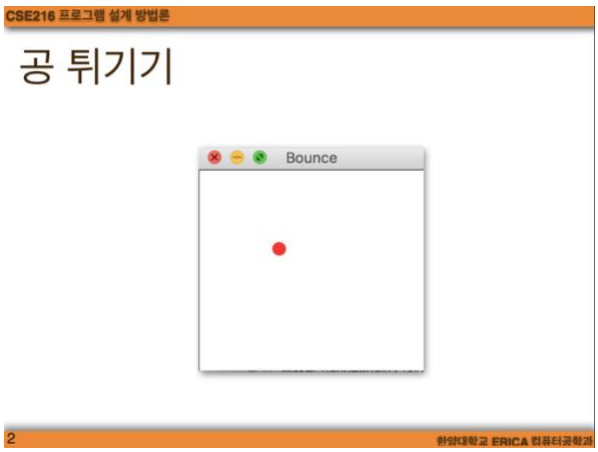
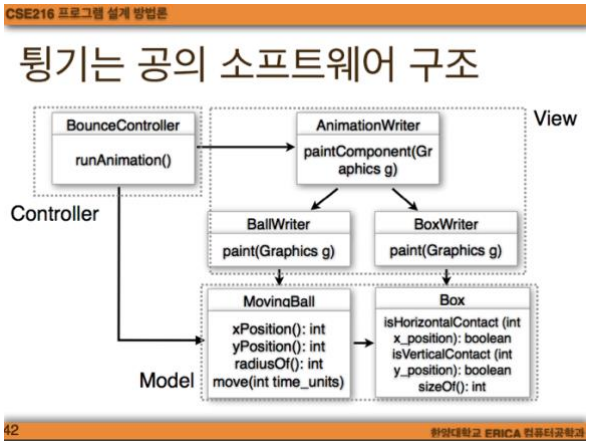
프로그램 설계 방법론

<Team Project>

- Bounce Ball –

소프트웨어학부 2014037729 강동혁
소프트웨어학부 2014038413 최준호

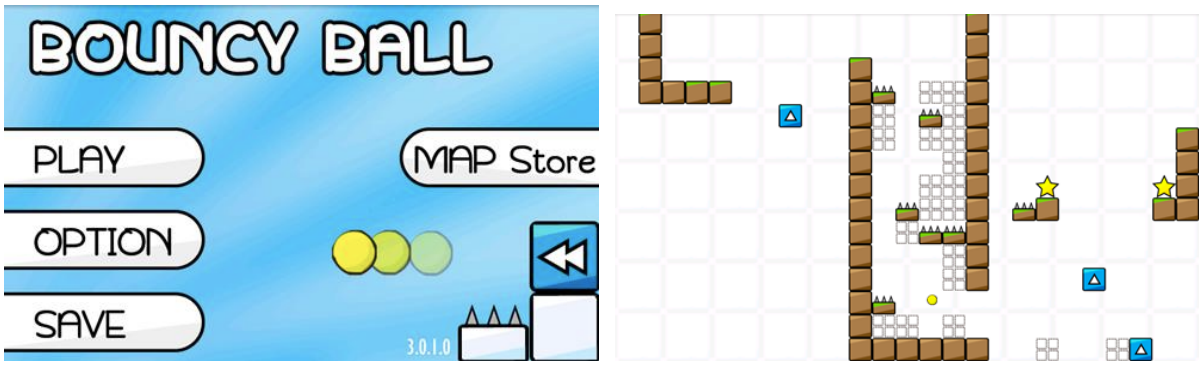




7. 반복: 루프와 재귀호출 에 나오는 '공 튀기기'를 주제로 선정하여 기능을 확장하였습니다. 기존에 예시로 나와있는 코드의 경우에는 ball, container 를 만들어 빨간 색 ball 이 정해진 container 안을 움직이며 벽에 부딪히면 튕기는 움직임만을 표현하였는데, 저희 팀의 경우 여기서 아이디어를 얻어 'Bounce Ball' 이라는 게임을 만들게 되었습니다.

- **Bounce Ball ?**

이미 BounceBall 이라는 게임은 PC, mobile 에 다양한 개발자들에 의해 많이 출시되어 있습니다. 저희는 그 중에서도 mobile 환경에서 플레이 할 수 있는 바운스볼 게임을 목표로 삼아 최대한 비슷하게 만들어보았습니다.



