NAME OF THE APPLICATION **:** Ping Pong Tennis

LANGUAGE **:** Core Java

GROUP MEMBER

NAME & ROLL NO. **:** Bhawesh Kumar Verma (47)

Abhishek Kumar Yadav (L7)

Mukul Ray (38)

SUBJECT NAME **:** Object Oriented Programming

SUBJECT CODE **:** CS504D

PROBLEM DESCRIPTION

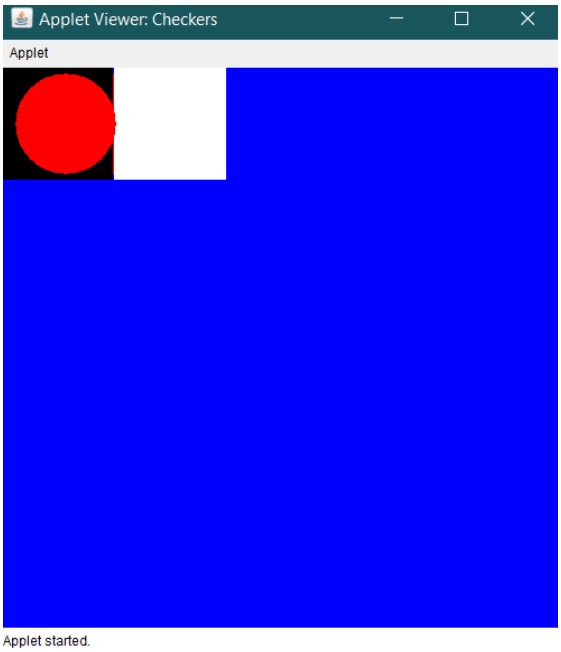
In the following source code, i.e., checkers.java, we can see that there is a red color ball that is animating through two boxes which are of black and white color. We can also see that there is a blue background. So, here the main problem is that the user does not get any information or no control over the animation going on. It is basically useless piece of code that is just animating the ball from left to right and then again resetting its position to extreme left, i.e., the initial position. If we observe closely, it has a flicker and also has a not so smooth motion. The color contrast is also not pleasing to our eyes. Its just a simple program code that shows how basic animation works using the concept of just threading. But as we know multi-threading has a lot more potential and usefulness. Therefore, it is not quit a good example to be shown. Below is the source code of ***checkers.java***.

SOURCE CODE

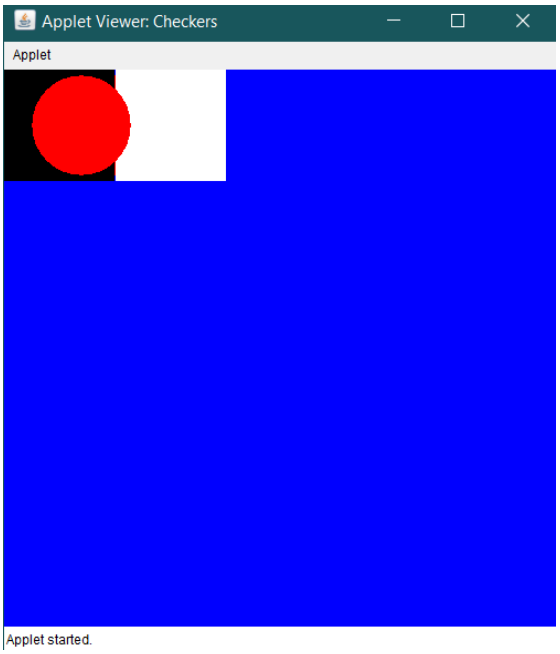
```
import java.awt.Graphics;
import java.awt.Color;
public class Checkers extends java.applet.Applet implements Runnable {
 Thread runner;
int xpos;
int ux1,ux2;
 public void start() {
        if (runner == null) {
                runner = new Thread(this);
                runner.start();
                }
}
public void stop() {
        if (runner != null) {
                runner.stop();
                runner = null;
                }
 }
```

```
public void run() {
        setBackground(Color.blue);
        while (true) {
                for (xpos = 5; xpos <= 105; xpos+=4) {
                if (xpos == 5) ux2 = size().width;
                else ux2 = xpos + 90;
                repaint();
                try { Thread.sleep(100); }
                catch (InterruptedException e) { }
                if (ux1 == 0) ux1 = xpos;
                }
        xpos = 5;
      }
}
public void update(Graphics g) {
        g.clipRect(ux1, 5, ux2 - ux1, 95);
                paint(g);
        }
public void paint(Graphics g) {
        // Draw background
        g.setColor(Color.black);
        g.fillRect(0, 0, 100, 100);
        g.setColor(Color.white);
        g.fillRect(101, 0, 100, 100);
         // Draw checker
        g.setColor(Color.red);
        g.fillOval(xpos, 5, 90, 90);
        // reset the drawing area
        ux1 = ux2 = 0;  }
}
```

