Drew Sweeney

Dr. Hu

CSC341

March 31, 2022

Homework 7: Pong

This assignment was fun. I really enjoyed the freedom that the assignment gave us to create a game or animation of our choice. I really wanted to create the game pong in a two player format so that I could play with a friend. I felt and still feel that this was an ambiguous goal that I set. However, I was able to reach this goal. There are definitely some things that could be better but for the most part I am extremely happy with how my work turned out.

In this assignment I taught myself a lot of things. I had to learn Collisions and how to implement them into my code. Stackoverflow really came in handy in giving me good ideas to improve my own code. I also had to always incorporate the idea of a moving object for this I found a useful class that was already created called bufferStrategy. Truthfully, I am not exactly sure about how it works but I know its for animation and the drawing of objects. When I implemented it into my program it was simply for trial and error but then I was happy with the results that it should making it a crisper experience to play.

Another issue that wasn't as big but I had to fix was the problem with the blockers going off the game screen at first when this happened I was about ready to give up but I found a function that with the proper parameter would solve my issue

I also had to learn how to implement the game to take constant keystrokes from the user. This was a major pain at first I found a good resource to learn from but then it would just make it so the key pressed was constantly held down and not released right away and it was very frustrating. Finally I figured out how to do but there was still just a bit of lag between the user pressing and the key being released.

I also learned a little bit about the speed and directional changes about the ball before starting this project I did not know that 1 was up and -1 was down. This leads me to what I thought was the absolute hardest part of this project which was trying to get the ball to bounce in a specific direction based on the way it hit the blocker. Ultimately every try I had failed completely I have no idea how to get the ball to bounce in directions. I gave up and found a simple math solution to get the ball to bounce at the same angles repeatedly with the help of stackoverflow

I tried to incorporate a winning screen. I was able to get the game to finish when a certain score was reached however, I was not able to get the winning screen to show the game simply just stop and needs to be reset to be able to be played again

I also had a little trouble with telling the players the controls. I just wanted to simply display the text but not have it get in the way and I went back and fourth for a little while. Ultimately I wished to have the text shown for about 10 seconds and then have it fade off but I was not sure how to do that.

Though the assignment did say we were supposed to use the framework laid out for us in class I could just not wrap my head around it but I took a lot of time organizing this project and laying the code out in a readable way so I am still really happy with it.

Class ball: package homework7;

```
import java.awt.Color;
import java.awt.Graphics;

public class Ball
{
	public static final int SIZE = 16;

	private int x, y;
	private int xVelocity, yVelocity; //1 or -1 1 == up and -1 == down
	private int speed = 5; //speed of ball

	public Ball()
	{
		reset();
	}

	private void reset()
	{
		//set initial position of ball
		x = Game.WIDTH / 2 - SIZE / 2;
		y = Game.HEIGHT / 2 - SIZE / 2;

		//initial velocities
```

```
            xVelocity = Game.sign(Math.random() * 2.0 - 1);
            yVelocity = Game.sign(Math.random() * 2.0 - 1);
}

public void changeYDirection()
{
            yVelocity *= -1;
}
public void changeXDirection()
{
            xVelocity *= -1;
}

public void draw(Graphics g)
{
            g.setColor(Color.white);
            g.fillRect(x, y, SIZE, SIZE);

}

public void update(Blocker blocker1, Blocker blocker2)
{
            //updates movement
            x += xVelocity * speed;
            y += yVelocity * speed;

            //collisions
            if (y + SIZE >= Game.HEIGHT || y <= 0)
            {
```

```
                changeYDirection();
            }

            //with walls collisions
            if (x + SIZE >= Game.WIDTH)
            {
                //if over value add score
                blocker1.addScore();
                //resets ball from center in random direction
                reset();
            }

            if (x <= 0)
            {
                blocker2.addScore();
                reset();
            }
        }

        public int getX()
        {
            return x;
        }

        public int getY()
        {
            return y;
        }
}
```