<mobile 환경의 Bouncy Ball>

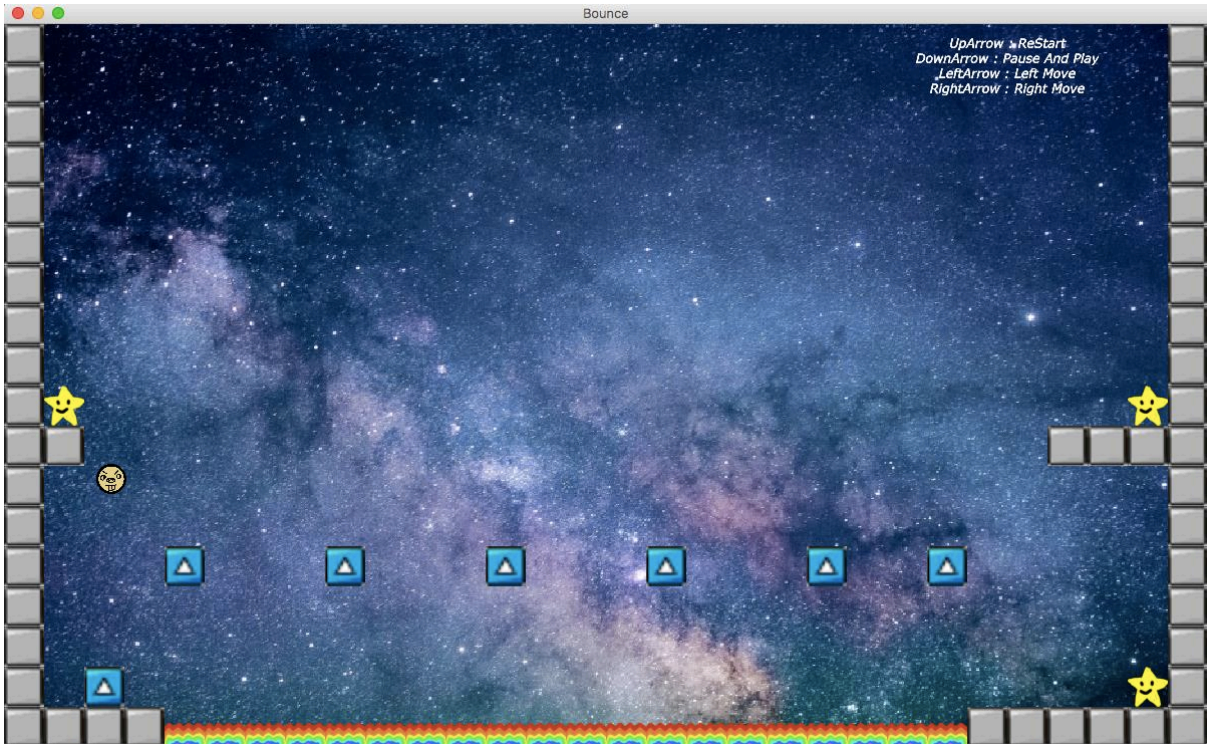
- **Bounce Ball !**

저희는 프로젝트를 진행하는데 **Model, View, Controller (MVC)** 를 나누어 이후에 유지보수를 하는데 보다 편하게 하기 위해서 체계적으로 코드를 작성하려 노력하였습니다.

그리고 저희 팀에서는 좀 더 편하게 Controller에 접근을 하기 위해서 **'싱글톤 패턴'** 을 이용해서 코딩을 하였습니다.

또한 각 stage를 표현하는데 '**Tilemap**' 방식을 활용하였습니다.

```
1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
3 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
4 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
5 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
6 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
7 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
8 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
9 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
10 1 9 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 9 1
11 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
12 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
13 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
14 1 0 0 0 3 0 0 0 3 0 0 0 3 0 0 0 3 0 0 0 3 0 0 3 0 0 0 0 1
15 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
16 1 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
17 1 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 9 1
18 1 1 1 1 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 1 1 1 1 1 1
```



크고 장면(Scene, Stage) 이 많을 경우에 배경이 되는 맵을 모두 메모리에 읽어들여야만 한다면 큰 메모리가 필요하기도 하고, 사실 java를 이용해 만든 'bounce ball' 게임의 경우 그렇게 큰 메모리가 필요하지는 않지만 추가로 'Tilemap'에 도전해보고 싶어 사용하였습니다.

1. **Block.java**

Bounce Ball 게임 특성상 ball의 역할보다 다양한 block들의 역할이 훨씬 중요합니다. Block.java 에서는 다양한 block들에 대해서 프로그래밍하고 저희가 원하는 위치에 각 stage마다 다르게 block들을 배치합니다.

```
class BlockRenderer extends ObjectRenderer {
    private BufferedImage[] blockImage;

    public BlockRenderer() {
        blockImage = new BufferedImage[10];
        blockImage[1] = FileManager.I.getImage("src/image/normal.png");
        blockImage[2] = FileManager.I.getImage("src/image/cloud.png");
        blockImage[3] = FileManager.I.getImage("src/image/superjump.png");
        blockImage[4] = FileManager.I.getImage("src/image/dead.png");
        blockImage[5] = FileManager.I.getImage("src/image/boom.png");
        blockImage[6] = FileManager.I.getImage("src/image/blank.png");
        blockImage[9] = FileManager.I.getImage("src/image/star.png");
    }
    public boolean isEnable() {
        return false;
    }
    public void paint(Graphics g) {
        int yIndex, xIndex;
        int blockSize = Information.I.blockSize;

        for(yIndex = 0; yIndex < Information.I.yMaxIndex; yIndex++) {
            for (xIndex = 0; xIndex < Information.I.xMaxIndex; xIndex++) {
                if(BlockController.I.isBlockEnable(xIndex, yIndex)) {
                    Block block = BlockController.I.getBlock(xIndex, yIndex);
                    g.drawImage(blockImage[block.getType()],
                            block.getXPosition(), block.getYPosition(),
                            blockSize, blockSize, GameRenderer.I);
                }
            }
        }
    }
}
```

그리고 각 block마다 직접 이미지를 그려 FileManager 를 통해 이미지를 부여하였습니다. **(View)**

- 각 block들 **(Model)**

 blockImage[1] : normal block

: 일반 벽의 역할을 하는 별다른 특징이 없는 block입니다.

 blockImage[2] : cloud block

: 밟으면 사라지는 block입니다.

+ cloud block의 경우

cloudDestroy 0 ~ 4 : cloud block을 밟았을 경우 AnimationRenderer 을 이용해 마치 유리가 깨지는 듯한 느낌으로 block이 사라지는 그림입니다.

 blockImage[3] : superjump block

: 밟으면 ball이 기존의 두 배 높이로 점프하게 되는 block입니다.

 blockImage[4] : dead block

: 밟으면 ball이 사라지고, 게임을 새로 시작하게 하는 block입니다.

 blockImage[5] : boom block

: 4방향 어디든 ball이 닿으면 ball이 사라지고, 게임을 새로 시작하게 하는 block입니다.

blockImage[6] : blank block

: normal block의 투명한 버젼입니다. (약간의 식별을 위해 빨간점들을 찍어 놓았습니다.)