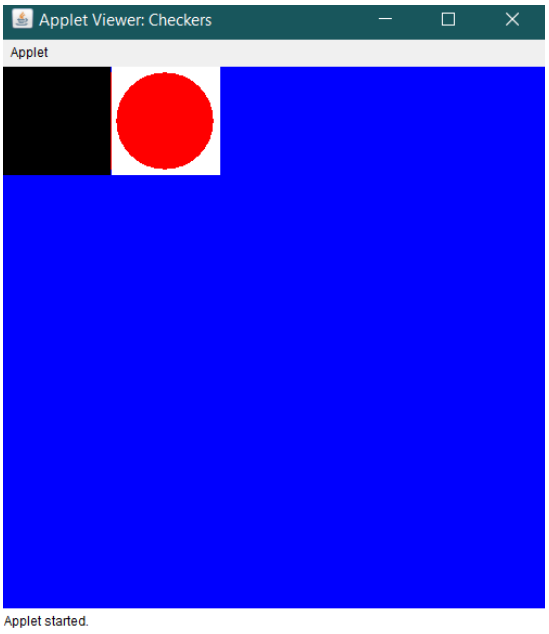
OUTPUT



1.Initial State



2.Transition State



3.Final State

NEW MODIFICATION

Inspired by the above java project, we have made some major modification in the code and made a game named “PING PONG TENNIS” which has easy to use UI (User Interface) and is fun playing. Not only that, you can play this game with your friend as it is multiplayer. Unlike the above project, it has smooth motion and great color combination which is not too harsh on your eyes and you don’t just have to sit and watch the ball animating as you can be the part of the animation and control it playing it.

We have used multi-threading to achieve multiple task simultaneously without any flicker or rough motion. Also, we have used several AWT packages like Buttons, ActionListener, KeyListener, etc., which is a good example for understanding java.awt and its working better.

Unlike above project, we have used different file present inside package to understanding basics of how package works and also it made easy to code the game.

The design is also modified and is much better than the above project. It has several options and buttons with clear labeling which helps for the user to understand better and easily.

Following are the source codes of the modified java project.

