AIR HOCKEY GAME

Submitted

For

Partial fulfilment of requirements for the

JAVA PROGRAMMING LAB

course of

II year B.Tech (AI and DS) program

Submitted by

Deepak Srinivas
21071A7217
Sai Rohith
21071A7218
Vinay Kalayan Gajula
21071A7219
Godavarthi Akanksha
21071A7220

Under the Guidance of
Mrs. E. Lalitha

Assistant professor & internal guide
Department of CSE-CYS,DS,AI&DS

VNR Vignana Jyothi Institute of Engineering and Technology – Hyderabad

Department of CSE-CYS,DS,AI&DS

2022-23

DECLARATION

This is to certify that the project work entitled **"Air Hockey game"** Submitted to VNR Vignana Jyothi Institute of Engineering & Technology in partial fulfilment of requirement for the award of Bachelor of Technology in Artificial Intelligence and Data Science. It is a bonafied report of the work carried out by us under the guidance and supervision of Mrs E. Lalitha(Assistant Professor), Department of CSE-CYS,VNRVJIET. To the best of our knowledge, this report has not been submitted in any form of any university or institution for the award of any degree
or diploma

Deepak Srinivas	Sai Rohith	Vinay Kalayan Gajula	Godavarthi Akanksha
21071A7217	21071A7218	21071A7219	21071A7220

IIB.Tech-CSE-AI&DS

VNRVJIET

ACKNOWLEDGEMENT

Behind every achievement lies the heartfelt gratitude to those who activiate in completing the project. To them we lay the words of gratitude imprinting within us.

We are indebted to our venerable principal Dr.C.D Naidu for this unflinching devotion, which lead us to complete the project. The support, encouragement given by him and his motivation lead us to complete this project.

We Express our sincere thanks to internal guide **Mrs. E. Lalitha** and also head of the department Dr.M. Raja shekar for having provided us a lot of facilities to undertake the project work and guide us to complete the project.

We take the opportunity to express our thanks to our faculty of our college VNR Vignana Jyothi Institute of Engineering and Technology who extended their valuable support in helping us to complete the project in time.

Deepak Srinivas Sai Rohith Vinay Kalayan Gajula Godavarthi Akanksha

21071A7217 21071A7218 21071A7219 21071A7220

II B.Tech-CSE-

AI&DS VNRVJIET

ABSTRACT

This project aims to develop an Air Hockey game using Java programming language. The game will feature a 2D playing feld where players can use their paddles to hit the puck and score points against their opponent. The game will have diferent levels of difculty,providing a challenging experience for players of all skill levels.

The project will utilize fundamental Java programming concepts such as object-oriented programming, GUI design, and event handling to create a user-friendly gaming experience.The goal of this project is to get hands-on experience in developing a computer game usingJava and have a deeper understanding of Java programming to create a functional gamethat can be played on any device.

Contents

Abstract
1. Intro into Java Swing Library
2. System specifications
3. Code
4. Snapshots
5. Conclusion