blockImage[9] : star block

: 게임의 clear 조건으로, 모든 star block을 모으게 되면 게임이 다음 stage로 넘어가게 해주는 block 입니다.

```
class BlockController {
    public static BlockController I;
    private Block[][] blocks;

    public BlockController() {
        I = this;

        int yIndex, xIndex;
        int blockSize = Information.I.blockSize;

        blocks = new Block[Information.I.yMaxIndex][];
        for(yIndex = 0; yIndex < Information.I.yMaxIndex; yIndex++) {
            blocks[yIndex] = new Block[Information.I.xMaxIndex];
            for(xIndex = 0; xIndex < Information.I.xMaxIndex; xIndex++) {
                blocks[yIndex][xIndex] = new Block(
                        xIndex * blockSize, yIndex * blockSize,
                        xIndex, yIndex, false);
            }
        }
    }
    public void setMap(int [][] map) {
        int yIndex, xIndex;

        for(yIndex = 0; yIndex < Information.I.yMaxIndex; yIndex++) {
            for (xIndex = 0; xIndex < Information.I.xMaxIndex; xIndex++) {
                blocks[yIndex][xIndex].setType(map[yIndex][xIndex]);
                blocks[yIndex][xIndex].setEnable(map[yIndex][xIndex] != 0);
            }
        }
    }
    public void setBlockType(int xIndex, int yIndex, int type) {
        blocks[yIndex][xIndex].setType(type);
    }
    public void setBlockEnable(int xIndex, int yIndex, boolean enable) {
        blocks[yIndex][xIndex].setEnable(enable);
    }
    public int getBlockType(int xIndex, int yIndex) {
        return blocks[yIndex][xIndex].getType();
    }
    public boolean isBlockEnable(int xIndex, int yIndex) {
        return blocks[yIndex][xIndex].isEnable();
    }
    public Block getBlock(int xIndex, int yIndex) {
        return blocks[yIndex][xIndex];
    }
}
```

넘겨받은 map 정보에 따라서 block들에게 type을 부여합니다.

(Controller)

2. **Ball.java**

Ball은 Bounce Ball 게임을 진행하는 주인공 캐릭터입니다. 좌, 우로 움직여 여러 block들을 밟고 지나가 stage의 모든 star들을 모으는게 목표이며, loop를 돌려 항상 제자리에서 튀기고 있습니다. (튀기는 코드는 class Ball 안에서 일괄 처리하였습니다.)

```
    public void gravity(int deltaTime) {
        int yIndex, xIndex;
        int blockSize = Information.I.blockSize;
        int blockType;

        saveYPosition = yPosition;
        yPosition = yPosition + yVelocity * deltaTime;

        xIndex = xPosition / blockSize;
        yIndex = yPosition / blockSize;

        if(Information.I.isValidIndex(xIndex, yIndex)) {
            if (BlockController.I.isBlockEnable(xIndex, yIndex)) {
                blockType = BlockController.I.getBlockType(xIndex, yIndex);
                if(blockType == 9) {
                    BlockController.I.setBlockEnable(xIndex, yIndex, false);
                    GameManager.I.discountStar();
                }
                else if(blockType == 5)
                    GameManager.I.readStage(GameManager.I.getStage());
                else if (yVelocity >= 0) {
                    yVelocity = -14;

                    if(blockType == 2) {
                        BlockController.I.setBlockEnable(xIndex, yIndex, false);
                        GameRenderer.I.addRenderer(new AnimationRenderer(
                                "cloudDestroy", 5, 50,
                                xIndex * blockSize, yIndex * blockSize,
                                blockSize, blockSize, false));
                    }
                    else if(blockType == 3)
                        yVelocity = -18;
                    else if(blockType == 4)
                        GameManager.I.readStage(GameManager.I.getStage());
                } else
                    yVelocity = 3;
                yPosition = saveYPosition + yVelocity * deltaTime;
            }

            if (yVelocity <= 15)
                yVelocity += 1;
        }
    }
```

Gravity 안에는 제자리에서 튀기게 해주는 코드, 충돌처리, 각 block들에 닿았을 때 ball에게 변화를 주는 코드들이 있습니다.

ball : 게임 player **(Model)**

: ball을 조작하여 게임을 진행할 수 있습니다.

```
class BallRenderer extends ObjectRenderer
{
    private Ball ball;
    private Color color;
    private BufferedImage ballImage;

    public BallRenderer(Ball ball) {
        this.ball = ball;
        this.color = Color.red;
        ballImage = FileManager.I.getImage("src/image/ball.png");
    }
    public boolean isEnable() {
        return false;
    }
    public void paint(Graphics g) {
        int blockSize = Information.I.blockSize - 10;
        int radius = ball.getRadius();

        g.drawImage(
                ballImage,
                ball.getXPosition() - radius + 7, ball.getYPosition() - radius,
                blockSize, blockSize, GameRenderer.I);
    }
}
```

Ball을 나타내주는 코드입니다. **(View)**

3. FileManager.java

```
public BufferedImage getImage(String fileName) {
    try {
        return ImageIO.read(readFile(fileName));
    } catch (Exception e) {
        return null;
    }
}
```

각 model들 (ball, blocks) 에 맞는 image파일들을 불러오는데 사용되는 getImage입니다.

```
public int readStage(int stage) {
    int starCount = 0;

    String fileName = "src" + File.separator+"stage" +
            File.separator + "stage" + stage + ".txt";

    try {
        FileReader fileReader = new FileReader(readFile(fileName));

        int readChar;
        int index = 0;
        int blockSize = Information.I.blockSize;
        int yIndex, xIndex;
        int yMaxIndex = Information.I.yMaxIndex, xMaxIndex = Information.I.xMaxIndex;

        map = null;
        map = new int[yMaxIndex][];
        for(yIndex = 0; yIndex < yMaxIndex; yIndex++) {
            map[yIndex] = new int[xMaxIndex];
            for(xIndex = 0; xIndex < xMaxIndex; xIndex++)
                map[yIndex][xIndex] = 0;
        }

        while((readChar = fileReader.read()) != -1) {
            if(readChar >= 48 && readChar <= 58) {
                map[index / xMaxIndex][index % xMaxIndex] = readChar - 48;
                if(readChar-48 == 8) {
                    startXPosition = index % xMaxIndex * blockSize;
                    startYPosition = index / xMaxIndex * blockSize;
                    map[index / xMaxIndex][index % xMaxIndex] = 0;
                }
                if(readChar - 48 == 9)
                    starCount += 1;
                index++;
            }
        }

        fileReader.close();
    } catch (FileNotFoundException e) {
        System.out.println("file not found");
    } catch (IOException e) {
        System.out.println("error");
    }
    return starCount;
}
```

Txt 파일로 준비 된 각 stage들을 불러오는 역할을 해주는 readStage입니다.

