
우주탈출
Space Escape
정은지 외 3명

게임소개

우주탈출에 관한 게임으로
우주선을 작동할 수 있는 연료아이템과
우주인의 생명을 연장시켜주는
산소 아이템을 모아
우주를 탈출하여 지구로 돌아가는
스펙터클한 SF 생존게임이다.

게임규칙

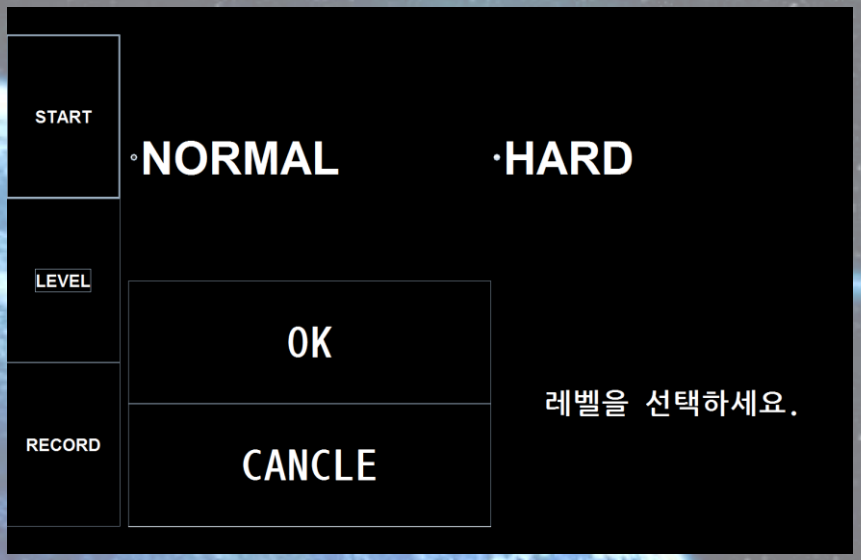
기본규칙

1. 우주의 흩뿌려진 아이템들을 먹는다.
2. 아이템을 일정 개수 먹을 때마다 우주선에 점차 불이 켜진다.
3. 우주선에 불이 다 들어오면 우주선에 돌아오라는 신호가 뜬다.
4. 준비완료가 된 우주선으로 돌아가면 지구로 돌아갈 수 있다.

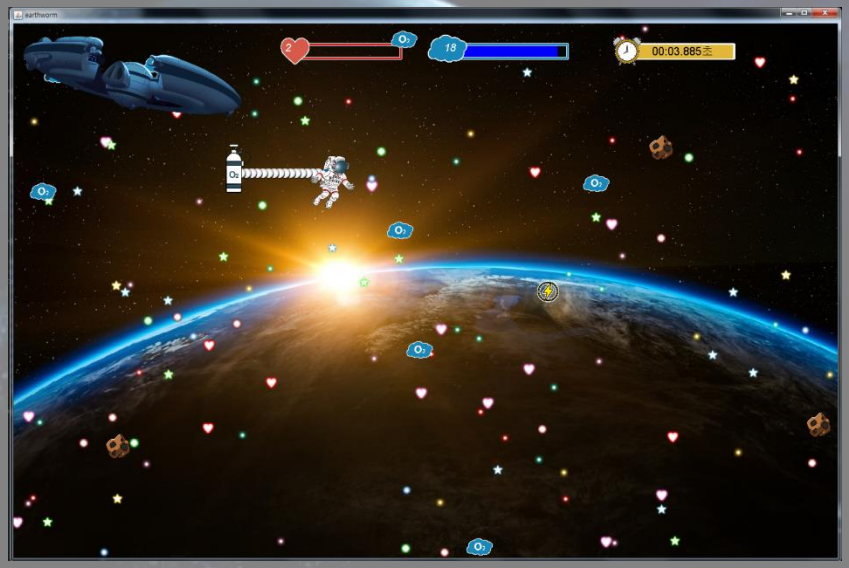
그 외 규칙

1. 별,하트(연료) 아이템을 먹으면 하트지수가 증가한다.
2. O2를 먹으면 산소통의 크기가 커지고, o2 막대 그래프가 채워진다.
3. 번개를 먹으면 스피드가 증가한다.
4. 운석과 충돌하면 2초 동안 게임이 멈춘다.
5. 스페이스를 누르면 일시 정지 된다.