MODIFIED SOURCE CODE

MainUI.java

```
package assets;

import java.applet.Applet;

import java.awt.Button;

import java.awt.Color;

import java.awt.Font;

import java.awt.Graphics;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;


public class MainUI extends Applet implements ActionListener{

        Button start;

        Button retry;

        Button exit;

        Button crazy;
```

```
public static boolean play=true;
public void init() {
	if(play==true) {
	start = new Button("START");
	exit = new Button("EXIT");
	crazy = new Button("CRAZY");
	setBackground(Color.black);
	Font font = new Font("Courier", Font.BOLD,25);
	setFont(font);
	start.setBackground(Color.yellow);
	start.setBounds(230, 350, 400, 80);
	exit.setBounds(230, 450, 400, 80);
	crazy.setBounds(230, 350, 400, 80);
	start.addActionListener((ActionListener) this);
	exit.addActionListener((ActionListener) this);
	crazy.addActionListener((ActionListener) this);

	add(start);
	add(exit);

	setSize(900, 820);
	setLayout(null);
	System.out.println("Started");
	}
	else if(play==false) {
		retry = new Button("RETRY");
		exit = new Button("EXIT");
		setBackground(Color.black);
		Font font = new Font("Courier", Font.BOLD,25);
		setFont(font);
```

```
                retry.setBackground(Color.yellow);
                retry.setBounds(230, 450, 400, 80);
                exit.setBounds(230, 550, 400, 80);
                retry.addActionListener((ActionListener) this);
                exit.addActionListener((ActionListener) this);

                add(retry);
                add(exit);
                setSize(900, 820);
                setLayout(null);
                System.out.println("Retry?");
        }
}
public void paint(Graphics grp) {
        if(play==true) {
        grp.setFont(new Font("Magneto", Font.CENTER_BASELINE,80));
        grp.setColor(Color.yellow);
        grp.drawString("Ping Pong Tennis", 63, 250);
        grp.setFont(new Font("Calibri", Font.PLAIN,20));
        grp.setColor(Color.white);
        grp.drawString("Made By Bhawesh Kr. Verma © 2019", 275, 770);
        grp.setColor(Color.red);
        grp.fillRect(0, 790, 900, 30);
        for(int i=0; i<=900; i+=50) {
        int xt[] = {i, i+50, i+25};
        int yt[] = {790, 790, 810};
        int nop = 3;
        grp.setColor(Color.black);
        grp.fillPolygon(xt, yt, nop);
        }
```

```
                grp.dispose();
                }
                if(play==false) {
                        //GAME_OVER
                        grp.setColor(Color.red);
                        grp.fillRect(185, 100, 500, 250);
                        grp.setFont(new Font("Courier", Font.BOLD,70));
                        grp.setColor(Color.white);
                        grp.drawString("GAME OVER!", 230, 200);
                        }
                grp.dispose();
}
        public void actionPerformed(ActionEvent e) {
                if(e.getSource() == start){
                        remove(start);
                        remove(exit);
                        BallBounce ob;
                        ob = new BallBounce();
                        add(ob);
                        try {
                                Thread.sleep(3000);
                                play = true;
                                ob.init();
                                }catch(Exception ex) {}
                        }
                if(e.getSource() == retry){
                        play = true;
                        remove(retry);
                        remove(exit);
                        BallBounce ob = new BallBounce();
```

```
                        ob.init();
                        add(ob);
                        }
                        else if(e.getSource() == exit){
                        System.out.println("Game Exit");
                        play = false;
                        System.exit(0);
                        }
        }
}
```