파일이 없을 경우 file not found, 또 다른 에러가 발생했을 경우 error라는 문구가 나타나게 처리하였습니다.

4. **AnimationRenderer.java**

```
public AnimationRenderer(String fileName, int frameSize, int frameTime,
                         int xPosition, int yPosition,
                         int width, int height, boolean isLoop) {
    int index;

    this.fileName = fileName;
    this.frameSize = frameSize;
    this.frameTime = frameTime;
    this.xPosition = xPosition;
    this.yPosition = yPosition;
    this.width = width;
    this.height = height;
    this.isLoop = isLoop;
    isLooped = false;

    frame = 0;

    animationImage = new BufferedImage[frameSize];
    for(index = 0; index < frameSize; index++) {
        animationImage[index] = FileManager.I.getImage(
                "src/image/" + fileName + index + ".png");
    }
    isAnimation = true;
}
public void paint(Graphics g) {
    if(isLooped == false) {
        g.drawImage(
                animationImage[frame],
                xPosition, yPosition,
                width, height, GameRenderer.I);
        if(GameManager.I.nowTime >= nowTime) {
            nowTime = GameManager.I.nowTime + frameTime;
            frame = (frame + 1) % frameSize;
            if (frame == 0 && isLoop == false)
                isLooped = true;
        }
    }
}
```

게임을 진행하는데 필요한 Animation들을 처리해주기 위해 만든 코드입니다. 저희 프로젝트 내부에서는 유리모양의 cloudblock을 밟았을 경우 깨지는 모양의 animation밖에 없기 때문에 이것을 처리해주었습니다.

5. KeyManager.java

```
class KeyManager implements KeyListener{
    final int LEFT = 37;
    final int UP = 38;
    final int RIGHT = 39;
    final int DOWN = 40;

    public void keyTyped(KeyEvent e) {
        // TODO Auto-generated method stub

    }
    public void keyPressed(KeyEvent e) {
        if(e.getKeyCode() == UP) {
            GameManager.I.readStage(GameManager.I.getStage());
        }
        if(e.getKeyCode() == RIGHT) {

            Ball.I.setXVelocity(5);
        }
        if(e.getKeyCode() == DOWN) {
            GameManager.I.pauseAndplay();
        }
        if(e.getKeyCode() == LEFT) {
            Ball.I.setXVelocity(-5);
        }

    }
    public void keyReleased(KeyEvent e) {
        if(Ball.I.getXVelocity() < 0 && e.getKeyCode() == LEFT)
            Ball.I.setXVelocity(0);
        else if(Ball.I.getXVelocity() > 0 && e.getKeyCode() == RIGHT)
            Ball.I.setXVelocity(0);
    }
}
```

각 키 (위, 아래, 오른쪽, 왼쪽 방향키) 를 입력했을 경우 그에 맞는 action을 취하게 프로그래밍 하였습니다.

- **keyPressed** : key를 눌렀을 경우

위 방향키 : 게임 재시작
오른쪽 방향키 : 오른쪽으로 x의 velocity를 5증가
왼쪽 방향키 : 왼쪽으로 x의 velocity를 5증가 (오른쪽으로 5 감소)
아래쪽 방향키 : 게임 멈춤, 재입력 시 이어서 시작

- **keyReleased** : key를 손에서 떼었을 경우

왼쪽, 오른쪽 방향키를 떼었을 경우 x의 velocity가 0으로 초기화되도록 설정

6. GameManager.java

게임의 총괄 **Controller**입니다.

```
public void clearGame() {
    ball = null;
    blockController = null;
    gRenderer = null;
    isPlay = true;
}
```

게임을 시작하기에 앞서 clearGame을 통해 초기화합니다.
다양한 Scene전환에 대비하기 위해서 만들었습니다.

```
public void enterGame() {
    starCount = FileManager.I.readStage(stage);

    clearGame();

    blockController = new BlockController();
    blockController.setMap(FileManager.I.getMap());

    ball = new Ball();
    ball.setInformation(
            FileManager.I.getStartXPosition(),
            FileManager.I.getStartYPosition(),
            blockSize/2);

    gRenderer = new GameRenderer();
    gRenderer.addRenderer(new BlockRenderer());
    gRenderer.addRenderer(new BallRenderer(ball));

    isPlay = true;
}
```

게임에 실질적으로 진입 할 때, enterGame을 통해 각 object에 메모리를 할당합니다.

```
public int CollideBlock(int xIndex, int yIndex) {
    int type = blockController.getBlockType(xIndex, yIndex);

    if(blockController.isBlockEnable(xIndex, yIndex) == false)
        type = 0;

    switch(type) {
        case 2:
            if(ball.getYVelocity() >= 0)
                blockController.setBlockEnable(xIndex, yIndex, false);
                gRenderer.addRenderer(new AnimationRenderer(
                        "cloudDestroy", 5, 50,
                        xIndex * blockSize, yIndex * blockSize,
                        blockSize, blockSize, false));
            break;
        case 3:
            break;
        case 4:
            if(ball.getYVelocity() >= 0) {
                GameManager.I.readStage(GameManager.I.getStage());
                //SoundManager.I.PlaySound("src/sound/GameOver.wav");
            }
            break;
        case 5:
            GameManager.I.readStage(GameManager.I.getStage());
            //SoundManager.I.PlaySound("src/sound/GameOver.wav");
            break;
        case 9:
            blockController.setBlockEnable(xIndex, yIndex, false);
            GameManager.I.discountStar();
            break;
    }

    return type;
}
```