메인화면

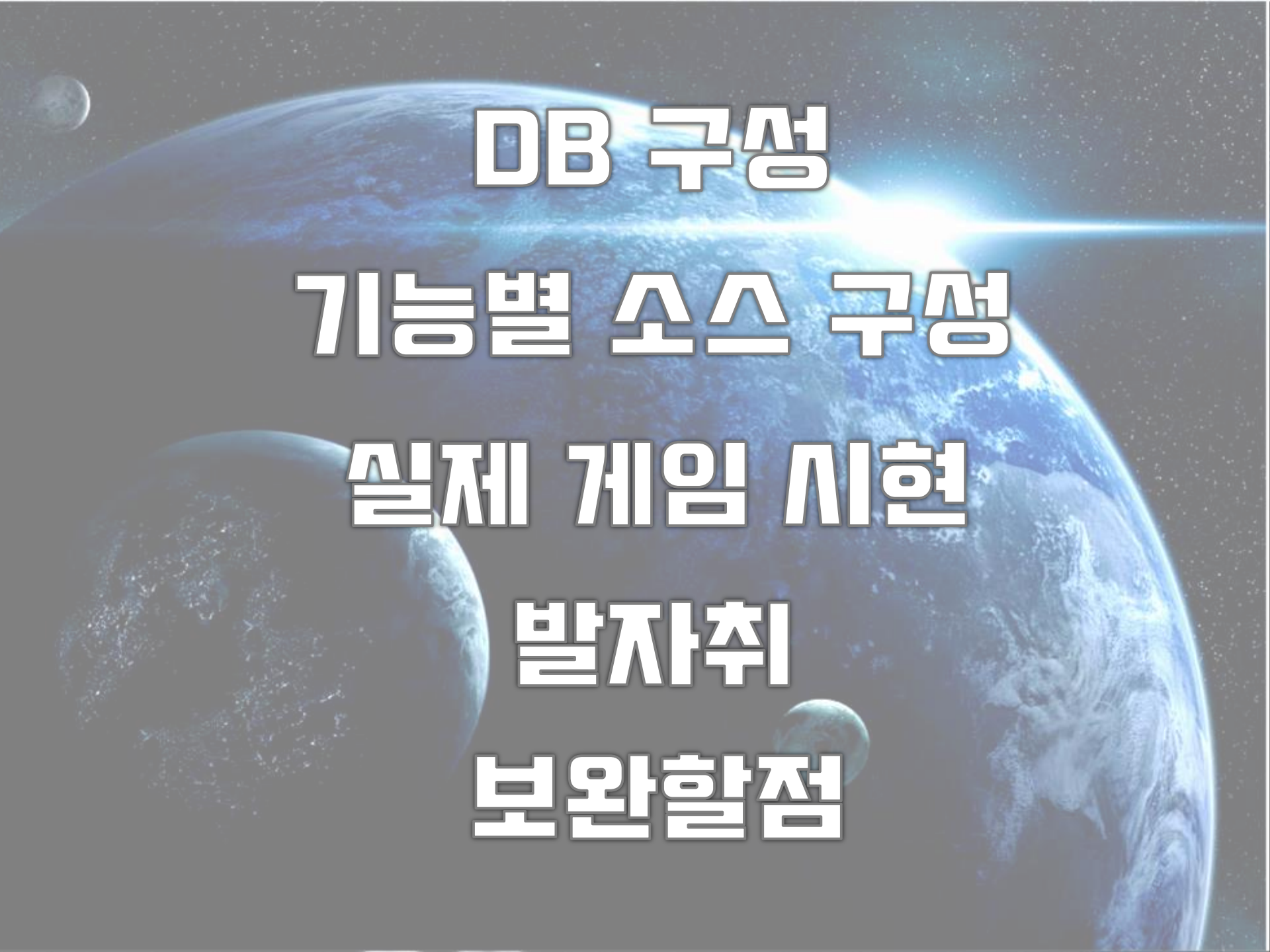


시작화면



게임화면

랭킹화면


DB 구성
기능별 소스 구성
실제 게임 시현
발자취
보완할점


```java
@Override
public void actionPerformed(ActionEvent e) {

    switch(e.getActionCommand()) {
    case "START" :
        mainPanel.removeAll();
        Thread thread = new Thread(MainFrame.this);
        thread.start();
        break;
```

Main화면에서 START를 누르면
MainFrame의 Thread가 작동하게 됩니다.

```java
@Override
public void run() {

    MainFrame.this.dispose();
    cardChange.main(null);
    try {
        Thread.sleep(3500);
    } catch (InterruptedException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    Start11.main(null);
}
```