The Java Swing Library

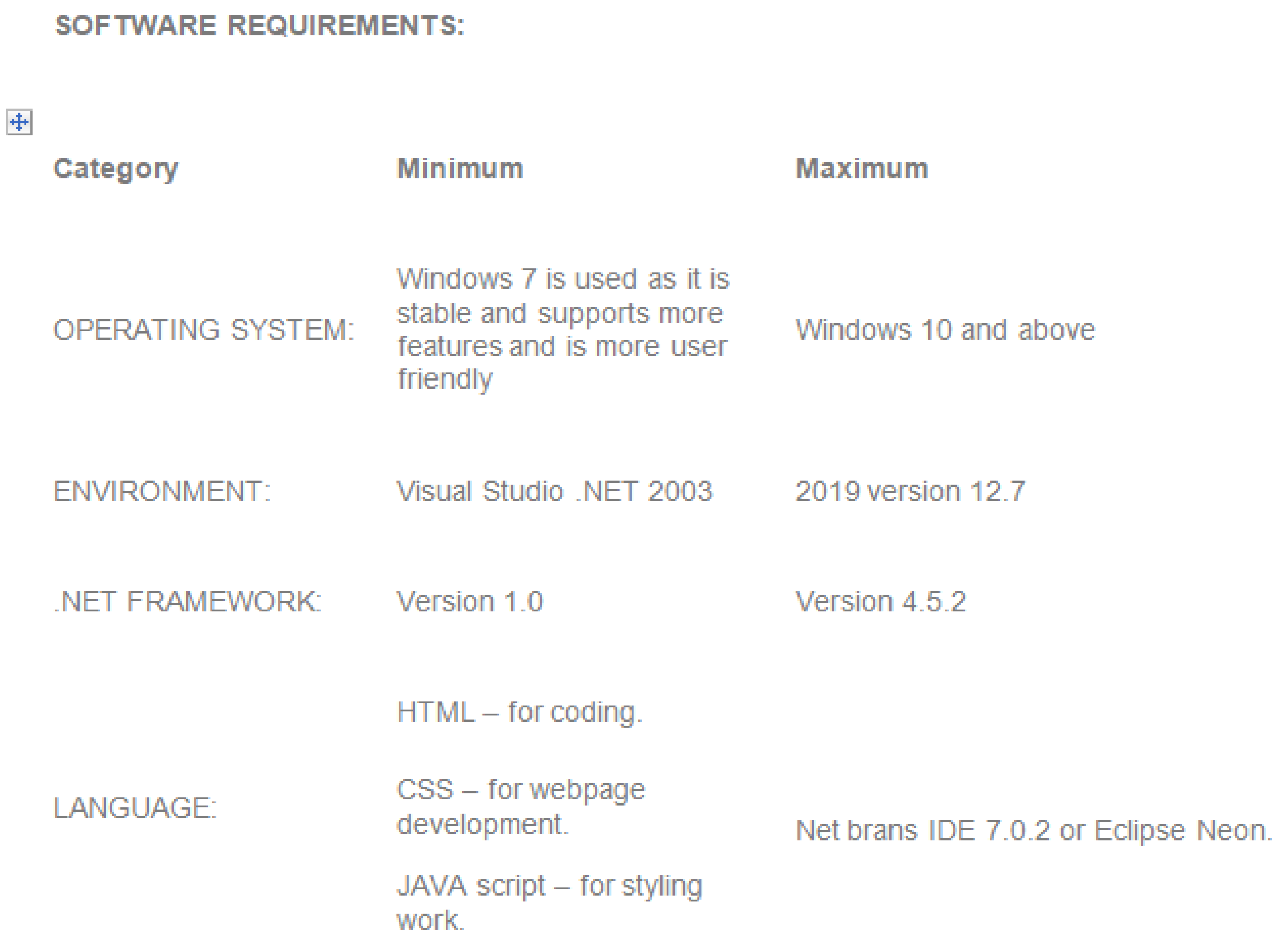
Swing has about four times the number of User Interface [UI] components as AWT and is part of the standard Java distribution. By today's application GUI requirements, AWT is a limited implementation, not quite capable of providing the components required for developing complex GUI's required in modern commercial applications. The AWT component set has quite a few bugs and really does take up a lot of system resources when compared to equivalent Swing resources. Netscape introduced its Internet Foundation Classes [IFC] library for use with Java. Its Classes became very popular with programmers creating GUI's for commercial applications.

- Swing is a Set Of API (API- Set Of Classes and Interfaces)
- Swing is Provided to Design Graphical User Interfaces
- Swing is an Extension library to the AWT (Abstract Window Toolkit)
- Includes New and improved Components that have been enhancing the looks and Functionality of GUIs'
- Swing can be used to build(Develop) The Standalone swing GUI Apps Also as Servlets And Applets
- It Employs model/view design architecture
- Swing is more portable and more flexible than AWT, The Swing is built on top of the AWT
- Swing is Entirely written in Java
- Java Swing Components are Platform-independent And The Swing Components are lightweight
- Swing Supports a Pluggable look and feels And Swing provides more powerful components
- such as tables, lists, Scrollpanes, Colourchooser, tabbedpane, etc
- Further Swing Follows MVC.