Class Blocker: package homework7;

```
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;

public class Blocker
{
	Game game;
	private int x, y;
	private int velocity = 0;
	private int speed = 10;
	private int width = 22;
	private int height = 85;
	private int score = 0;
	private Color color;
	private boolean left;
	private boolean gameOver;

	public Blocker(Color c, boolean left)
	{
		color = c;

		this.left = left;
```

```
		//initialize position of blocker
		if (left)
		{
			x = 0;
		}
		else
		{
			x = Game.WIDTH - width;
		}

		y = Game.HEIGHT / 2 - height /2;

}

public void addScore()
{
		if (score < 2)
		{
			score++;
		}
		else
		{
			gameOver = true;
		}
}

public void draw(Graphics g)
{
		if (gameOver)
```

```
{
        game.drawBackgroundGameOver(g);
}
else
{
        //draw blocker
        g.setColor(color);
        g.fillRect(x, y, width, height);

        //draw score
        int stringX;
        //from score to string text
        String scoreText = Integer.toString(score);
        //set font
        Font font = new Font("Roboto", Font.PLAIN, 50);
        //sets width of text and gets text
        int stringWidth = g.getFontMetrics(font).stringWidth(scoreText) + 1;
        //distance between center line and each score
        int padding  = 25;
        int padding2 = 450;
        int padding3 = 500;

        //string so user knows how to play
        String howToPlay = "W and S : ArrowUp and ArrowDown";

        //Max score till game ends text
        String endScore = "First one to 3 Wins!";

        if (left)
```

```
                {
                        stringX = Game.WIDTH / 2 - padding - stringWidth;
                }
                else
                {
                        stringX = Game.WIDTH / 2 + padding;
                }

                g.setFont(font);
                g.drawString(scoreText, stringX, 50);

                // size and font change for long text
                Font myFont = new Font("Serif", Font.BOLD, 12);
                g.setFont(myFont);
                g.drawString(howToPlay, 100, padding2);
                g.drawString(endScore, 150, padding3);
        }
}

public void update(Ball ball)
{
        //update position only if possible
        y = Game.possibleRange(y += velocity, 0, Game.HEIGHT - height);

        int ballX = ball.getX();
        int ballY = ball.getY();

        //collisions with the ball and blockers
        if (left)
```

```
		{
			//collision for left blocker
			if (ballX <= width && ballY + Ball.SIZE >= y && ballY <= y + height)
			{
				ball.changeXDirection();
			}
		}
		else
		{
			//collision for right blocker
			if (ballX + Ball.SIZE >= Game.WIDTH - width && ballY + Ball.SIZE >= y && ballY <=
y + height)
			{
				ball.changeXDirection();
			}
		}
	}

	public void switchDirection(int direction)
	{
		velocity = speed * direction;
	}

	public void stop()
	{
		velocity = 0;
	}
}
```