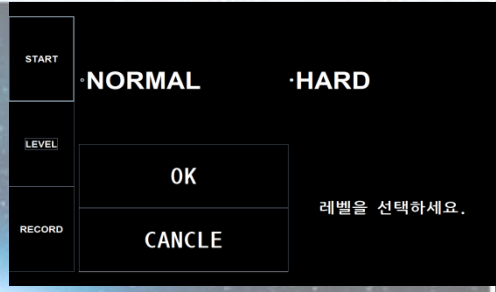
MainFrame의 run 메소드

MainFrame의 run 메소드를 살펴봅니다.
MainFrame을 종료시키고,
cardChange 메소드를 실행한 뒤,
게임화면이 시작되는 Start11 메소드를 실행한다.

```java
@Override
public void run() {
        for(int i = 0; i<label.length; i++) {
        count++;
        card.show(this, cardName[i]);

        try {Thread.sleep(1000);} catch (InterruptedException e1) {e1.printStackTrace();}

        }
        cardChange.this.dispose();
}
```

CardChange의 run 메소드

게임의 난이도 설정



게임의 난이도를 Levelflag로 기준을 잡는다.
MainFrame의 itemStateChanged 메소드에서
flag가 true일 때 NORMAL,
아닐 때 HARD로 인식한 뒤 시작한다.

```
if(e.getActionCommand().equals("OK")) {
    levelflag = flag;
    mainPanelView();

}else if(e.getActionCommand().equals("CANCLE")) {
    mainPanelView();
}else if(e.getActionCommand().equals("OUT")) {
    mainPanelView();
}
```

```
@Override
public void itemStateChanged(ItemEvent e) {
    Object object = e.getSource();
    JRadioButton item = (JRadioButton) object;

    if(item.getText().equals("NORMAL")) {
        flag = true;
    }else {
        flag = false;
    }

}
```

난이도에 따라 달라지는 것에는
산소의 감소 속도, 운석의 크기 등이 있습니다.

```
if(MainFrame.levelflag) {
    System.out.println(shift);


    if(sanso) {
            if(shift%30==5 && size>2) {
                size--;
                sanso = false;
            }
            if(size==2) {
                result=false;
            }
    }
}else {
    if(sanso) {
        if(shift%15==0 && size>2) {
            size--;
            sanso = false;
        }
        if(size==2) {
            result=false;
        }
    }

}
```

```
if(MainFrame.levelflag) {
        g.drawImage(Boob, boobx[i], booby[i], 50, 50, this);
        if (boobx[i] >= xpos[0] - mainR / 2 && boobx[i] <= xpos[0] + mainR / 2
                && booby[i] >= ypos[0] - mainR / 2 && booby[i] <= ypos[0] + mainR / 2) {
            flag = true;
            size--;
            countNum_ -= 5;
            countNum = String.valueOf(countNum_);
            boobx[i] += (random1.nextInt(1000) - 500);
            booby[i] += (random1.nextInt(1000) - 500);
        }
    }
}else { //난이도 하드
    for (int i = 0; i < 25; i++) {
        g.drawImage(Boob, boobx[i], booby[i], 100, 100, this);
        if (boobx[i] >= xpos[0] - mainR / 2 && boobx[i] <= xpos[0] + mainR / 2
                && booby[i] >= ypos[0] - mainR / 2 && booby[i] <= ypos[0] + mainR / 2) {
            flag = true;
            size--;
            countNum_ -= 5;
            countNum = String.valueOf(countNum_);
            boobx[i] += (random1.nextInt(1000) - 500);
            booby[i] += (random1.nextInt(1000) - 500);
        }
    }
```

게임종료


```java
public void actionPerformed(ActionEvent e) {
    boolean yN = false;
    switch (e.getActionCommand()) {

    case "기록보기":
        yN = true;
        System.out.println("클릭");
        MainFrame.mainPanel.removeAll();
        Record Record = new Record();
        window.add(MainFrame.mainPanel);
        window.setVisible(true);
        break;
```

```java
case "재시작":
    window.remove(MainFrame.mainPanel);
    countNum_ = 0;
    result = true;
    stop = true;
    move = false;
    space = true;
    flag = false;
    panel.remove(button);
    panel.remove(button1);
    panel.remove(button2);
    panel.remove(button3);
    panel.repaint();
    window.remove(panel);
    window.repaint();
    window.dispose();
    start55.main(null);
    break;
case "종료":
    System.exit(0);
    break;
```

```java
case "메인화면으로":
    countNum_ = 0;
    result = true;
    stop = true;
    move = false;
    space = true;
    flag = false;

    window.remove(MainFrame.mainPanel);

    panel.remove(button);
    panel.remove(button1);
    panel.remove(button2);
    panel.remove(button3);
    panel.repaint();
    window.remove(panel);
    window.repaint();
    window.dispose();
    MainFrame main = new MainFrame();
    break;
```