Many programmers think that JFC and Swing are one and the same thing, but that is not so.
JFC contains Swing [A UI component package] and quite a number of other items:

- Cut and paste: Clipboard support
- Accessibility features: Aimed at developing GUI's for users with disabilities
- The Desktop Colors Features Has been Firstly introduced in Java 1.1
- Java 2D: it has Improved colors, images, and also texts support

System Specifications



SOFTWARE REQUIREMENTS:

Category	**Minimum**	**Maximum**
OPERATING SYSTEM:	Windows 7 is used as it is stable and supports more features and is more user friendly	Windows 10 and above
ENVIRONMENT:	Visual Studio .NET 2003	2019 version 12.7
.NET FRAMEWORK:	Version 1.0	Version 4.5.2
LANGUAGE:	HTML – for coding. CSS – for webpage development. JAVA script – for styling work.	Net brans IDE 7.0.2 or Eclipse Neon.

Programs

Main

```
package airHockey;

import java.awt.GridLayout;
import java.awt.event.*;

import javax.swing.*;

import startUp.*;

public class AirHockey extends JDialog {
 private static final long serialVersionUID = 1L;

 private MouseListener mouse = new MouseAdapter() {
  @Override
  public void mouseEntered(MouseEvent e) {
   // TODO when make nicer graphics use the following code to highlight
   // whichever button
   // the mouse is hovering over
   // JButton button = (JButton) e.getSource();
   // button.setBorder(border);
  }

  @Override
  public void mouseExited(MouseEvent e) {
   // JButton button = (JButton) e.getSource();
   // button.setBorder(null);
  }
 };

 public AirHockey() {
  setSize(150, 125);
  setUndecorated(true);
  setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
  setLocationRelativeTo(null);
  setResizable(false);
  setLayout(new GridLayout(3, 1));

  // create menu option to choose if server, client, or testing so want both
  JButton server = new JButton("SERVER");
  server.addMouseListener(mouse);
  server.addActionListener(new ServerListener(this));
  add(server);

  JButton client = new JButton("CLIENT");
  client.addMouseListener(mouse);
  client.addActionListener(new ClientListener(this));
  add(client);

  // creates both a server and client using localhost
  JButton testMode = new JButton("TEST MODE");
  testMode.addMouseListener(mouse);
  testMode.addActionListener(new TestModeListener(this));
  add(testMode);

  setVisible(true);
 }


}
```

```
 public static void main(String[] args) {
  try {
  UIManager.setLookAndFeel(UIManager.getCrossPlatformLookAndFeelClassName());
  /* LookAndFeel lat = UIManager.getLookAndFeel();
  * UIDefaults defaults = lat.getDefaults(); defaults.replace(key, value);
  * for(Object key: UIManager.getLookAndFeel().getDefaults().keySet()) {
  * System.out.println(key + " = " + UIManager.get(key));
  * } */
  }
  catch (ClassNotFoundException | InstantiationException | IllegalAccessException | UnsupportedLookAndFeelException e) {
  e.printStackTrace();
  }
  new AirHockey();
  }
```

in-game chat

```
package airHockey;

import java.awt.BorderLayout;
import java.awt.Dimension;
import java.io.IOException;

import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.border.EtchedBorder;

public class Chat extends JPanel {
  private static final long serialVersionUID = 1L;
  private JTextArea area;
  private JTextField text;

  public Chat() {
    setLayout(new BorderLayout());
    setPreferredSize(new Dimension(World.GAMEWIDTH, World.FRAMEHEIGHT));

    area = new JTextArea("Welcome to chat!\n");
    area.setEditable(false);
    area.setLineWrap(true);
    JScrollPane pane = new JScrollPane(area);
    add(pane, BorderLayout.CENTER);

    text = new JTextField();
    text.setBorder(new EtchedBorder());
    add(text, BorderLayout.SOUTH);
  }

  protected String readText() throws IOException {
    String line = text.getText();
    area.append("Me: " + line + "\n");

    text.setText("");
    return line;
  }

  protected void updateChat(String msg) {
    area.append("Opponent: " + msg + "\n");
  }

}
```