CollidBlock에서는 block에 충돌 했을 때 block에 대한 이벤트를 처리합니다.

```
public void readStage(int stage) {
    this.stage = stage;
    gRenderer.clearAnimation();

    starCount = FileManager.I.readStage(stage);

    ball.setInformation(
            FileManager.I.getStartXPosition(),
            FileManager.I.getStartYPosition(),
            blockSize/2);
    blockController.setMap(FileManager.I.getMap());
}
public int getStage() {
    return stage;
}
```

readStage, getStage를 통해 stage를 불러오고 현재 어떤 stage인지를 반환합니다.

```
public void discountStar() {
    starCount -= 1;
    if(starCount == 0)
        readStage(stage + 1);
}
```

Star block을 다 모았을 경우 다음 스테이지로 넘어가게 해주는 discountStar 입니다.

7. GameRenderer.java

```
public void paintComponent(Graphics g) {
    int index;

    g.drawImage(backgroundImage,
            0, 0,
            Information.I.screenWidth, Information.I.screenHeight,
            this);
    g.drawImage(exImage,
            Information.I.screenWidth - 300, -50,
            200, 200, this);
    for(index = 0; index < rendererList.size(); index++) {
        rendererList.get(index).paint(g);
        if(rendererList.get(index).isEnable() == true) {
            rendererList.remove(index);
        }
    }
```

전체적인 화면을 그려주는 Renderer 입니다. **(View)**

8. Information.java

```
class Information {
    public static Information I;

    public int screenHeight, screenWidth;
    public int containerHeight, containerWidth;
    public int yMaxIndex, xMaxIndex;
    public int blockSize;

    public Information() {
        I = this;
    }

    public void setScreenSize(int width, int height) {
        screenHeight = height;
        screenWidth = width;
    }
    public void setContainerSize(int width, int height) {
        containerHeight = height;
        containerWidth = width;
        xMaxIndex = containerWidth / blockSize;
        yMaxIndex = (containerHeight / blockSize) + 1;
    }
    public void setBlockSize(int blockSize) {
        this.blockSize = blockSize;
    }
    public boolean isValidXIndex(int xIndex) {
        return xIndex >= 0 && xIndex < xMaxIndex;
    }
    public boolean isValidYIndex(int yIndex) {
        return yIndex >= 0 && yIndex < yMaxIndex;
    }
    public boolean isValidIndex(int xIndex, int yIndex) {
        return xIndex >= 0 && xIndex < xMaxIndex &&
                yIndex >= 0 && yIndex < yMaxIndex;
    }
}
```

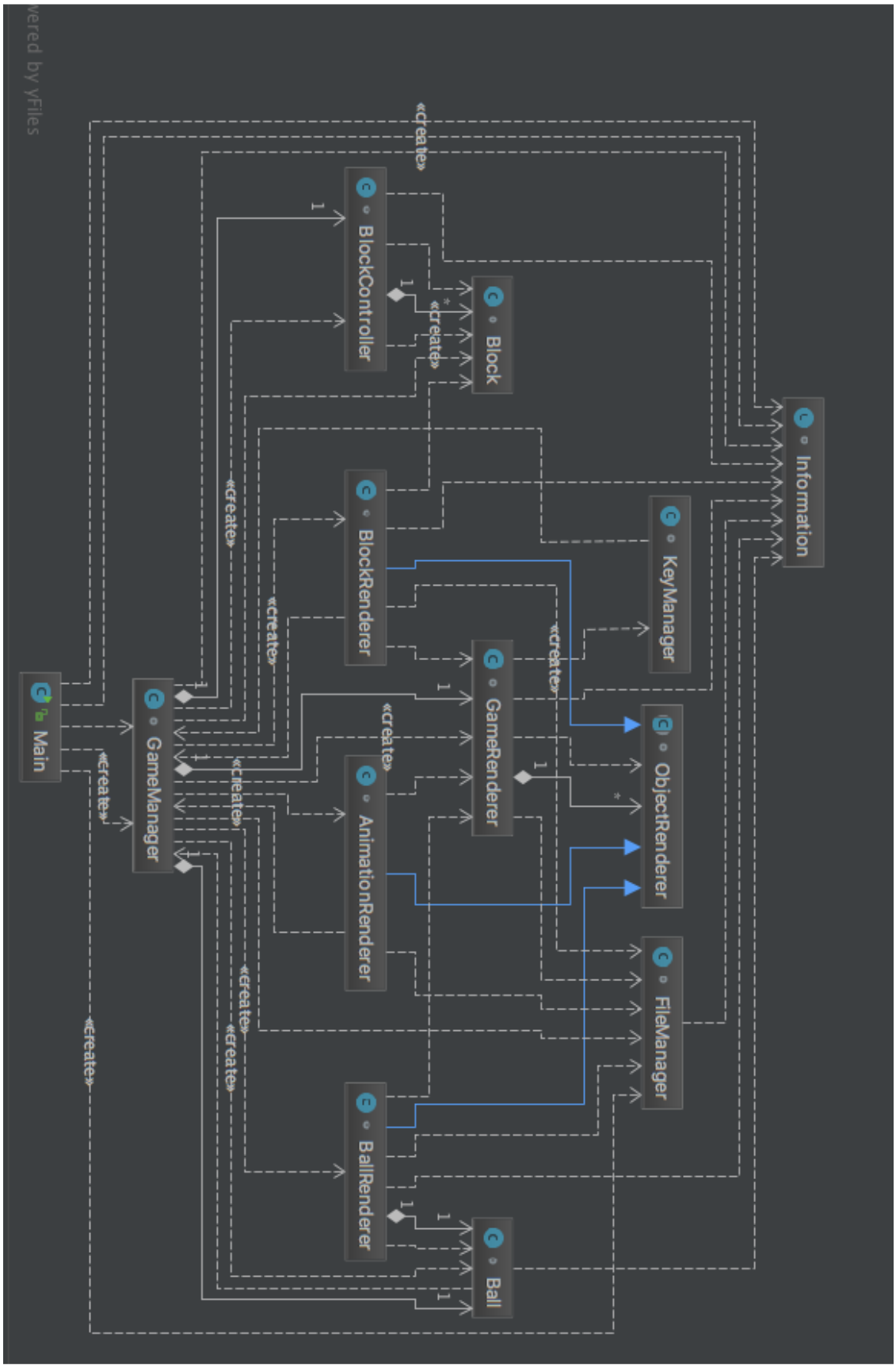
게임의 전반적인 정보를 닮고있습니다. (Screen size, block size 등)