Class Game:

```
package homework7;

import java.awt.BasicStroke;
import java.awt.Canvas;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Stroke;
import java.awt.image.BufferStrategy;

public class Game extends Canvas implements Runnable
{

	private static final long serialVersionUID = 1L;

	//to have a good 9:16 aspect ratio
	public static final int WIDTH = 1000;
	public static final int HEIGHT = WIDTH * 9/16;

	public boolean running = false;
	private Thread gameThread;
```

```
private Ball ball;
private Blocker blocker1;
private Blocker blocker2;

//main...runs game
public static void main(String[] args)
{
	new Game();
}

//game constructor
public Game()
{
	//set up window for game
	canvasSetup();
	//brings in classes for game
	initialize();

	new Window("Pong", this);

	this.addKeyListener(new UserInput(blocker1, blocker2));

	this.setFocusable(true);
}

private void initialize()
{
	//initialize ball
	ball = new Ball();
```

```
        //initialize blocker
        blocker1 = new Blocker(Color.BLUE, true);
        blocker2 = new Blocker(Color.RED, false);
}

private void canvasSetup()
{
        this.setPreferredSize(new Dimension(WIDTH, HEIGHT));
        this.setMaximumSize(new Dimension(WIDTH, HEIGHT));
        this.setMinimumSize(new Dimension(WIDTH, HEIGHT));
}

@Override
public void run()
{
        this.requestFocus();
        long lastTime = System.nanoTime(); //popular game loop from stackoverflow
        double amountOfTicks = 60.0;
        double ns = 1000000000 / amountOfTicks;
        double delta = 0;
        long timer = System.currentTimeMillis();
        //int frames = 0;
        while (running)
        {
                long now = System.nanoTime();
                delta += (now - lastTime) / ns;
                lastTime = now;
                while (delta >= 1)
                {
```

```
                update();
                delta--;
            }
            if (running)
            {
                draw();
                //frames++;
            }
            if (System.currentTimeMillis() - timer > 1000)
            {
                timer += 1000;
                //System.out.println("FPS: "  + frames); count frames
                //frames = 0;
            }
        }
        stop();
}

private void draw()
{
        //initializing the drawing tools
        //buffer strategy found online
        //used for drawing to the screen with buffer.show
        BufferStrategy buffer = this.getBufferStrategy();

        if (buffer == null)
        {
            this.createBufferStrategy(3);
            return;
```

```
                }

                Graphics g = buffer.getDrawGraphics();

                //draw background
                drawBackground(g);

                //draw blockers and score
                blocker1.draw(g);
                blocker2.draw(g);
                //draw ball
                ball.draw(g);
                //draw
                g.dispose();
                buffer.show();
        }

        private void drawBackground(Graphics g)
        {
                //black back
                g.setColor(Color.black);
                g.fillRect(0, 0, WIDTH, HEIGHT);

                //white dotted line
                g.setColor(Color.white);
                Graphics2D g2d= (Graphics2D) g;
                Stroke dashedLine = new BasicStroke(3, BasicStroke.CAP_BUTT,
BasicStroke.JOIN_BEVEL, 0, new float[] {10}, 0);
                g2d.setStroke(dashedLine);
```

```
        g2d.drawLine(WIDTH / 2, 0, WIDTH / 2, HEIGHT);
}

public void drawBackgroundGameOver(Graphics g)
{
        String gameOverString = "Game Over :(";
        //black back
        g.setColor(Color.black);
        g.fillRect(0, 0, WIDTH, HEIGHT);
        Font myFontGameOver = new Font("Serif", Font.BOLD, 60);
        g.setFont(myFontGameOver);
        g.setColor(Color.WHITE);
        g.drawString(gameOverString, 100, 400);
}

private void update()
{
        //update ball
        ball.update(blocker1, blocker2);
        //update blockers
        blocker1.update(ball);
        blocker2.update(ball);
}

public void start()
{
        gameThread = new Thread(this);
        gameThread.start();
        running = true;
```

```
}

public void stop()
{
	try
	{
		gameThread.join();
		running = false;
	} catch (InterruptedException e)
	{
		e.printStackTrace();
	}
}

//gets velocity value for direction/speed
public static int sign(double d)
{
	if (d <= 0)
	{
		return -1;
	}
	else
	{
		return 1;
	}
}

//keeps blockers in the window so they cannot go outside it
public static int possibleRange(int value, int min, int max)
```

```
        {
                return Math.min(Math.max(value, min), max);


        }
}
```