BallBounce.java

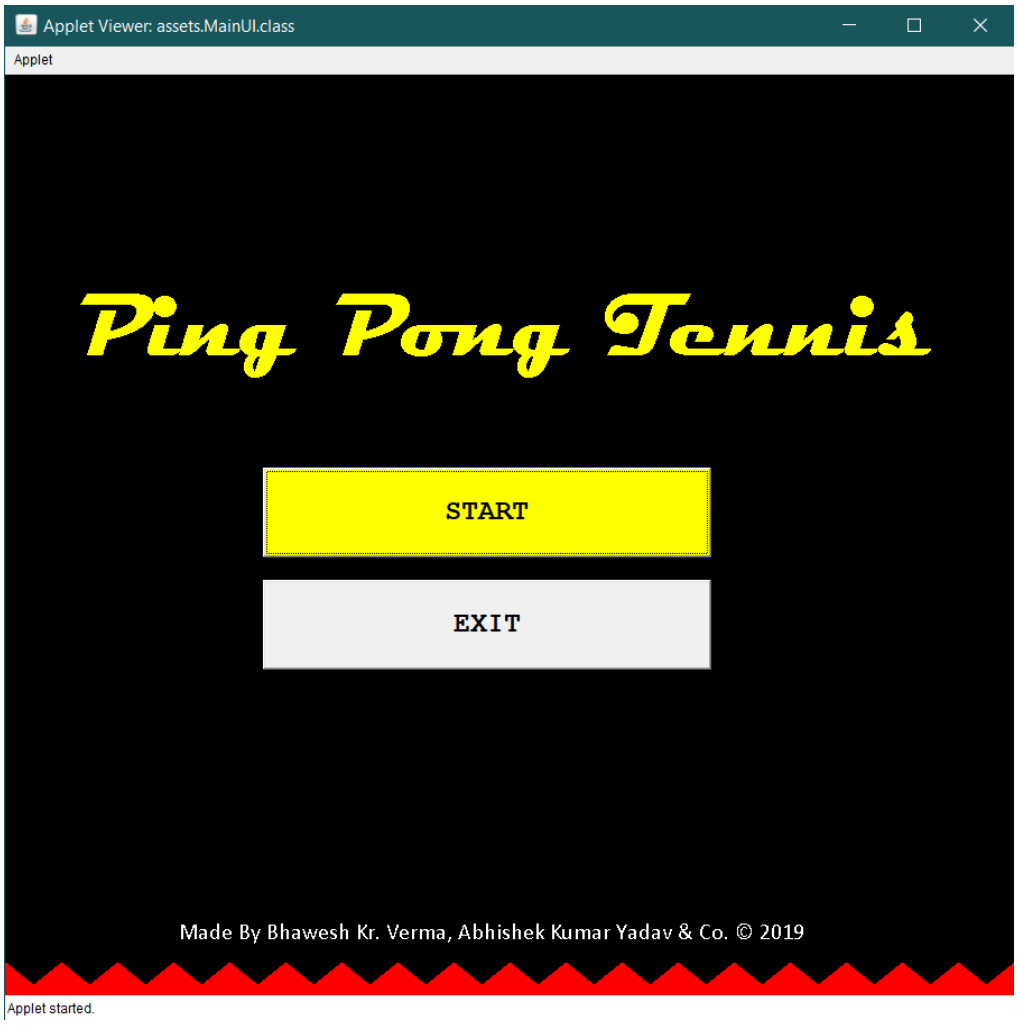
```
package assets;

import java.applet.*;
import java.awt.*;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

public class BallBounce extends Applet implements Runnable, KeyListener{

        public int width,height,x,y,dx,dy,size,speed,widths,width2s,heights;
        final double GRAVITY = 0.5;
        int inxBX = 0;
        int inxBY = 0;
        final double SPEED = .1;
        public boolean leftAccel = false;
        public boolean rightAccel = false;
        public boolean cleftAccel = false;
        public boolean crightAccel = false;
        double xps, yps, xVel, yVel, xcs, ycs, cxVel;
        public boolean timer = true;
        public boolean retry;
        Thread t1 = new Thread(this);

        public void init() {
                setSize(900, 785);
                t1.start();
        }


        public BallBounce(){
                System.out.println("Play");
                retry = false;
                speed=50;
                width=880;
                widths = 120;
```

```
                width2s = 60;
                heights = 15;
                height=770;
                x=430;
                xcs = 370;
                xps = 370;
                xVel = 0;
                cxVel=0;
                yVel = 0;
                y=350;
                yps = 770;
                ycs = 5;
                size=20;
                dx=3;
                dy=2;
                this.addKeyListener(this);

        }


public void paint(Graphics grp){
        if(MainUI.play == true) {
                grp.setColor(Color.green);
                grp.fillOval(x,y,size,size);
                grp.setColor(Color.blue);
                grp.fillRect((int)xps, (int)yps, widths, heights);
                grp.fillRect((int)xcs, (int)ycs, widths, heights);
                grp.setColor(Color.yellow);
                grp.fillRect(0, 400, 900, 5);
                grp.drawOval(350, 320, 180, 180);
                grp.dispose();
        }
        try{
                Thread.sleep(5);
        }
        catch(InterruptedException e){}
}
public void keyPressed(KeyEvent e) {
        if (e.getKeyCode() == KeyEvent.VK_A) {
                leftAccel = true;
        } else if (e.getKeyCode() == KeyEvent.VK_D) {
                rightAccel = true;
        } else if (e.getKeyCode() == KeyEvent.VK_LEFT) {
                cleftAccel = true;
        } else if (e.getKeyCode() == KeyEvent.VK_RIGHT) {
                crightAccel = true;
        }
}
public void run(){
        while(MainUI.play){
                repaint();
                x+=dx;
                y+=dy;
                if(x<=0 || x>=width)
                        dx*=-1;
                if(y>=heights+730 && x>=xps && x<=xps+widths) {
                        dy*=-1;
                        dx*=+1.5;
                }
```

