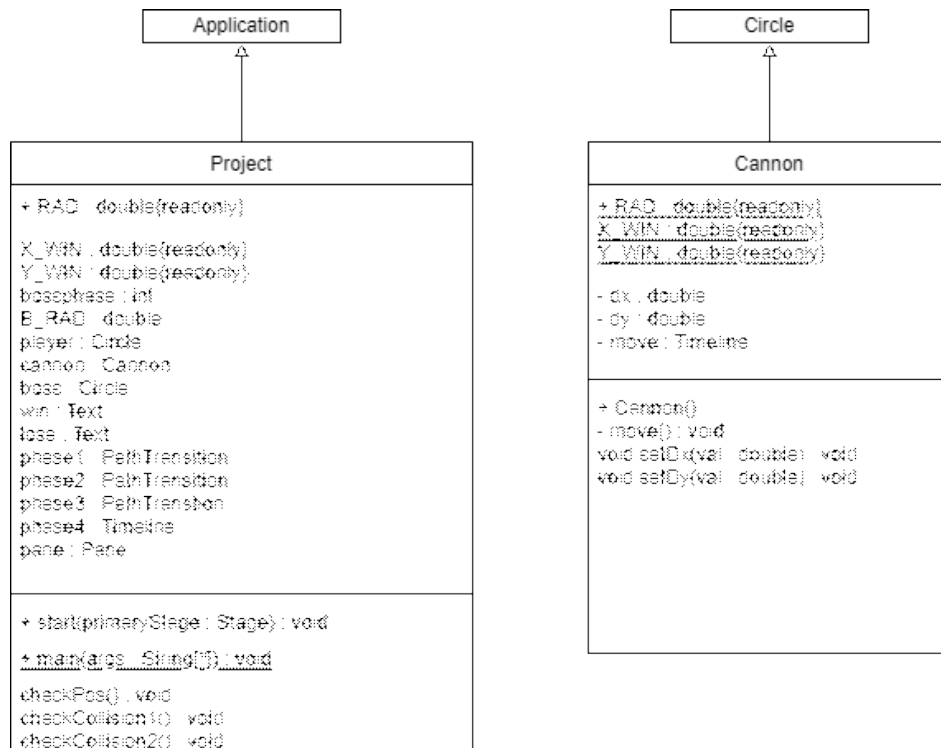
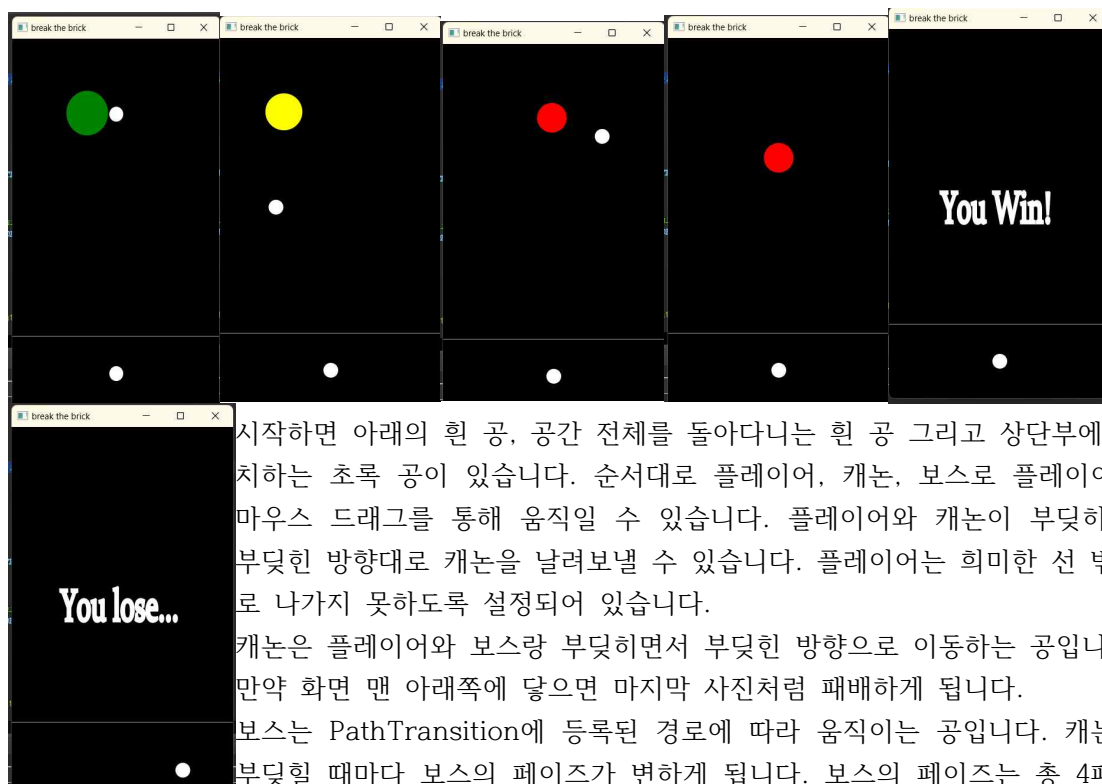


break the brick



어플리케이션의 목적 : 오락

사용 케이스, 돌렸을 때의 결과 :