Client World

```
package airHockey;

import java.io.IOException;
import java.net.Socket;

import javax.sound.sampled.LineUnavailableException;
import javax.sound.sampled.UnsupportedAudioFileException;

public class ClientWorld extends World {
  private static final long serialVersionUID = 1L;

  public ClientWorld(String serverAddress) throws IOException, LineUnavailableException,
UnsupportedAudioFileException {
    setLocationRelativeTo(null);
    socket = new Socket(serverAddress, 3769);
    setUp(1); // 1 = server second
  }
}
```

Game Threads

```
package airHockey;

import java.io.IOException;

import javax.sound.sampled.LineUnavailableException;
import javax.sound.sampled.UnsupportedAudioFileException;

public class GameLoopThread extends Thread {
  private World world;
  //private long lastExecutedTime;

  public GameLoopThread(World world) {
    this.world = world;
  }

  @Override
  public void run() {
    while (true) {
      //long newTime = System.currentTimeMillis();
      //double delta = (newTime - lastExecutedTime) / 1000;
      // int speed = (int) (world.getPuckSpeed() * delta);
      int speed = world.getPuckSpeed();
      if (speed > 0) {
        try {
          world.movePuck();
          world.syncPuck();
        }
        catch (LineUnavailableException | IOException | UnsupportedAudioFileException | InterruptedException e) {
          e.printStackTrace();
        }
      }
      try {
        // sleep(16);
        sleep(21 - speed);
      }
      catch (InterruptedException e) {
        e.printStackTrace();
      }
      //lastExecutedTime = newTime;
    }

  }
}
```

Mallets

```
package airHockey;

import java.awt.Graphics;
import java.awt.MouseInfo;
import java.awt.Point;
import java.io.IOException;

import javax.imageio.ImageIO;

import commands.MalletCommand;

public class Mallet extends Positionable {
  protected static final int MALLETRADIUS = 20;

  public Mallet(int sideCenter) throws IOException {
    posX = World.GAMEWIDTH / 2;
    posY = sideCenter;
    image = ImageIO.read(getClass().getResource("pics/mallet.jpg"));
  }

  public void setMalletXY(double x, double y) {
    Point point = MouseInfo.getPointerInfo().getLocation();
    if (point.getY() - y - (MALLETRADIUS * 2) >= World.FRAMEHEIGHT / 2) {
      posX = point.getX() - x;
      posY = point.getY() - y - MALLETRADIUS * 2;
    }
  }

  @Override
  protected void draw(Graphics g) {
    g.drawImage(image, (int) posX - MALLETRADIUS, (int) posY - MALLETRADIUS, MALLETRADIUS * 2, MALLETRADIUS * 2, null);
  }

  @Override
  public MalletCommand getCommand() {
    return new MalletCommand(posX, posY);
  }
}
```

Mallet Movements

```
package airHockey;

import java.awt.Point;
import java.awt.event.MouseEvent;
import java.awt.event.MouseMotionAdapter;
import java.io.IOException;

import javax.sound.sampled.LineUnavailableException;
import javax.sound.sampled.UnsupportedAudioFileException;

public class MalletMotionListener extends MouseMotionAdapter {
  private World world;

  public MalletMotionListener(World world) {
    this.world = world;
  }

  // mallet moves with mouse
  @Override
  public void mouseMoved(MouseEvent e) {
    // move your mallet to wherever the mouse pointer is located
    Point point = world.getLocation();
    try {
      world.moveMallet(point.getX(), point.getY());

      // send location of your mallet to second players
      world.sendCommand(world.table.getCommand('m'));
    }
    catch (IOException | LineUnavailableException | UnsupportedAudioFileException | InterruptedException e1) {
      e1.printStackTrace();
    }

    world.repaint();
  }
}
```

Positioning

```
package airHockey;

import java.awt.Graphics;
import java.awt.Image;

import commands.Command;

// both Mallet and Puck extend Positionable
public abstract class Positionable {
  protected double posX;
  protected double posY;

  //FIXME had problems serializing since image is not serializable
  protected Image image;

  protected void updateCoordinates(double x, double y) {
    posX = x;
    posY = y;
  }

  protected abstract void draw(Graphics g);

  protected abstract Command getCommand();
}
```

Reader Listener

```
package airHockey;

import java.net.Socket;

import commands.Command;

public interface ReaderListener {

  void onObjectRead(Command command);

  void onCloseSocket(Socket socket);

}
```