Class window:

```
package homework7;

import java.awt.BasicStroke;
import java.awt.Canvas;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Stroke;
import java.awt.image.BufferStrategy;

public class Game extends Canvas implements Runnable
{

	private static final long serialVersionUID = 1L;

	//to have a good 9:16 aspect ratio
	public static final int WIDTH = 1000;
	public static final int HEIGHT = WIDTH * 9/16;

	public boolean running = false;
	private Thread gameThread;
```

```
private Ball ball;
private Blocker blocker1;
private Blocker blocker2;

//main...runs game
public static void main(String[] args)
{
	new Game();
}

//game constructor
public Game()
{
	//set up window for game
	canvasSetup();
	//brings in classes for game
	initialize();

	new Window("Pong", this);

	this.addKeyListener(new UserInput(blocker1, blocker2));

	this.setFocusable(true);
}

private void initialize()
{
	//initialize ball
```

```
        ball = new Ball();
        //initialize blocker
        blocker1 = new Blocker(Color.BLUE, true);
        blocker2 = new Blocker(Color.RED, false);
}

private void canvasSetup()
{
        this.setPreferredSize(new Dimension(WIDTH, HEIGHT));
        this.setMaximumSize(new Dimension(WIDTH, HEIGHT));
        this.setMinimumSize(new Dimension(WIDTH, HEIGHT));
}

@Override
public void run()
{
        this.requestFocus();
        long lastTime = System.nanoTime(); //popular game loop from stackoverflow
        double amountOfTicks = 60.0;
        double ns = 1000000000 / amountOfTicks;
        double delta = 0;
        long timer = System.currentTimeMillis();
        //int frames = 0;
        while (running)
        {
                long now = System.nanoTime();
                delta += (now - lastTime) / ns;
                lastTime = now;
                while (delta >= 1)
```

```
                {
                        update();
                        delta--;
                }
                if (running)
                {
                        draw();
                        //frames++;
                }
                if (System.currentTimeMillis() - timer > 1000)
                {
                        timer += 1000;
                        //System.out.println("FPS: "  + frames); count frames
                        //frames = 0;
                }
        }
        stop();
}

private void draw()
{
        //initializing the drawing tools
        //buffer strategy found online
        //used for drawing to the screen with buffer.show
        BufferStrategy buffer = this.getBufferStrategy();

        if (buffer == null)
        {
                this.createBufferStrategy(3);
```

```
                return;
            }

            Graphics g = buffer.getDrawGraphics();

            //draw background
            drawBackground(g);

            //draw blockers and score
            blocker1.draw(g);
            blocker2.draw(g);
            //draw ball
            ball.draw(g);
            //draw
            g.dispose();
            buffer.show();
        }

        private void drawBackground(Graphics g)
        {
            //black back
            g.setColor(Color.black);
            g.fillRect(0, 0, WIDTH, HEIGHT);

            //white dotted line
            g.setColor(Color.white);
            Graphics2D g2d= (Graphics2D) g;
            Stroke dashedLine = new BasicStroke(3, BasicStroke.CAP_BUTT,
BasicStroke.JOIN_BEVEL, 0, new float[] {10}, 0);
```

```
            g2d.setStroke(dashedLine);
            g2d.drawLine(WIDTH / 2, 0, WIDTH / 2, HEIGHT);
    }

    public void drawBackgroundGameOver(Graphics g)
    {
            String gameOverString = "Game Over :(";
            //black back
            g.setColor(Color.black);
            g.fillRect(0, 0, WIDTH, HEIGHT);
            Font myFontGameOver = new Font("Serif", Font.BOLD, 60);
            g.setFont(myFontGameOver);
            g.setColor(Color.WHITE);
            g.drawString(gameOverString, 100, 400);
    }

    private void update()
    {
            //update ball
            ball.update(blocker1, blocker2);
            //update blockers
            blocker1.update(ball);
            blocker2.update(ball);
    }

    public void start()
    {
            gameThread = new Thread(this);
            gameThread.start();
```

```
		running = true;
	}

	public void stop()
	{
		try
		{
			gameThread.join();
			running = false;
		} catch (InterruptedException e)
		{
			e.printStackTrace();
		}
	}

	//gets velocity value for direction/speed
	public static int sign(double d)
	{
		if (d <= 0)
		{
			return -1;
		}
		else
		{
			return 1;
		}
	}

	//keeps blockers in the window so they cannot go outside it
```

```
	public static int possibleRange(int value, int min, int max)
	{
		return Math.min(Math.max(value, min), max);

	}
}
```

Class userInput:

```
package homework7;

import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;

public class UserInput extends KeyAdapter
{
	private Blocker blocker1;
	private boolean up1 = false;
	private boolean down1 = false;

	private Blocker blocker2;
	private boolean up2 = false;
	private boolean down2 = false;

	public UserInput(Blocker b1, Blocker b2)
	{
		blocker1 = b1;
		blocker2 = b2;
	}

	//get the keys pressed from user
	public void keyPressed(KeyEvent e)
	{
```

```
        int key = e.getKeyCode();

        //1 == down direction while -1 == up
        //controls with arrows // Right player
        if (key == KeyEvent.VK_UP)
        {
                blocker2.switchDirection(-1);
                up2 = true;
        }
        if (key == KeyEvent.VK_DOWN)
        {
                blocker2.switchDirection(1);
                down2 = true;
        }

        //controls with letters // Left player
        if (key == KeyEvent.VK_W)
        {
                blocker1.switchDirection(-1);
                up1 = true;
        }
        if (key == KeyEvent.VK_S)
        {
                blocker1.switchDirection(1);
                down1 = true;
        }
}

//key released from user
```