시작하면 아래의 흰 공, 공간 전체를 돌아다니는 흰 공 그리고 상단부에 위치하는 초록 공이 있습니다. 순서대로 플레이어, 캐논, 보스로 플레이어는 마우스 드래그를 통해 움직일 수 있습니다. 플레이어와 캐논이 부딪히면, 부딪힌 방향대로 캐논을 날려보낼 수 있습니다. 플레이어는 희미한 선 밖으로 나가지 못하도록 설정되어 있습니다.

캐논은 플레이어와 보스랑 부딪히면서 부딪힌 방향으로 이동하는 공입니다. 만약 화면 맨 아래쪽에 닿으면 마지막 사진처럼 패배하게 됩니다.

보스는 PathTransition에 등록된 경로에 따라 움직이는 공입니다. 캐논과 부딪힐 때마다 보스의 페이지가 변하게 됩니다. 보스의 페이지는 총 4페이지로 1 페이지에는 초록색에 좌우로만 움직입니다. 2페이지에는 반지름이 줄고 노란색으로 바뀌며 지그재그로 움직입니다. 3 페이지에는 반지름이 한번 더 줄고 빨간색으로 바뀌며 빠른

속도로  $\propto$  식으로 움직입니다. 마지막 페이지가 되면 보스는 천천히 아래로 떨어지며 플레이어가 이깁니다.

```
package application;
```

```
import javafx.animation.KeyFrame;
```

```
import javafx.animation.Timeline;
```

```
import javafx.scene.layout.Pane;
```

```
import javafx.scene.paint.Color;
```

```
import javafx.scene.shape.Circle;
```

```
import javafx.util.Duration;
```

```
public class Cannon extends Circle{
```

```
    final static public double RAD = 10;
```

```
    final static double X_WIN = 300;
```

```
    final static double Y_WIN = 500;
```

```
    private double dx = 0 , dy = 1;
```

```
    private Timeline move;
```

```
    // RAD : 캐논의 반지름 , X_WIN : 화면의 가로사이즈 , Y_WIN : 화면의 세로 사  
이즈
```

```
    // dx , dy : 캐논의 x이동방향, y이동방향
```

```
    // move : 매 키프레임마다 캐논의 움직임을 표현
```

```
    public Cannon() {
```

```
        super(150, 300, RAD, Color.WHITE);
```

```
        // Circle생성자 작동 x위치,y위치,반지름, 색깔 설정
```

```
        move = new Timeline(
```

```
            new KeyFrame(Duration.millis(10), e-> move());
```

```
        move.setCycleCount(Timeline.INDEFINITE);
```

```
        move.play();
```

```
        // move에 키프레임 설정 및 매 프레임마다 작동할 함수 설정. 무한 반복하게  
해서 플레이
```

```
    }
```

```
    private void move() {
```

```
        if(getCenterX() < RAD || getCenterX() > X_WIN - RAD ) {
```

```
            dx = -dx;
```

```
        }
```

```
        // 만약 캐논이 왼쪽 벽이나 오른쪽 벽에 부딪친다면 x 이동방향을 반대로
```

한다.

```
if(getCenterY() < RAD) {  
    dy = -dy;  
}  
// 만약 캐논이 위쪽 벽에 부딪친다면 y 이동방향을 반대로 한다.
```

```
if(getCenterY() > Y_WIN - RAD) {  
    move.stop();  
    setFill(Color.BLACK);  
}
```

// 만약 캐논이 아래쪽 벽에 부딪친다면 캐논을 없애고(배경이 검은색이라 캐논을 검은색으로 하면 안보임), 동작을 멈춘다.

```
setCenterX(getCenterX() + dx);  
setCenterY(getCenterY() + dy);  
// 매 프레임마다 현재 x,y에다가 dx, dy를 더해 캐논의 위치를 정한다.  
}
```

```
void setDx(double val) {  
    this.dx = val;  
}
```

```
void setDy(double val) {  
    this.dy = val;  
}
```