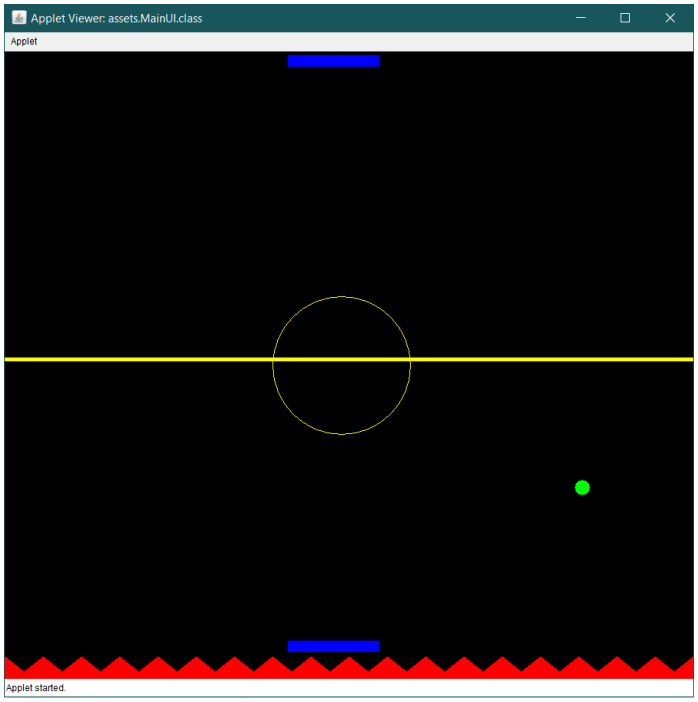
```
			if(y<=heights && x>=xcs && x<=xcs+widths) {
				dy*=-1;
				dx*=+1.8;
			}
			if(y>=760 || y<=0) {
				dy*=0;
				dx*=0;
				MainUI.play = false;
				System.out.println("Game Over");
				MainUI ob = new MainUI();
				ob.init();
				add(ob);
			}
			try{
				Thread.sleep(100/speed);
			}
			catch(InterruptedException e){}
				if (!leftAccel && !rightAccel)
					xVel *= GRAVITY;
				if (!cleftAccel && !crightAccel)
					cxVel *= GRAVITY;
				if (leftAccel) {
					xVel -= SPEED;
					}
				if (cleftAccel) {
					cxVel -= SPEED;
					}
				if (rightAccel)
				{
					xVel += SPEED;
				}
				if (crightAccel)
				{
					cxVel += SPEED;
				}

				xps += xVel;
				xcs += cxVel;

				if (xps >= 760)
					xps -= 1;
				else if (xps <= 30)
					xps += 1;
				if (xcs >= 760)
					xcs -= 1;
				else if (xcs <= 30)
					xcs += 1;
		try{
			Thread.sleep(5);
		}
		catch(InterruptedException e){}
			}
		}
	public void keyReleased(KeyEvent e) {
		if (e.getKeyCode() == KeyEvent.VK_A) {
			leftAccel = false;
		} else if (e.getKeyCode() == KeyEvent.VK_D) {
			rightAccel = false;
		}
```

```
        if (e.getKeyCode() == KeyEvent.VK_LEFT) {
                cleftAccel = false;
        } else if (e.getKeyCode() == KeyEvent.VK_RIGHT) {
                crightAccel = false;
        }
}
```