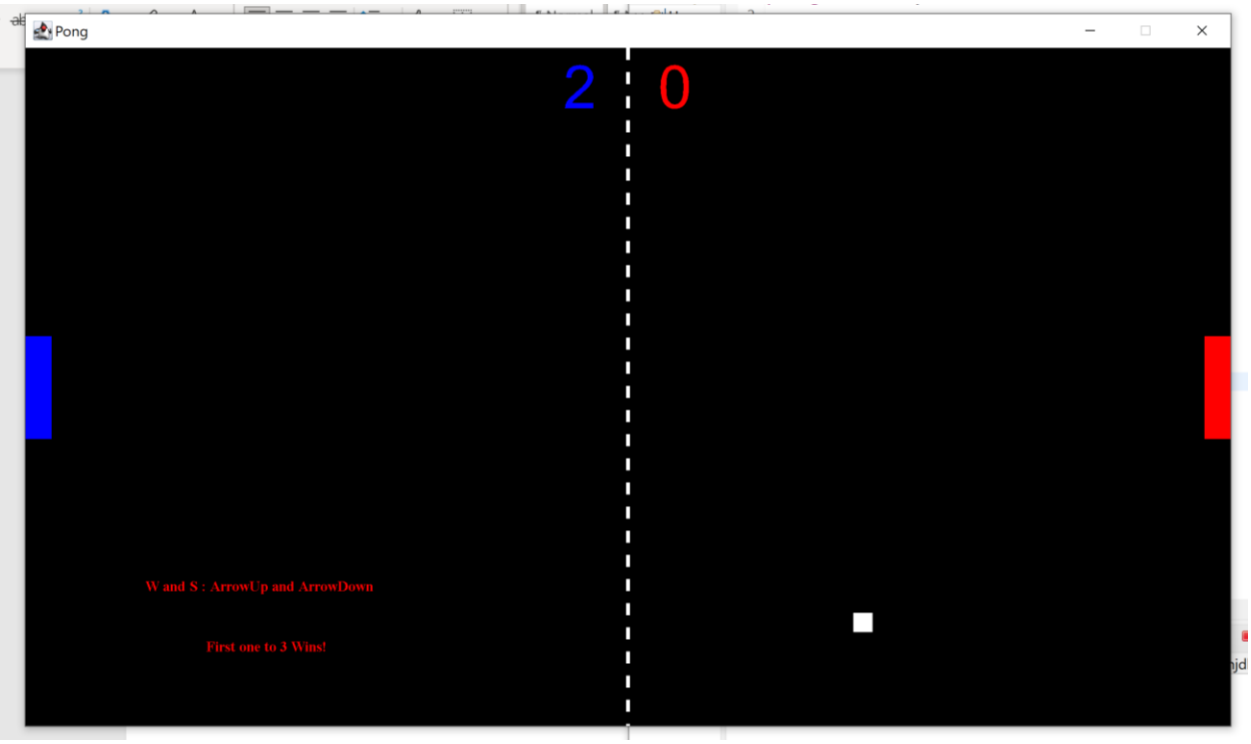
```
public void keyReleased(KeyEvent e)
{
	int key = e.getKeyCode();

	if (key == KeyEvent.VK_UP)
	{
		up2 = false;
	}
	if (key == KeyEvent.VK_DOWN)
	{
		down2 = false;
	}

	if (key == KeyEvent.VK_W)
	{
		up1 = false;
	}
	if (key == KeyEvent.VK_S)
	{
		down1 = false;
	}

	//magic to not lag otherwise game is very hard to play
	if (!up1 && !down1)
	{
		blocker1.stop();
	}
	if (!up2 && !down2)
	{
```

```
                    blocker2.stop();
                }
            }
        }
```


Pong
2
0
W and S : ArrowUp and ArrowDown
First one to 3 Wins!