Reader Thread

```
public class ReaderThread extends Thread {
  private Socket socket;
  private ReaderListener listener;

  public ReaderThread(Socket socket, ReaderListener listener) {
    this.socket = socket;
    this.listener = listener;
  }

  @Override
  public void run() {
    try {
      InputStream in = socket.getInputStream();
      ObjectInputStream objIn = new ObjectInputStream(in);

      while (true) {
        Command command = (Command) objIn.readObject();
        listener.onObjectRead(command);
      }
      // in.close();
      // onjIn.close();
    }
    catch (ClassNotFoundException | IOException e) {
      e.printStackTrace();
    }
    listener.onCloseSocket(socket);
  }
}
```

Puck/Ball

```
package airHockey;

import java.awt.Color;
import java.awt.Graphics;
import java.io.IOException;
import java.util.concurrent.Executors;
import java.util.concurrent.ScheduledExecutorService;
import java.util.concurrent.TimeUnit;

import javax.sound.sampled.LineUnavailableException;
import javax.sound.sampled.UnsupportedAudioFileException;

import commands.PuckCommand;

public class Puck extends Positionable {
  protected static final int PUCKRADIUS = 12;
  private int width = World.GAMEWIDTH;
  private int height = World.FRAMEHEIGHT;
  private int speed;
  private float colorNum;
  private int resety;

  // the current slope the puck is moving in
  private double deltaX;
  private double deltaY;

  private boolean goal;
  private ScheduledExecutorService executor;
  private int time;

  private Runnable timer = new Runnable() {
    public void run() {
      time--;
      if (time == 0) {
        goal = false;
      }
    }
  };

  public Puck() throws IOException {
    // image = ImageIO.read(getClass().getResource("pics/puck.jpg"));
    posX = width / 2;
    posY = height / 2;
    resety = height / 4;
    speed = 0;
    colorNum = 0;
    executor = Executors.newScheduledThreadPool(1);
  }

  protected int move() throws LineUnavailableException, IOException,
UnsupportedAudioFileException {
    posX += deltaX;
    posY += deltaY;

    // if hit side wall
    if (posX - PUCKRADIUS <= 4 || posX + PUCKRADIUS >= width - 4) {
      colorNum += .02;
      deltaX = -deltaX;
      decreaseSpeed();
      changeColor();
    }
```