기록을 입력하려면 100개 이상 달성하였을 때만 입력할 수 있다.

```
Record.view();
if(countNum_ >= 100) {
    boolean flag2 = false;
        for (int i = 0; i < 5; i++) {
            if (Record.rank[i] < countNum_ * 100 + (1000000000 - ((end_time - start_time) * 100))) {
                flag2 = true;
            }
        }
        if (flag2) {
            hrname = JOptionPane.showInputDialog(null, "이름을 입력하세요", "기록달성", JOptionPane.DEFAULT_OPTION);
            record = String.valueOf(countNum_ * 100 + (1000000000 - ((end_time - start_time) * 100)));
            number = String.valueOf(countNum_);
            time = sdf.format(end_time - start_time - 32400000);
            Record.insert(hrname, record, number, time);
        }
}
```

기록입력

```
//  점수 업데이트
public static void insert(String name, String record, String number, String time) {
    boolean flag = true;
    if(name.length() == 0) {
        JOptionPane.showMessageDialog(null, "이름이 입력되지 않았습니다");
        flag = false;
    }

    if(flag) {

        try {

            Connection conn = DBUtil.getMySQLConnection();
            String sql = "insert into javagame(name, record, number, time) values (?, ?, ?, ?)";
            PreparedStatement pstmt = conn.prepareStatement(sql);
            pstmt.setString(1, name);
            pstmt.setString(2, record);
            pstmt.setString(3, number);
            pstmt.setString(4, time);
            pstmt.executeUpdate();
            DBUtil.close(conn);
            DBUtil.close(pstmt);

        } catch (SQLException e) {
            e.printStackTrace();
        }
```

기록보기

```
//  점수 기록내역 확인
public static void view() {
    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = MemoProjectDAO.select();


    for(int i=model.getRowCount() - 1 ; i>=0 ; i--) {
        model.removeRow(i);
    }

    try {
        int cnt = 0;
        if(rs.next()) {

            String[] rowData = new String[4];
            do {

                rowData[0] = rs.getString("name");
                rowData[1] = rs.getString("record");
                rowData[2] = rs.getString("number");
                rowData[3] = rs.getString("time");
                model.addRow(rowData);
                rank[cnt] = Integer.parseInt(rowData[1]);

                cnt++;
                if(cnt==5) {
                    break;
                }
            } while(rs.next());
        }
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
```

기록보기


```
JScrollPane jsp = new JScrollPane(table);
jsp.setPreferredSize(new Dimension(1270, 600));
jsp.getViewport().setBackground(Color.BLACK);
MainFrame.mainPanel.add(jsp);
view();
```

Record.java

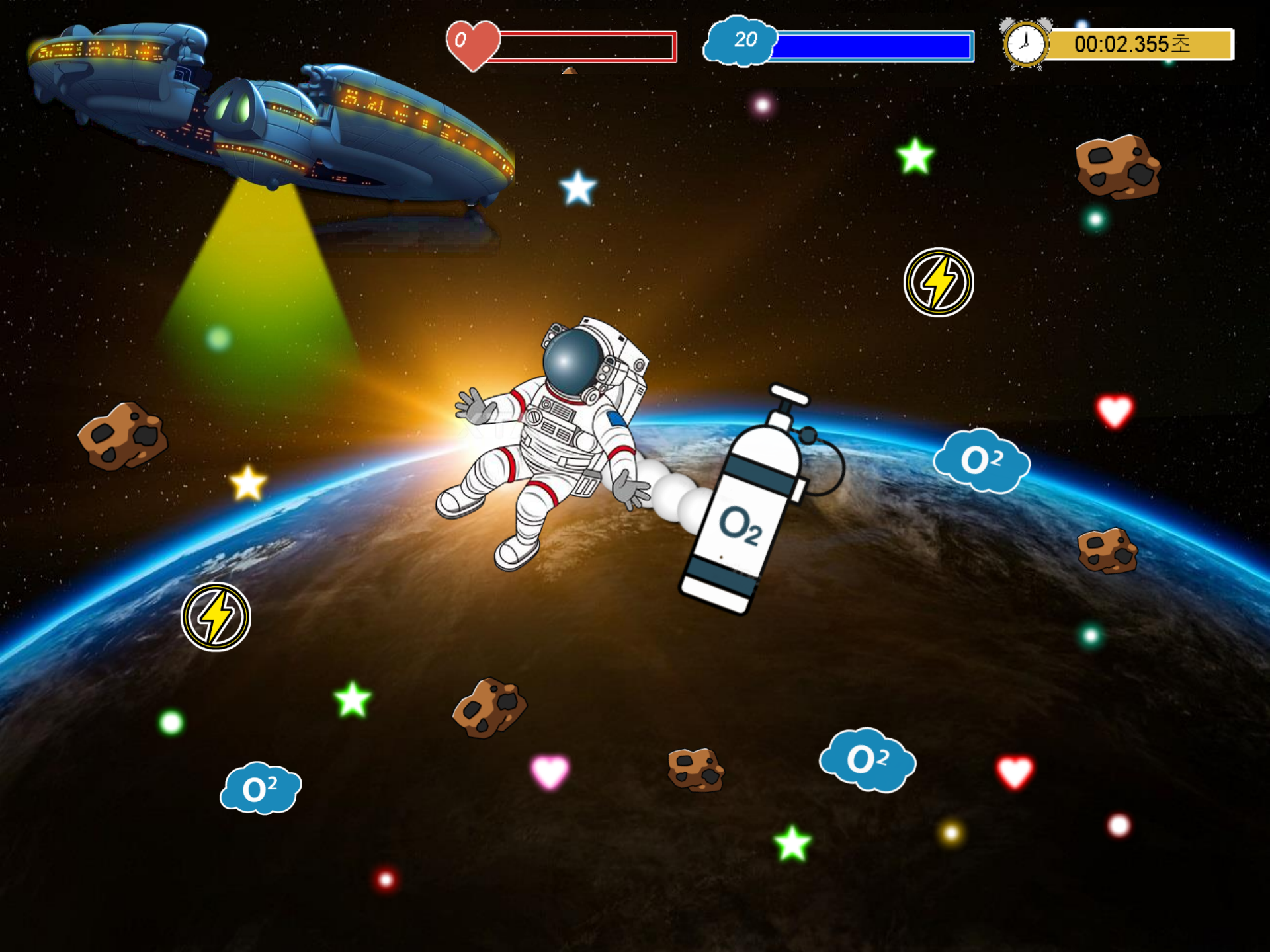
기록을 mainFrame의 mainPanel에 담고 필요할 때 불러온다.