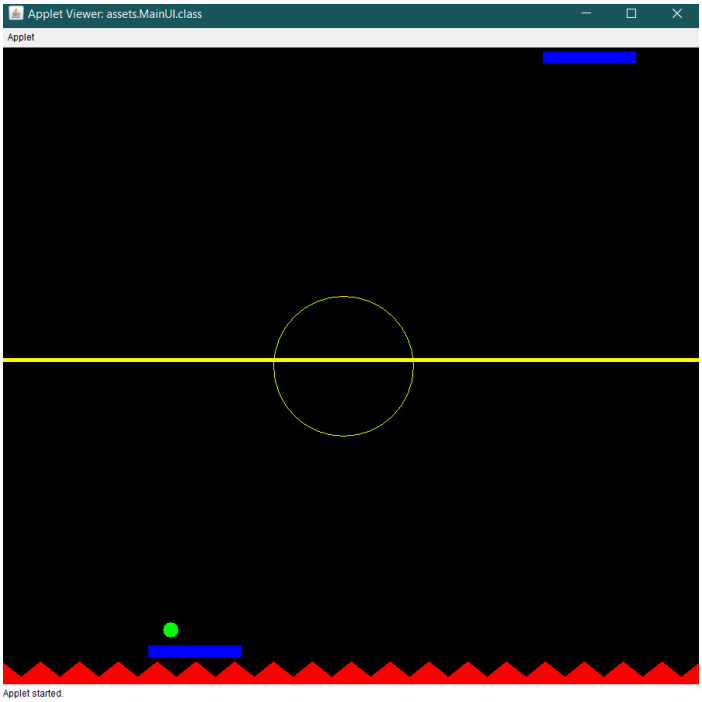
OUTPUT



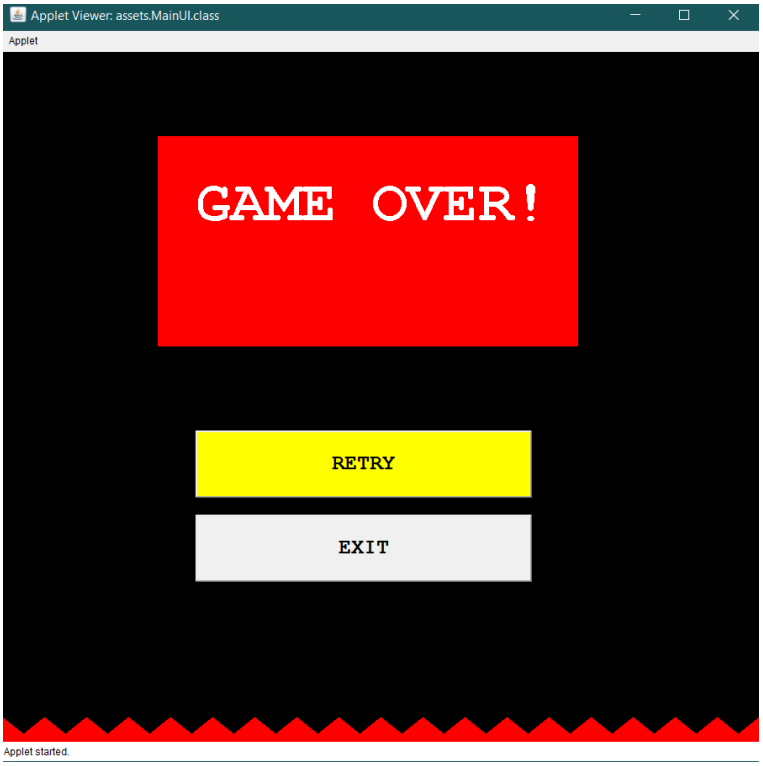
1. Start Screen



2. Initial Gameplay



3. Mid Gameplay



3. Final Game Screen

COMPARISONS ON OUTPUT

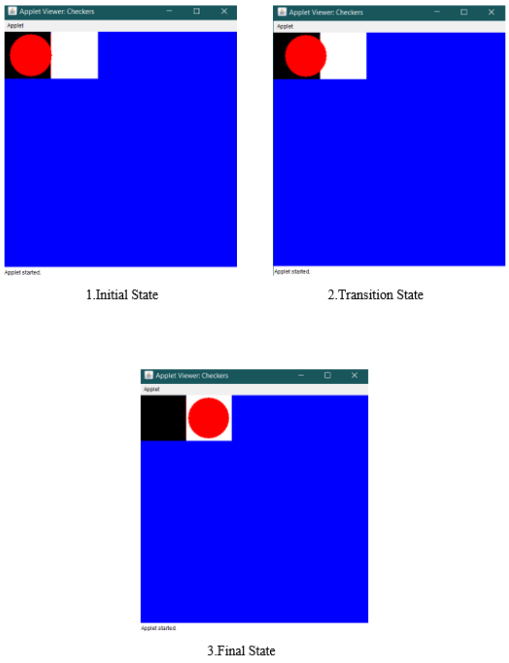


Fig1: *Checkers*

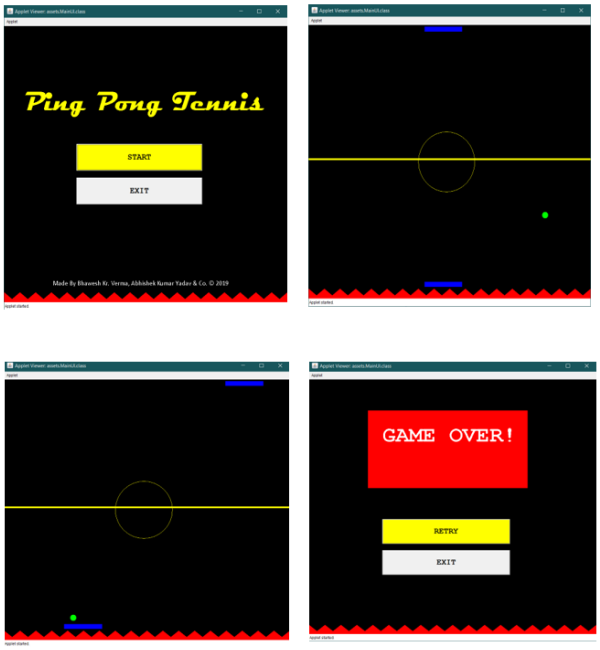


Fig2: *Ping Pong Tennis*

- If we compare the outputs of Fig1 & Fig2, we can clearly see the difference in design, i.e., unlike Fig1 the colors in our project is rich and attractive. Design itself is much more superior than that of Fig1.
- Next thing we can compare from output is that the features Fig1 provides is negligible, user have no control over the animation going on. On the other hand, our project (Fig2) gives lots of user interface with which you can actually interact and serve your purpose.
- Another aspect to compare on is the motion of the ball itself. In Fig1, the motion is restricted to only X-axis, as in Fig2 you can see that the motion is completely dynamic that is it moves along both X & Y-axis simultaneously. Not only that but also it reacts to the boundary and reflects back and change its direction.

DISCUSSION

This project we have made is a much better version of *checkers,* or you can say inspired from it. We have made a lot of changes and modifications to serve of purpose of making something useful.

At first, we have improved and smooth ball movement, with some speed control and ball size control. The ball has sense of direction and object detection that we can see as the ball collides with another object in the scene.

Another, one of the most important part of our project is usability, that is user can enjoy the animation and not only that but can also control the ball motion by the paddles provided. It's basically like tennis game, but instead it consists of a ping pong ball. User can enjoy this game with his/her friend and have a great time unlike the *checkers* where all you do is watch a ball moving from left to right.