//캐논의 dx, dy설정 함수

}

```
package application;
```

```
import java.util.*;
import javafx.animation.KeyFrame;
import javafx.animation.PathTransition;
import javafx.animation.Timeline;
import javafx.application.Application;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
import javafx.scene.shape.Polyline;
import javafx.scene.text.Text;
import javafx.stage.Stage;
import javafx.util.Duration;
```

```
public class Project extends Application {
    final public double RAD = 10;
    final double X_WIN = 300;
    final double Y_WIN = 500;
    double B_RAD = 30;
    int bossPhase = 1;
    // RAD : 플레이어와 캐논의 반지름, X_WIN, Y_WIN : 화면의 가로 사이즈, 세로
    사이즈
    // B_RAD : 보스의 반지름
    // bossPhase : 보스의 현재 페이지
    Circle player = new Circle(150, 450, RAD, Color.WHITE);
    Cannon cannon = new Cannon();
    Circle boss = new Circle(150, 100, B_RAD, Color.GREEN);
    // player , cannon , boss : 각각 플레이어 , 캐논 , 보스

    Text win, lose;
    // win, lose : 게임을 이기거나 질 때 나오는 텍스트

    PathTransition phase1, phase2, phase3;
    Timeline phase4;
    //phase1, phase2, phase3 : 각각 페이지 1, 페이지 2 , 페이지 3일 때의 보스 이
    동 경로대로 움직이게 하는 애니메이션
    //phase4 : 보스가 마지막 페이지일 때 매 프레임마다 보스의 위치를 정하는 타임라
```

인

```
Pane pane;
//pane : 화면을 표현하기 위한 pane

@Override
public void start(Stage primaryStage) {

    Polyline dontCross = new Polyline(0, 400, X_WIN, 400);
    dontCross.setStroke(Color.WHITE);
    dontCross.setOpacity(0.5);
    // 플레이어가 넘지 말아야하는 선 dontCross 초기화

    Polyline path1 = new Polyline(0, 100, X_WIN, 100);
    Polyline path2 = new Polyline(0, 100, 100, 100, 50, 150, 200, 150,
150, 200, X_WIN, 200);
    Polyline path3 = new Polyline(150, 100, 100, 100, 100, 150, 200, 150,
150, 100, 100, 50, 200, 50, 150, 100);
    // 보스가 페이지1, 페이지2,페이지3일때 따라 움직일 경로들 초기화

    win = new Text(-250,-250,"You Win!");
    win.setScaleX(3); win.setScaleY(5);
    lose = new Text(-250,-250,"You lose...");
    lose.setScaleX(3); lose.setScaleY(5);
    win.setStroke(Color.WHITE);
    win.setFill(Color.WHITE);
    lose.setStroke(Color.WHITE);
    lose.setFill(Color.WHITE);
    // 승리, 패배시 텍스트 초기화

    pane = new Pane();
    pane.setStyle("-fx-background-color: black;");
    pane.getChildren().addAll(cannon, player, boss, dontCross, win,
lose);

    // pane을 검은색으로 만들고 만든 객체들을 pane에 추가

    player.setOnMouseDragged(e ->
    {
        if(player.getCenterY()-RAD >= 400) {
            player.setCenterX(e.getSceneX());
            player.setCenterY(e.getSceneY());
        }
    });
}
```

```

        }
    }
};

// 플레이어를 드래그할 때 커서의 위치대로 움직이게 하는 함수, 플레이어
가 dontCross를 넘지 못하게 dontCross의 y위치 400보다
// 플레이어의 y 위치가 작다면 커서의 위치대로 움직이지 않는다.

Timeline updateTL = new Timeline();
updateTL.getKeyFrames().addAll(
    new KeyFrame(Duration.millis(0.5), e-> checkPos()),
    new KeyFrame(Duration.millis(0.5), e->
checkCollision1()),
    new KeyFrame(Duration.millis(0.5), e->
checkCollision2()));
updateTL.setCycleCount(Timeline.INDEFINITE);
updateTL.play();
// 매 프레임마다 플레이어, 캐논, 보스의 위치(checkPos), 플레이어와 캐논의
충돌 여부(checkCollision1),
//캐논과 보스의 충돌 여부(checkCollision2)를 확인하는 타임라인 updateTL

phase1 = new PathTransition(Duration.seconds(6), path1, boss);
phase1.setCycleCount(Timeline.INDEFINITE);
phase1.setAutoReverse(true);

phase2 = new PathTransition(Duration.seconds(4), path2, boss);
phase2.setCycleCount(Timeline.INDEFINITE);
phase2.setAutoReverse(true);

phase3 = new PathTransition(Duration.seconds(2), path3, boss);
phase3.setCycleCount(Timeline.INDEFINITE);
phase3.setAutoReverse(true);

phase4 = new Timeline(new KeyFrame(Duration.millis(1), e->{
    boss.setCenterY(boss.getCenterY() + 0.1);
}));
phase4.setCycleCount(Timeline.INDEFINITE);
// 각 phase를 초기화. 1, 2, 3페이지는 무한 반복하며 끝에 도달하면 반대로
가도록 설정