mainFrame.java 에서 호출

```
case "RECORD" :
    데이터베이스에 연결하여 1~5등까지 기록되어진 내용을 화면에 보여준다
    확인버튼
    mainPanel.removeAll();
    mainPanel = new JPanel();
    mainPanel.setPreferredSize(new Dimension(1270, 900));
    mainPanel.setBackground(Color.BLACK);
    Record.main(null);
    mainPanel.add(outBtn, BorderLayout.SOUTH);
    add(mainPanel, BorderLayout.EAST);

    setVisible(true);
    break;
```

Start11.java에서 호출

```
public void actionPerformed(ActionEvent e) {
    boolean yN = false;
    switch (e.getActionCommand()) {

    case "기록보기":
        yN = true;
        System.out.println("클릭");
        Record history = new Record();
        window.add(MainFrame.mainPanel);
        break;
```


0
20
00:02.355초
O2
O2
O2
O2

Keypressed 메서드

방향키 뿐만 아닌 Q,W,E,Z,X,C 키를 사용.

우주인이랑 산소줄 각각을 배열에 담아 키를 누르면 xpos[0], ypos[0]만 이동하고 keypressed에서 xpos[i+1]=xpos[i]로 하나씩 따라가도록 구현. 산소 아이템을 먹으면 산소 줄이 하나 늘어남.

```
window.addKeyListener(new KeyAdapter() {

    @Override
    public void keyPressed(KeyEvent e) {

        switch (e.getKeyCode()) {

        // 일시중지,재시작소스
        case KeyEvent.VK_SPACE:
            if (space) {
                thread.suspend();
            } else if (!space) { // 재실행
                thread.resume();
            }
            space = (!space);
            shift++;
            break;

        case KeyEvent.VK_LEFT:
        case KeyEvent.VK_A:
            for (int i = size - 1; i >= 0; i--) {
                xpos[i + 1] = xpos[i] - r / 2 * (up - 1);
                ypos[i + 1] = ypos[i];
                move = true;
            }
            faceimg2 = Toolkit.getDefaultToolkit().getImage("./src/images/main.png");
            xpos[0] -= r / 2 * up;
            xspeed = -20;
            shift++;
            break;

        case KeyEvent.VK_RIGHT:
        case KeyEvent.VK_D:
            for (int i = size - 2; i >= 0; i--) {
                xpos[i + 1] = xpos[i] + r / 2 * (up - 1);
                ypos[i + 1] = ypos[i];
                move = true;
            }
            faceimg2 = Toolkit.getDefaultToolkit().getImage("./src/images/main2.png");

            xpos[0] += r / 2 * up;
            xspeed = 20;
            shift++;
            break;
```

```
        case KeyEvent.VK_Q:
            for (int i = size - 2; i >= 0; i--) {
                xpos[i + 1] = xpos[i] - r / 2 * (up - 1);
                ypos[i + 1] = ypos[i] - r / 2 * (up - 1);
                move = true;
            }
            xpos[0] -= r / 2 * up;
            ypos[0] -= r / 2 * up;
            xspeed = -20;
            yspeed = -20;
            shift++;
            break;
        case KeyEvent.VK_Z:
            for (int i = size - 2; i >= 0; i--) {
                xpos[i + 1] = xpos[i] - r / 2 * (up - 1);
                ypos[i + 1] = ypos[i] + r / 2 * (up - 1);
                move = true;
            }
            xpos[0] -= r / 2 * up;
            ypos[0] += r / 2 * up;
            xspeed = -20;
            yspeed = 20;
            shift++;
            break;
        case KeyEvent.VK_C:
            for (int i = size - 2; i >= 0; i--) {
                xpos[i + 1] = xpos[i] + r / 2 * (up - 1);
                ypos[i + 1] = ypos[i] + r / 2 * (up - 1);
                move = true;
            }
            xpos[0] += r / 2 * up;
            ypos[0] += r / 2 * up;
            xspeed = 20;
            yspeed = 20;
            shift++;
            break;
        case KeyEvent.VK_E:
            for (int i = size - 2; i >= 0; i--) {
                xpos[i + 1] = xpos[i] + r / 2 * (up - 1);
                ypos[i + 1] = ypos[i] - r / 2 * (up - 1);
                move = true;
            }
            xpos[0] += r / 2 * up;
            ypos[0] -= r / 2 * up;
            xspeed = 20;
            yspeed = -20;
            shift++;
            break;
    }
```