9. **Main.java**

```
public class Main{
    public static Main I;
    private static String sceneName;

    public Main() {
        I = this;
    }

    public static void main(String[] args) {

        Information information = new Information();
        information.setScreenSize(1200, 740);
        information.setBlockSize(40);
        information.setContainerSize(1200, 700);

        //SoundManager soundManager = new SoundManager();
        //soundManager.I.PlaySound("src/sound/GameBGM.wav");

        new FileManager();

        setScene("Game");
        //GameManager.I.enterGame();
        //GameManager.I.gameLoop();
    }

    public static void setScene(String scene) {
        sceneName = scene;

        if(sceneName == "Game") {
            new GameManager();
            GameManager.I.enterGame();
            GameManager.I.gameLoop();
        }
    }
}
```

메인 함수입니다.

- 클래스 구조도 및 명세표 **(Class Diagram)**



■ **추상 클래스 및 인터페이스**

추상 클래스는 GameRenderer.java 의 ObjectRenderer로 BallRenderer, BlockRenderer, AnimationRenderer이 상속받아 사용하였습니다. 이유는 GameRenderer에서 각 Object(Animation) Renderer을 ArrayList를 이용해 편하게 호출 및 관리하기 위함입니다.

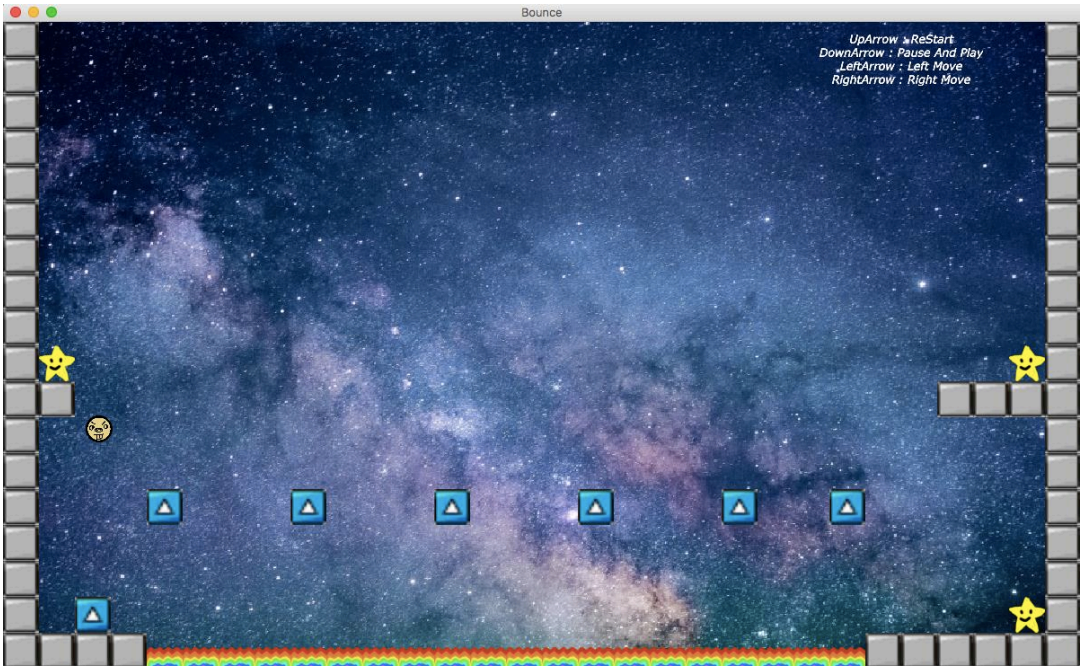
- **안정성**

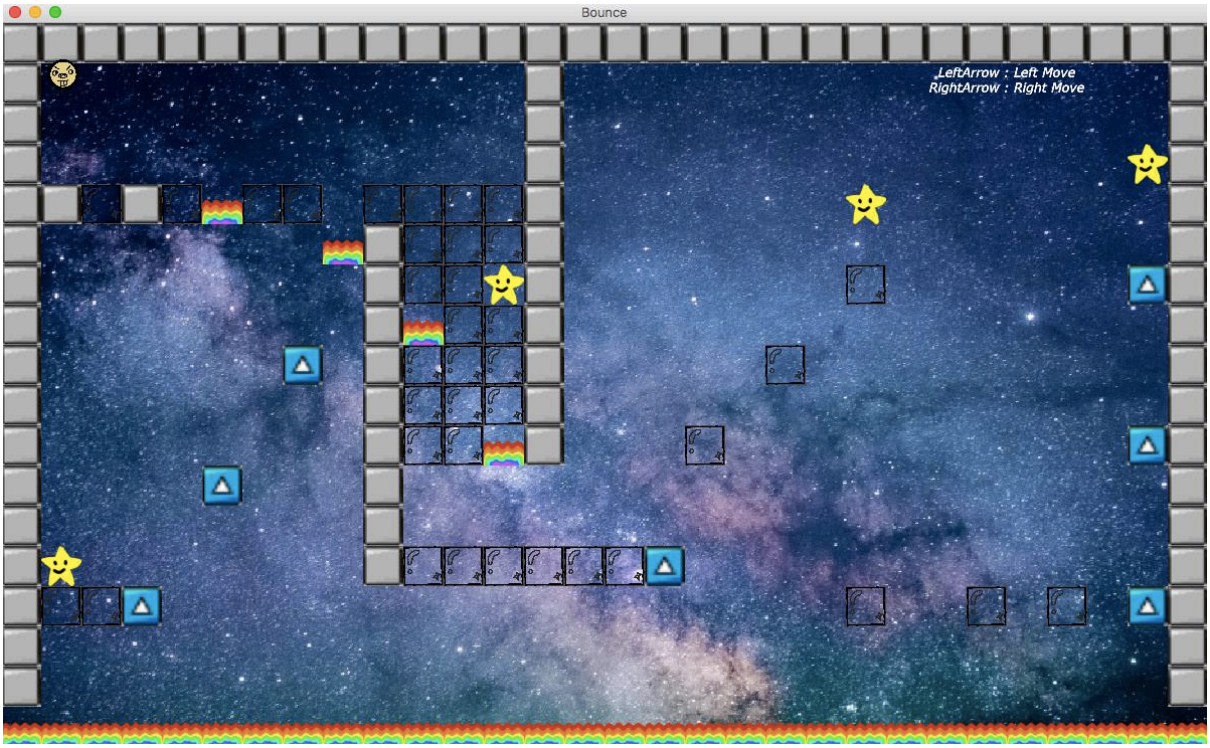
방향키만을 입력으로 받기 때문에 별다른 예외처리를 하지 않았습니다.

