BOUNCING BALL IN JAVA (OOPM)

INTRODUCTION

Our topic is about bouncing ball . As we have selected this program we are using java language using java code we are going to create a canvas area where the ball will bounce as per the program inputed. Java code is the core part by which the ball performs the bouncing task and other functions inputted . Java code is the core part by which the ball performs the bouncing task and other functions inputted in the program .We have used java code in bouncing ball because it is more easier than c++ programming . we have inputted various functions such as color changing of ball using java , on-off switch implementation in java, infinite loop in java which is a sequence of instruction that is continually repeated until a certain condition is reached . thus , using such function s our fully programmed bouncing ball game using java is created.

AIM

Bouncing Ball

This Mini-Project aims at:

1. Create a **Bouncing-Ball** Program.
2. Create and apply graphics to a ball to bounce.

THEORY

Let’s Learn More About Our Topic

WHAT IS BOUNING BALL?

#1

Bouncing ball animation in Java using FPanel, is a simple GUI(Graphical User Interface) Animation. Now Question is That how actually they are moving in a square so basically Bouncing ball animation in FPanel we are using a "If Loop" and running this “If Loop" Infinite and and changing there Ovals position.

#2

Writing a single ball bouncing inside a rectangular container box is straight forward, and can be accomplished with very few lines of codes, as follows

Dissecting BounceThread.java:

I assume that you understand Java Graphics programming (AWT/Swing and custom painting), and the multi-threading issues involved.

#3

In the constructor, we setup the UI components (set the preferred size for the JPanel). We then start a new thread to run the game update (moving the ball) .

we construct a JFrame as the application's main window. We set our custom JPanel as the content pane for the JFrame.

#4

This program, although works, is poor in design (in terms of modularity, reusability and expansibility). Moreover, the collision detection and response algorithm is crude. There is also no timing control.

SOURCE CODE

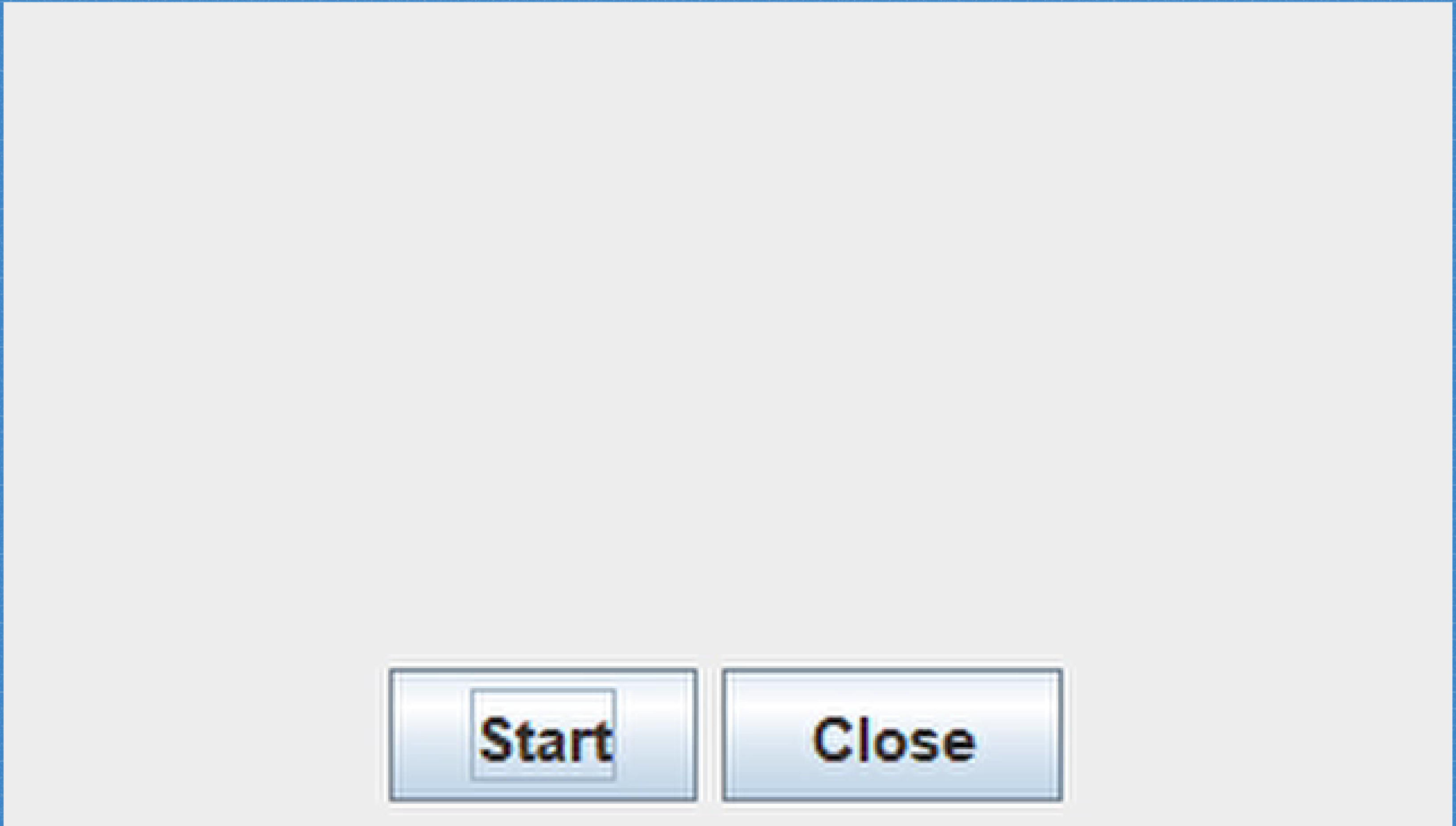
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class BounceThread
{
       public static void main(String[] args)
       {
              JFrame frame = new BounceThreadFrame();
              frame.show();
       }
}
```