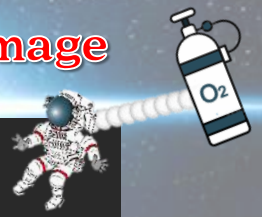
아이템의 좌표를 배열에 담는다.

```
static int boobx[] = new int[1000]; // 폭탄의 랜덤 x좌표값을 담는 배열
static int booby[] = new int[1000]; // 폭탄의 랜덤 y좌표값을 담는 배열

static int randomx[] = new int[1000]; // 별,하트 아이템의 랜덤 x좌표값을 담는 배열
static int randomy[] = new int[1000]; // 별,하트 아이템의 랜덤 y좌표값을 담는 배열
static int randomcolor[] = new int[256]; // 먹이 색깔

static int heartx[] = new int[1000]; // 산소의 랜덤 x좌표값을 담는 배열
static int hearty[] = new int[1000]; // 산소의 랜덤 y좌표값을 담는 배열

static int boosterx[] = new int[1000]; // 스피드아이템의 랜덤 x좌표값을 담는 배열
static int boostery[] = new int[1000]; // 스피드아이템의 랜덤 y좌표값을 담는 배열
```

여러 아이템들의 좌표를 랜덤으로 지정해야했다.
Paint 메소드에서 바로 random 함수를 사용하여 좌표를 설정하면 ,
그 때마다 repaint되어 깜박거림 현상이 심하였기 때문이다.
그렇기 때문에 좌표값을 배열에 담아 꺼내쓰는 방식을 택하였다.

Paint 메소드에서의 구현

메인 우주인의 몸, 산소 연결줄, 산소통을 구현하는 drawImage




```
// 메인 우주인
for (int i = 1; i < size - 1; i++) {
    g.drawImage(body, xpos[i], ypos[i], r, r, this);
}
g.drawImage(o2, xpos[size - 1] - mainR, ypos[size - 1] - mainR, mainR * 2, mainR * 2, this);
g.drawImage(faceimg2, xpos[0] - mainR / 2, ypos[0] - mainR / 2, 2 * mainR, 2 * mainR, this);
```

아이템들을 골고루 배치하기 위한 for문을 사용한 drawImage

```
// 아이템 뿌리기
for (int i = 0; i < boobx.length; i++) {
    g.drawImage(items[random1.nextInt(12)], randomx[i], randomy[i], 30, 30, this);
    if (randomx[i] >= xpos[0] - mainR / 2 && randomx[i] <= xpos[0] + mainR / 2
            && randomy[i] >= ypos[0] - mainR / 2 && randomy[i] <= ypos[0] + mainR / 2) {
        randomx[i] += (random1.nextInt(1000) - 500);
        randomy[i] += (random1.nextInt(1000) - 500);
        countNum_++; // count 증가
        countNum = String.valueOf(countNum_);
        xpos[size - 1] = xpos[size - 2] + r / 2;
        ypos[size - 1] = ypos[size - 2];
    }
}
```

먹을 때 마다 증가하는

먹은 아이템이 사라지게 하기 위해 다른 좌표로 이동하는 방법을 택함.