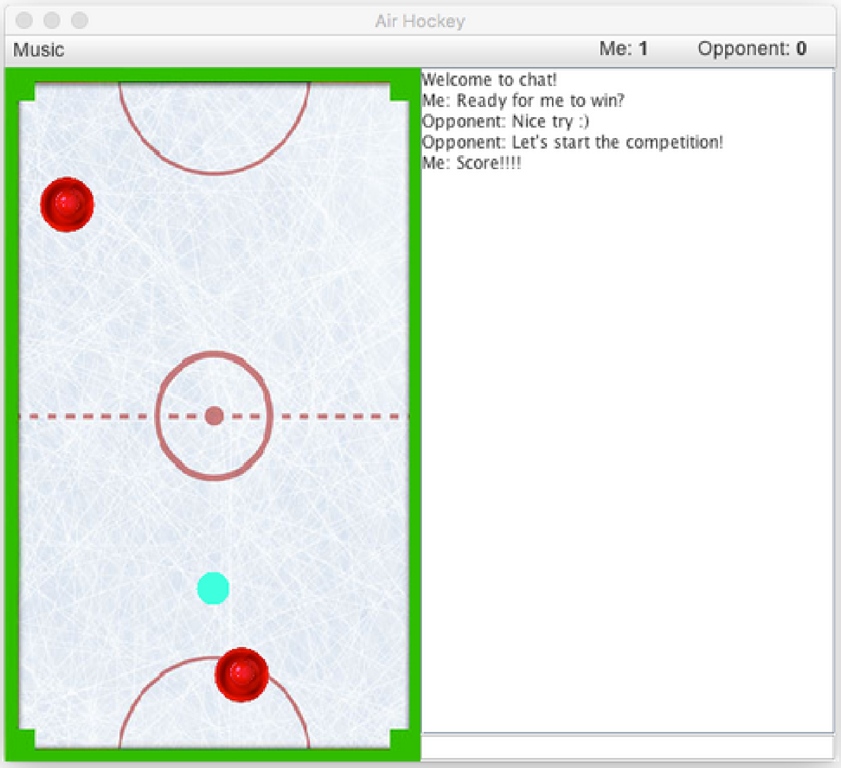
```
// if hit side wall
  if (posX - PUCKRADIUS <= 4 || posX + PUCKRADIUS >= width - 4) {
   colorNum += .02;
   deltaX = -deltaX;
   decreaseSpeed();
   changeColor();
  }

  // hit top/bottom walls
  // if the puck hit a side within the goal range, returns the player who
  // scored a point
  // otherwise returns 0
  else {
   if (posY - PUCKRADIUS <= 4) {
    return checkGoal(1);
   }
   else if (posY + PUCKRADIUS >= height - 4) {
    return checkGoal(2);
   }
  }
  return 0;
 }

 // if there is a goal, this method will return the player that called checkGoal, otherwise, will return 0
 private int checkGoal(int player) throws LineUnavailableException, IOException, UnsupportedAudioFileException {
  // if puck within goal range, return player who scores
  if (posX > 70 && posX < width - 70) {
   time = 2;
   // use executor to ensure that goal shows for right amount of time
   executor.scheduleAtFixedRate(timer, 0, 4, TimeUnit.SECONDS);
   reset();
   // TODO sound.changeTrack("sound/goal.wav");
   goal = true;
   return player;
  }

  // otherwise bounce off wall
  deltaY = -deltaY;
  decreaseSpeed();
  changeColor();
  return 0;
 }

 private void reset() {
  posX = width / 2;
  if (resety == (int) height / 4) {
   resety *= 3;
  }
  else {
   resety = height / 4;
  }
  posY = resety;
  speed = 0;
 }

 protected void setSlope(double malletX, double malletY) {
  deltaY = posY - malletY;
  deltaX = posX - malletX;

  // keep track if original x or y was negative so know which end direction should be negative
  // otherwise 2 negatives will just cancel out or don't know if x or y was negative
  int xneg = 1;
  int yneg = 1;
  if (deltaX < 0) {
   xneg = -1;
  }
```

```
if (deltaY < 0) {
yneg = -1;
}
deltaX = Math.abs(deltaX);
deltaY = Math.abs(deltaY);
if (deltaX != 0 && deltaY != 0) {
// calculate the inverse tangent of the slope
double angle1 = Math.atan(deltaX / deltaY);
double angle2 = 90 - angle1;
deltaY = Math.sin(angle2) * yneg;
deltaX = Math.sin(angle1) * xneg;
}
else if (deltaX == 0) {
deltaY = yneg;
}
else {
deltaX = xneg;
}
}

protected void decreaseSpeed() {
speed--;
}

protected int getSpeed() {
return speed;
}

public void changeColor() {
colorNum += .02;
}

protected boolean getGoal() {
return goal;
}

protected void setResetY(int num) {
resety = height / 4 * num;
}

protected void hit() {
changeColor();
speed = 20;
}

protected boolean isMoving() {
return speed > 0;
}

protected void update(double x, double y, int speed) {
updateCoordinates(x, y);
this.speed = speed;
}

@Override
protected void draw(Graphics g) {
g.setColor(Color.getHSBColor(colorNum, 1, 1));
g.fillOval((int) (posX - PUCKRADIUS), (int) (posY - PUCKRADIUS), PUCKRADIUS * 2, PUCKRADIUS * 2);
}

@Override
protected PuckCommand getCommand() {
return new PuckCommand(posX, posY, speed);
}
}
```