```
class BounceThreadFrame extends JFrame
{
       private JPanel canvas;
       public BounceThreadFrame()
       {
                      setSize(300,200);
                      setTitle("Bounce");
                      Container contentPane = getContentPane();
                      canvas = new JPanel();
                      contentPane.add(canvas, "Center");
                      JPanel p = new JPanel();

              addButton(p, "Start",new ActionListener()
              {
```

```
public void actionPerformed(ActionEvent evt)
       {
       Ball b = new Ball(canvas);
       b.start();
       return;
       }
       \});
       addButton(p, "Close",new ActionListener()
        {
public void actionPerformed(ActionEvent evt){
       canvas.setVisible(false);
       System.exit(0);
       }
       });
       contentPane.add(p,"South");
       }
public void addButton(Container c,String title, ActionListener a)
       {
       JButton b = new JButton(title);
       c.add(b);
       b.addActionListener(a);
       }
}

class Ball extends Thread
{
       private JPanel box;
       private static final int XSIZE = 10;
       private static final int YSIZE = 10;
       private int x = 0;
       private int y = 0;
       private int dx = 2;
       private int dy = 2;

public Ball(JPanel b)
{
       box = b;
}

public void draw()
{
              Graphics g = box.getGraphics();
              g.fillOval(x,y,XSIZE,YSIZE);
              g.dispose();
}

public void move()
{
       if(!box.isVisible()) return;
       Graphics g = box.getGraphics();
       g.setXORMode(box.getBackground());
       g.fillOval(x,y,XSIZE,YSIZE);
       x += dx;
       y += dy;
       Dimension d = box.getSize();
       if(x<0)
```

```
{
        x = 0;
        dx = -dx;
}
        if (x + XSIZE >= d.width)
{
        x = d.width - XSIZE;
        dx = -dx;
}
        if(y<0)
{
        y = 0;
        dy = -dy;
}
```

```
       if (y + YSIZE >= d.height)
{
       y = d.height - YSIZE;
       dy = -dy;
}
       g.fillOval(x,y,XSIZE,YSIZE);
       g.dispose();
}

public void run()
{
       try
       {
              draw();
       while(true)
{
       move();
       sleep(5);
}
}
       catch(Exception e)
{}
}
}
```


Start
Close

CONCLUSION

Let’s Overview
What We Gained

Skill Developed / Learning out of this Mini-Project

- ❑ **Discipline knowledge: Apply Computer engineering discipline- specific knowledge to solve core computer engineering related problem.**
- ❑ **Experiments and practice: Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.**
- ❑ **Engineering tools: Apply relevant Computer technologies and tools with an understanding of the limitations.**
- ❑ **Communication: Communication effectively in oral and written form & increasing prodctivity by cordination.**

Reference Links

https://www3.ntu.edu.sg/home/ehchua/programming/java/J8a_GameIntro-BouncingBalls.html

https://youtu.be/LtqsENxj5TQ

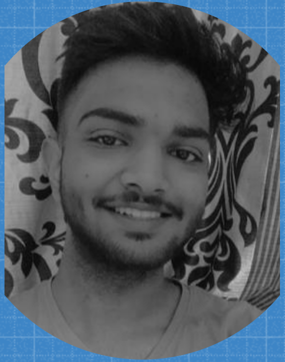
https://www.programmingwithbasics.com/2017/11/bouncing-ball-program-in-java-using.html?m=1

https://1000projects.org/magic-bouncing-ball-java-game.html?amp=1#

CREDITS

Our Group
Members

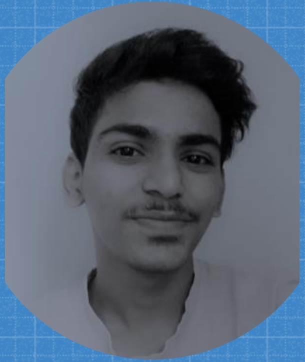
TEAM PRESENTATION



Sarvesh Kadam

Atharva Shelke

Parth Gorathe



Kalpesh Vangujar

Thank You!