우주선의 단계별 불켜짐

6단계

```
//우주선 출발준비
g.drawImage(ship[0], 10, 20, 400, 150, this);
if(Integer.parseInt(countNum)>=20) {
    g.drawImage(ship[1], 10, 20, 400, 150, this);
}
if(Integer.parseInt(countNum)>=40) {
    g.drawImage(ship[2], 10, 20, 400, 150, this);
}if(Integer.parseInt(countNum)>=60) {
    g.drawImage(ship[3], 10, 20, 400, 150, this);
}if(Integer.parseInt(countNum)>=80) {
    g.drawImage(ship[4], 10, 20, 400, 150, this);
}if(Integer.parseInt(countNum)>=100) {
    g.drawImage(ship[5], 10, 20, 400, 380, this);

}
```

운석과 충돌했을 때 Flag를 사용하여 2초 정지

```
// 폭탄 먹으면 일시정지
for (int i = 0; i < 25; i++) {
    g.drawImage(Boob, boobx[i], booby[i], 50, 50, this);
    if (boobx[i] >= xpos[0] - mainR / 2 && boobx[i] <= xpos[0] + mainR / 2
            && booby[i] >= ypos[0] - mainR / 2 && booby[i] <= ypos[0] + mainR / 2) {
        flag = true;
        size--;
        countNum_ -= 5;
        countNum = String.valueOf(countNum_);
        boobx[i] += (random1.nextInt(1000) - 500);
        booby[i] += (random1.nextInt(1000) - 500);
    }
}
```

```
    if (flag) {
        g.drawImage(ohno,480,50,550,170,this);

        /// count 감소
        g.setColor(Color.red);
        g.setFont(new Font("SANS_SERIF", Font.ITALIC, 60));
        g.drawString("--5", width / 2 + 100, height / 4);
        ///
    }
    g.setColor(Color.white);
    g.setFont(new Font("SANS_SERIF", Font.PLAIN, 20));
    g.drawString(timelapse, 1200, 50);
}
```

Flag를 사용하여 운석과 충돌했을 때
flag가 true가 되면서
2초동안 일시정지가 되고. 하트 지수도 감소한다.

스피드 먹으면 SPPED UP!

```
for (int i = 0; i < 25; i++) {
    g.drawImage(booster, boosterx[i], boostery[i], 50, 50, this);
    g.drawString("speed="+String.valueOf(speed), 400, 400);
    g.drawString("size="+String.valueOf(size), 800, 400);
    if (boosterx[i] >= xpos[0] - mainR / 2 && boosterx[i] <= xpos[0] + mainR / 2
            && boostery[i] >= ypos[0] - mainR / 2 && boostery[i] <= ypos[0] + mainR / 2) {
        booster_time = System.currentTimeMillis();
        g.drawImage(speedup, 400, 400, this);
        up=3;
        speed=25;
        boosterx[i] += (random1.nextInt(1000) - 500);
        boostery[i] += (random1.nextInt(1000) - 500);
    }
    if(System.currentTimeMillis()-booster_time>=5000) {
        up=2;
        speed=15;
    }
}
```

Speed 아이템을 먹으면
Speed의 값이 25로 증가되고
5초 뒤에는 원래속도인 15로 감소된다.

O2 먹으면
산소 그래프의 증가


```
if(size>=0) {
    if(size>=20) {
        size=20;
    }
    g.setColor(Color.blue);
    g.fillRect(800, 35, size*10, 30);
}
g.drawImage(o2g, 800, 35, 200, 30, this);    //산소
g.drawImage(o2gi,745,20,75,50,this);
```

```
//산소가 점점 떨어지는 코드
if(shift%30==5 && size>2) {
    size--;
}
//산소가 떨어지면 죽는코드
if(size==2) {
    result=false;
}
```

```
case KeyEvent.VK_Q:
    for (int i = size - 2; i >= 0; i--) {
        xpos[i + 1] = xpos[i] - r / 2 * (up - 1);
        ypos[i + 1] = ypos[i] - r / 2 * (up - 1);
        move = true;
    }
    xpos[0] -= r / 2 * up;
    ypos[0] -= r / 2 * up;
    xspeed = -20;
    yspeed = -20;
    shift++;
    break;
```

기본 산소 20으로 설정하고
그 이상 증가하지 않도록 만들었다.
산소의 감소는 시간의 흐름에 따라가 아닌
keypressed 메서드에서 버튼의 동작횟수를
세는 shift라는 변수를 설정한 뒤,
나누어 떨어질 때 감소하도록 설정하였다.

O2 가 부족하면
경고창 화면 띄우기


```
if(size<=5) {
    g.setColor(new Color(255, 0, 0,100));
    g.fillRect(0, 0, width, height);
}
```

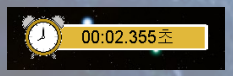
산소가 5이하가 되면
빨간 경고창을 띄우고,
6이상이 되면 경고창이 없어진다..

