; count_block++;
        if (n%2==0) { //짝수행
            if (n-1>=0 && m-1>=0 && class_block[n-1][m-1].draw_block && !class_block[n-1][m-1].check_block &&
                class_block[n][m].block_num==class_block[n-1][m-1].block_num) {
                list_check.add(class_check=new Bubble_Check(n-1, m-1)); //블럭 좌측상단
            }
            if (n-1>=0 && class_block[n-1][m].draw_block &&
                !class_block[n-1][m].check_block &&
                class_block[n][m].block_num==class_block[n-1][m].block_num) {
                list_check.add(class_check=new Bubble_Check(n-1, m)); //블럭 우측상단
            }
            if (m-1>=0 && class_block[n][m-1].draw_block &&
                !class_block[n][m-1].check_block &&
                class_block[n][m].block_num==class_block[n][m-1].block_num) {
                list_check.add(class_check=new Bubble_Check(n, m-1)); //블럭 좌측
            }
            if (m+1<=13 && class_block[n][m+1].draw_block &&
                !class_block[n][m+1].check_block &&
                class_block[n][m].block_num==class_block[n][m+1].block_num) {
                list_check.add(class_check=new Bubble_Check(n, m+1)); //블럭 우측
            }
            if (n+1<=13 && m-1>=0 && class_block[n+1][m-1].draw_block &&
                !class_block[n+1][m-1].check_block &&
                class_block[n][m].block_num==class_block[n+1][m-1].block_num) {
                list_check.add(class_check=new Bubble_Check(n+1, m-1)); //블럭 좌측하단
            }
            if (n+1<=13 && class_block[n+1][m].draw_block &&
                !class_block[n+1][m].check_block &&
                class_block[n][m].block_num==class_block[n+1][m].block_num) {
                list_check.add(class_check=new Bubble_Check(n+1, m)); //블럭 우측하단
            }
        }
    }
}
```

//이하 생략



※ 네트워크 TCP - 1

서버소켓 데이터 받기

서버소켓을 통해 데이터를 받습니다. 최초 상대방과 연결되면 다른 IP의 접속은 무시하게 됩니다. msgs[0] 조건문을 통해 로직이 각각 다르게 구성됩니다. (현재 삭제해둔 상태)

Connect : 접속시도, 수락/거절 가능

Accept : Connect 수락 시 상대방에게 보내는 메시지

Disconnect : 상대방이 거절했을 때 받는 메시지

class_ball : 볼의 발사 좌표값을 받는 메시지

game_over : 상대방이 게임오버 되면 받는 메시지

Message : 채팅 시 주고받는 메시지

```
ServerSocket server;
Socket sock;
InputStream is;
InetAddress inet;

try {
    server=new ServerSocket(port);
    while (true) {
        sock=server.accept();
        is=sock.getInputStream();
        byte buff[]=new byte [1024];
        int len=is.read(buff);
        inet=sock.getInetAddress();
        if (ip==null || ip.isEmpty() || ip.equals("localhost")) {
            ip=inetAddress.getHostAddress();
        }
        for (String S: list_blacklist) {
            if (S.equals(ip)) {ip=null;}
        }
        if (len==-1) {
            class_multi.dispose();
            frame_main.setLocation(MW/2-W/2, MH/2-H/2);
            ip=null; name_client=null;
            network_onoff=false;
            JOptionPane.showMessageDialog(null, "상대방과 네트워크 연결이 끊어졌습니다.", "
네트워크 종료 알림", JOptionPane.ERROR_MESSAGE);
        }
        String msg=new String(buff, 0, len);
        String msgs[]=msg.split("/");
        if (ip.equals(inetAddress.getHostAddress())) {
            if (msgs[0].equals("Connect")) {
            else if (msgs[0].equals("Accept")) {
            else if (msgs[0].equals("Disconnect")) {
            else if (msgs[0].equals("class_ball")) {
            else if (msgs[0].equals("game_over")) {
            else if (msgs[0].equals("Message")) {
            }
        }
    } catch (Exception e) {
    }
```



※ 네트워크 TCP - 2

소켓 데이터 보내기

상대방과 연결되었을 때 게임 진행 데이터를 보내는 로직입니다. 발사 좌표를 상대방에게 보내며 만약에 아이템을 사용하였을 경우에는 아이템 정보도 함께 보내게 됩니다. 기타 다른 전송 로직도 동일한 방법으로 전송됩니다.

```
Socket sock;
OutputStream os;

try {

    sock=new Socket(ip, port);
    os=sock.getOutputStream();

    //내가 아이템(상대블럭생성)을 사용했다면 메시지 끝에 /item3를 추가해서 보낸다.
    if (network_item3) {
        //msg=class_ball / 마우스X / 마우스Y / 볼넘버0 / 볼넘버1 / 볼넘버2 / 게임포인트 /
        item3
        String msg=String.format("class_ball/%d/%d/%d/%d/%d/%d/item3", class_ball_line.x2,
        class_ball_line.y2,
        class_ball[0].ball_num,class_ball[1].ball_num,class_ball[2].ball_num,point_game);
        os.write(msg.getBytes());

        //다시 false 해준다.
        network_item3=false;
    }else {
        //평상시에는 아래 메시지가 전송된다.
        //msg=class_ball / 마우스X / 마우스Y / 볼넘버0 / 볼넘버1 / 볼넘버2 / 게임포인트 /
        noitem
        String msg=String.format("class_ball/%d/%d/%d/%d/%d/%d/noitem", class_ball_line.x2,
        class_ball_line.y2,
        class_ball[0].ball_num,class_ball[1].ball_num,class_ball[2].ball_num,point_game);
        os.write(msg.getBytes());
    }

} catch (Exception e) {
}
```

Thank you for watching.

YoungJoon Kim