```

```

    phase1.play();
    //처음엔 보스의 페이지가 1이므로 페이지 1 시작

    Scene scene = new Scene(pane, 300, 500);
    primaryStage.setTitle("break the brick"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
    // 씬 설정 , X_WIN과 Y_WIN으로 사이즈를 설정함

}

public static void main(String[] args) {
    launch(args);
}

void checkPos() {
    if(player.getCenterY() - RAD < 400) {
        player.setCenterY(400 + RAD);
    }

    if(boss.getCenterY() > 600) {
        win.setX(120); win.setY(250);
    }

    if(cannon.getCenterY() > Y_WIN - RAD) {
        lose.setX(120); lose.setY(250);
        pane.getChildren().remove(boss);
    }

}

// 처음 if문은 플레이어가 dontCross를 넘으면 플레이어의 위치를 dontCross아래
로 설정해 dontCross에 막힌 것처럼 보이게 한다.
// 두 번째 if문은 보스가 화면 아래쪽에 도달하면 승리 텍스트를 보이게 한다.
// 세 번째 if문은 캐논이 아래쪽 벽에 닿으면 보스를 없애고 패배 텍스트를 보이게
한다.

void checkCollision1() {

```



```

double distX = cannon.getCenterX() - player.getCenterX();
double distY = cannon.getCenterY() - player.getCenterY();
if(Math.abs(distX) <= RAD * 2 && Math.abs(distY) <= RAD * 2 ) {
    cannon.setDx(distX / 10);
    cannon.setDy(distY / 7);
}
}
// 캐논과 플레이어 간의 x거리(distX), y거리(distY)를 계산한다. 이 두 거리가 RAD
*2보다 작으면 충돌했으므로
// 캐논의 방향을 바꾼다. 이때 /10 과 /7은 캐논의 속도 조절을 위해 넣었다.
void checkCollision2() {
    double distX = cannon.getCenterX() - (boss.getTranslateX() + 150);
    double distY = cannon.getCenterY() - (boss.getTranslateY() + 100);

    //캐논과 보스간의 x거리, y거리를 계산한다.
    //보스는 여기서 PathTransition을 통해 이동하므로 boss의 CenterX,
CenterY가 아닌 TranslateX, TranslateY를 가져와서 계산했다.
    //그리고 TranslateX와 TranslateY가 원하는 값에서 시작을 안해서 원하
는 값에서 시작하도록 150, 100을 더했다.

    if(Math.abs(distX) <= RAD + B_RAD && Math.abs(distY) <= RAD +
B_RAD ) {
        cannon.setDx(distX / 10);
        cannon.setDy(distY / 13);

        // 만약 둘의 x,y거리가 RAD+B_RAD보다 작다면 캐논의 방향을
조절한다.

        // /10 , /13은 속도 조절을 위해 넣었다.

        bossPhase++;
        // 캐논과 충돌할 때마다 보스의 페이즈가 1씩 올라간다.

        if(bossPhase == 2) {
            boss.setFill(Color.YELLOW);
            B_RAD -= 5;
            boss.setRadius(B_RAD);
            phase1.stop();
            phase2.play();
        }
        // 보스 페이즈가 2면 보스의 색깔을 노랑으로 바꾸고 반지름을 5
줄인 후 phase1을 멈추고 2를 시작한다.

```

```

else if (bossPhase == 3) {
    boss.setFill(Color.RED);
    B_RAD -= 5;
    boss.setRadius(B_RAD);
    phase2.stop();
    phase3.play();
}
// 보스 페이즈가 3이면 보스의 색깔을 빨강으로 바꾸고 반지름을
5줄인 후 phase2을 멈추고 3를 시작한다.
else {
    boss.setCenterX(150);
    boss.setCenterY(250);
    boss.setTranslateX(150);
    boss.setTranslateY(250);
    phase3.stop();
    phase4.play();
    cannon.setCenterX(-250);
    cannon.setCenterY(-250);
    pane.getChildren().remove(cannon);

}
// 보스 페이즈가 4면 보스가 화면의 가운데에서 살짝 위(150,
250)에 위치하게 하고 phase3를 멈추고 phase4를 시작한다.
// 캐논의 위치를 올려놓은 이유는 캐논을 제거했는데도 패배 텍스
트가 나오는 경우가 있어 추가하게 되었다.
// 가끔 보스가 제대로 된 위치가 아닌 곳에서 마지막 페이즈를 시
작하는 경우가 있는데 무엇이 원인인지 파악하지 못했다.
    }
}
}

```