Run 메소드에서의 구현

```
public void run() {

    start_time = System.currentTimeMillis();

    while (result) {

        if (flag) {
            repaint();
            try {
                Thread.sleep(2000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            flag = false;
        } else {
            if (xpos[0] > xpos[1] && ypos[0] == ypos[1]) {

                for (int i = 0; i < randomx.length; i++) {
                    randomx[i] -= speed;
                }
                for (int i = 0; i < heartx.length; i++) {
                    heartx[i] -= speed;
                    boobx[i] -= speed;
                    boosterx[i]-=speed;
                }

                for (int i = 0; i < size; i++) {
                    xpos[i] -= speed;
                    bgxpos -= speed;
                }
            } else if (xpos[0] < xpos[1] && ypos[0] == ypos[1]) {
                for (int i = 0; i < randomx.length; i++) {
                    randomx[i] += speed;
                }
                for (int i = 0; i < heartx.length; i++) {
                    heartx[i] += speed;
                    boobx[i] += speed;
                    boosterx[i]+=speed;
                }
                for (int i = 0; i < size; i++) {
                    xpos[i] += speed;
                    bgxpos += speed;
                }
            }
```

```
if (ypos[0] < ypos[1] && xpos[0] == xpos[1]) {
    for (int i = 0; i < randomx.length; i++) {
        randomy[i] += speed;
    }
    for (int i = 0; i < hearty.length; i++) {
        hearty[i] += speed;
        booby[i] += speed;
        boostery[i]+=speed;
    }
    for (int i = 0; i < size; i++) {
        ypos[i] += speed;
        bgypos += speed;
    }
} else if (ypos[0] > ypos[1] && xpos[0] == xpos[1]) {
    for (int i = 0; i < randomx.length; i++) {
        randomy[i] -= speed;
    }
    for (int i = 0; i < hearty.length; i++) {
        hearty[i] -= speed;
        booby[i] -= speed;
        boostery[i]-=speed;
    }
    for (int i = 0; i < size; i++) {
        ypos[i] -= speed;
        bgypos -= speed;
    }
}
if (xpos[0] > xpos[1] && ypos[0] > ypos[1]) {
    for (int i = 0; i < randomx.length; i++) {

        randomx[i] -= speed;
        randomy[i] -= speed;
    }

    for (int i = 0; i < hearty.length; i++) {
        heartx[i] -= speed;
        hearty[i] -= speed;
        boobx[i] -= speed;
        booby[i] -= speed;
        boosterx[i] -= speed;
        boostery[i] -= speed;
    }
    for (int i = 0; i < size; i++) {
        xpos[i] -= speed;
        ypos[i] -= speed;
        bgxpos -= speed;
        bgypos -= speed;
    }
```

```
        }
    } else if (xpos[0] < xpos[1] && ypos[0] < ypos[1]) {
        for (int i = 0; i < randomx.length; i++) {
            randomx[i] += speed;
            randomy[i] += speed;
        }
        for (int i = 0; i < hearty.length; i++) {
            heartx[i] += speed;
            hearty[i] += speed;
            boobx[i] += speed;
            booby[i] += speed;
            boosterx[i] += speed;
            boostery[i] += speed;
        }
        for (int i = 0; i < size; i++) {
            xpos[i] += speed;
            ypos[i] += speed;
            bgxpos += speed;
            bgypos += speed;
        }
    }
    try {
        Thread.sleep(150);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    end_time = System.currentTimeMillis();
    repaint();
}
timelapse = "## playtime : " + sdf.format(end_time - start_time - 32400000);
```



Play시간을 띄우기 위한
System.currentTimeMillis();을 사용.

제한된 윈도우 내에서 최대한의 공간적인 효과를주기 위하여
우주인의 움직임과 산소통 연결줄 (두번째 동그라미)을
비교하여 배경과 아이템들의 움직임의 효과를 극대화시킨다.
또한 운석을 먹으면 2초가 멈춰지는 효과가 있기 때문에
Flag를 사용하여 동작을 분리하였다.


START GAME

완성을 위한 발자국

문제점과 보완할 점

-큰 틀의 필요성
반복적인 수정, 오류, 합칠 때 번거로움.

-MouseListener의 포기
일정하게 움직이는 것이 아니라 마우스의 속도에 따라 좌표값이 저장되서
간격이 일정하지 않아 산소줄이 다른 데로 가는 경우가 있었음.

-랜덤지렁이 포기
게임이 시작될 때마다 다른 길이의 지렁이, 그리고 다른 좌표값, 다른 움직임을 구현해야 되는데 시간을 들여 구현해낼 만한 것인지 의미를 찾지 못함.

-재시작했을 때 전체적인 속도가 빨라짐 (원인불명)


GAME OVER