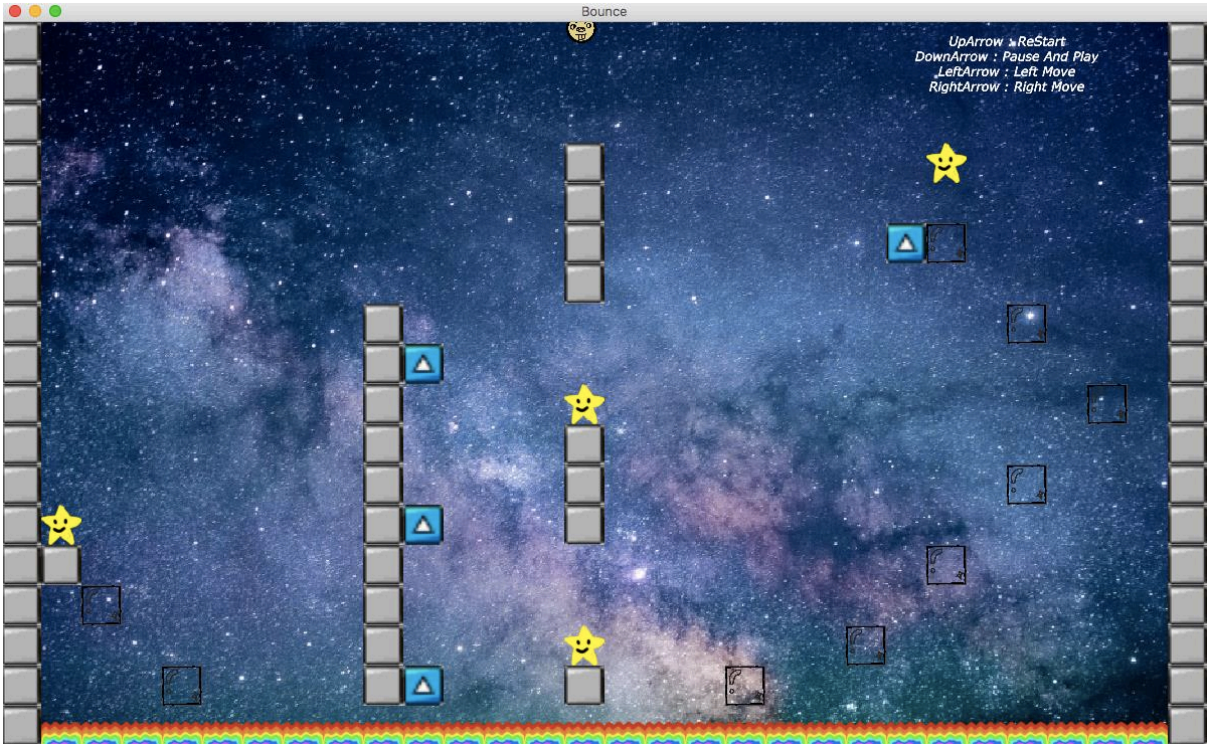
- **팀워크**

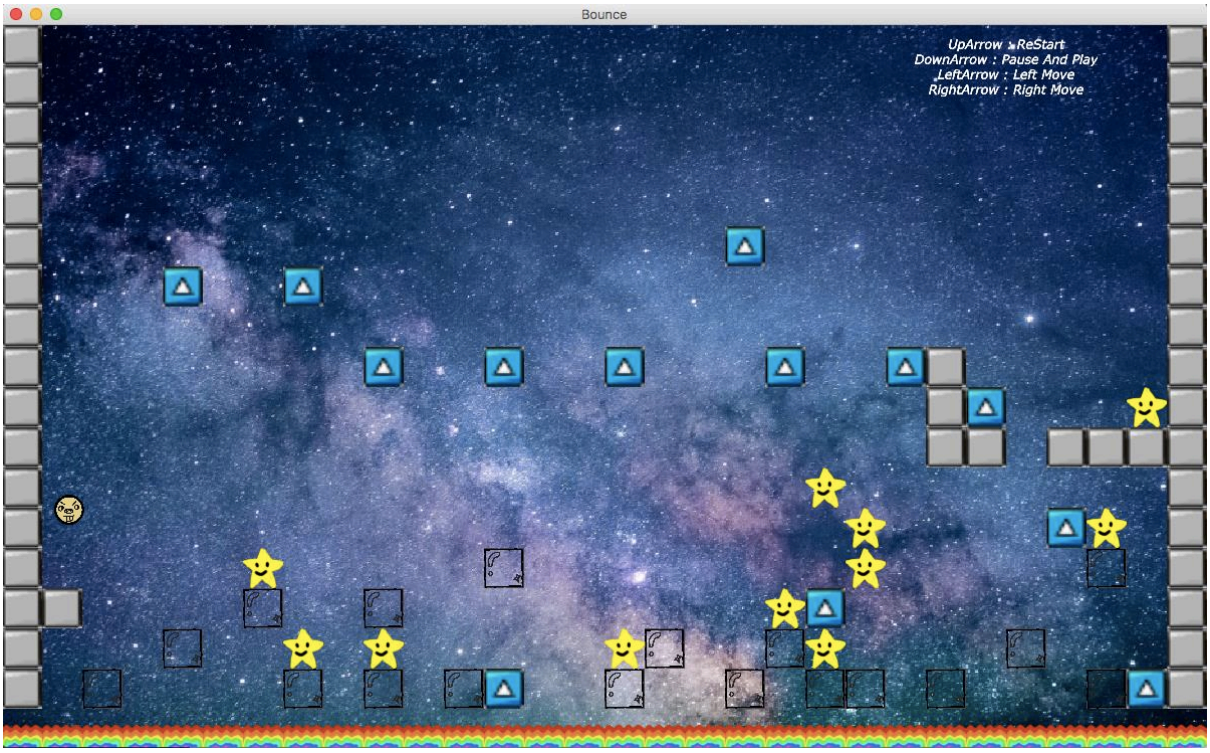
전반적인 게임 루프 : 강동혁
보고서 작성 , 이벤트 처리 , 스테이지 제작 : 최준호
클래스 구조도 설계 및 아이디어 구상 : 공동

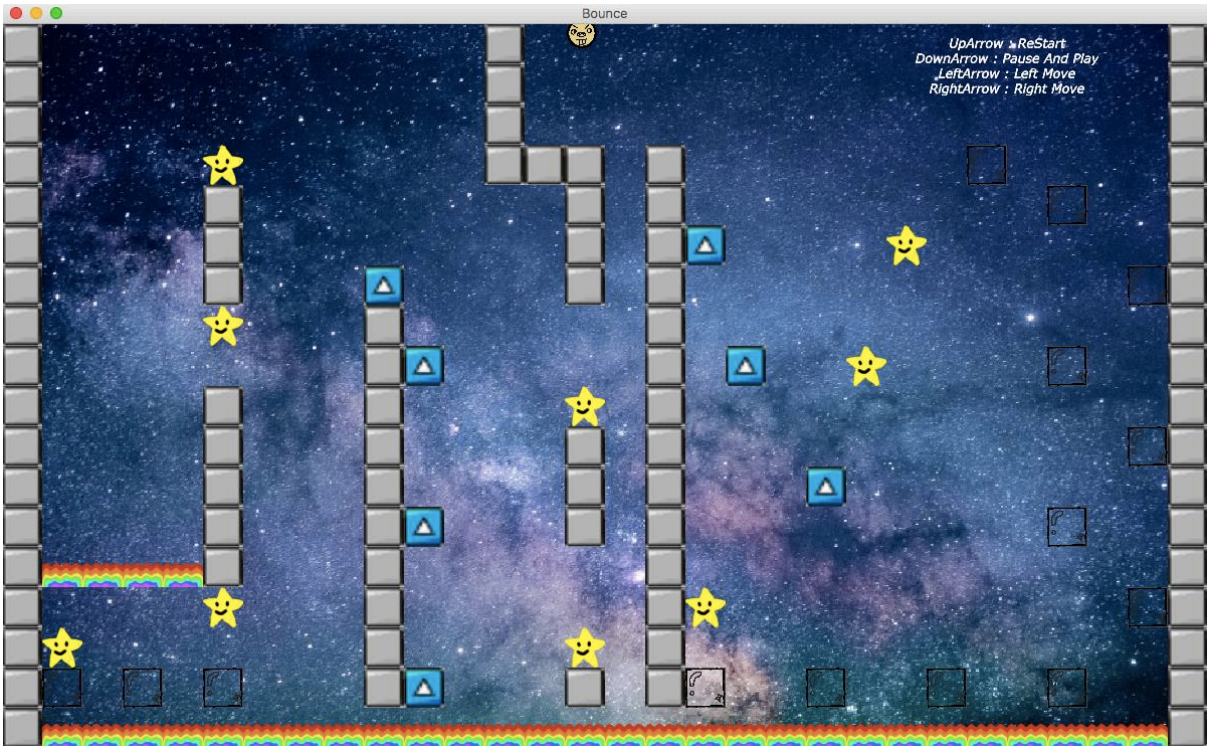
- **프로젝트 결과물**




Bounce
LeftArrow : Left Move
RightArrow : Right Move


Bounce
UpArrow : ReStart
DownArrow : Pause And Play
LeftArrow : Left Move
RightArrow : Right Move


Bounce
UpArrow : ReStart
DownArrow : Pause And Play
LeftArrow : Left Move
RightArrow : Right Move


Bounce
UpArrow : ReStart
DownArrow : Pause And Play
LeftArrow : Left Move
RightArrow : Right Move