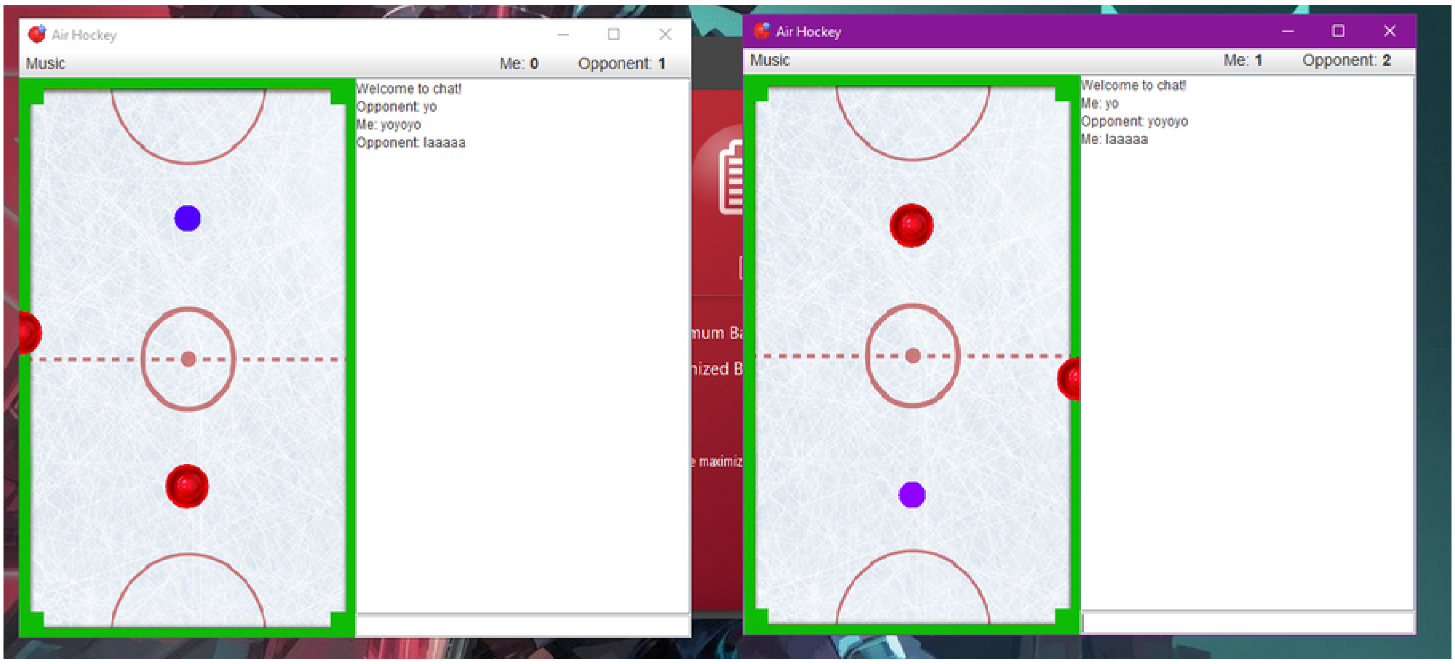
server world/layout

```
package airHockey;

import java.io.IOException;
import java.net.ServerSocket;

import javax.sound.sampled.LineUnavailableException;
import javax.sound.sampled.UnsupportedAudioFileException;

public class ServerWorld extends World {
  private static final long serialVersionUID = 1L;

  public ServerWorld(boolean test) throws IOException, LineUnavailableException,
UnsupportedAudioFileException {
    if (!test) {
      setLocationRelativeTo(null);
    }
    ServerSocket serverSocket = new ServerSocket(3769); // port num sent
    socket = serverSocket.accept();
    setUp(3); // 3 = server first
  }

  @Override
  public void syncPuck() throws IOException, InterruptedException {
    sendCommand(table.getCommand('p'));
  };
}
```

Snapshots



Welcome to chat!
Me: Ready for me to win?
Opponent: Nice try :)
Opponent: Let's start the competition!
Me: Score!!!!



Welcome to chat!
Opponent: yo
Me: yoyoyo
Opponent: laaaaa

Welcome to chat!
Me: yo
Opponent: yoyoyo
Me: laaaaa

Conclusion

Using Pure JAVA, we were able to build a game which is able to run on 2 different systems at the same time. This project could further be enhanced by adding a player database and their statistics via either MySQL or ORACLE DB.

This Application uses loads of MultiThreading, and is at a total size of 202 MB.

References:

- **Github**
- **Youtube**
- **java documentation**
- **freecodecamp**