~~Whack-A-Paul~~

Whack-A-Mole

Group IX

405850420 吳柏霆
405850206 鄒亞微
405850115 王薏婷
405850032 劉曉薇

Specifications

Programming language: JAVA
Developing environment: JAVA SE 10.0.1
Minimum screen resolution program required: 650 x 650 (px.)
Speaker required.
Mouse is suggested in order to get the best gaming experience.

Imported Classes within the project (Alphabetical order)

```
java.awt.Container;
java.awt.Cursor;
java.awt.Font;
java.awt.Graphics;
java.awt.Point;
java.awt.Toolkit;
java.awt.event.ActionEvent;
java.awt.event.ActionListener;
java.awt.event.MouseEvent;
java.awt.event.MouseListener;
java.awt.event.MouseMotionListener;
java.awt.event.WindowAdapter;
java.awt.event.WindowEvent;
java.awt.image.BufferedImage;
java.io.File;
java.io.IOException;
java.util.Random;
javax.imageio.ImageIO;
javax.sound.sampled.AudioFormat;
javax.sound.sampled.AudioInputStream;
javax.sound.sampled.AudioSystem;
javax.sound.sampled.DataLine;
javax.sound.sampled.SourceDataLine;
javax.swing.ImageIcon;
javax.swing.JButton;
javax.swing.JFrame;
javax.swing.JLabel;
javax.swing.Timer;
```

Properties and Methods

HitMouse.java	
Properties	
contentPane : Container	The panel where the GUI components rendered.
jButton_start : JButton	The GUI start button.
jButton_stop : JButton	The GUI stop button.
jLabel_time : JLabel	The label that shows the remaining time.
jLabel_score : JLabel	The label that shows the current score.
jLabel_copyright : JLabel	The label that shows the names of our group members.
random : Random	The variable that generates the random number for the mouse to show up next time.
hole : Hole[]	The 9 holes for the mouse to show up.
mode : int	Record the current hit status, hit or miss.
score : int	The variable is used to keep track of the current score.
time_left : double	The variable is used to keep track of the current time.
timer_refresh_screen : Timer	The countdown will invoke refresh screen periodically according to this timer.
cursor_normal : Cursor	The cursor is shown while the mouse is not pressed.
cursor_down : Cursor	The cursor is shown while the mouse is pressed.
image_normal : BufferedImage	The picture of the normal mouse.
image_hit : BufferedImage	The picture of the mouse being hit.
playSonud_hit : PlaySound	The sound effect of the mouse being hit.
playSonud_bg : PlaySound	The sound effect of the background music.

Methods	
HitMouse()	The constructor of the class. It will do some initialization while the object is generated.
paint(Graphics)	The method will render the JFrame and all other component stuffs up-to-date.
actionPerformed(ActionEvent)	This method is invoked by Listeners while the mouse events or the timer tick events are triggered.
mouseDragged(MouseEvent)	These methods are required due to the implementation of the MouseMotionListener and MouseListener, however, we do not use these methods in this project.
mouseMoved(MouseEvent)	
mouseEntered(MouseEvent)	
mouseExited(MouseEvent)	
mouseClicked(MouseEvent)	
mousePressed(MouseEvent)	The method is invoked when the mouse is pressed.
mouseReleased(MouseEvent)	The method is invoked when the mouse is released.
main(String[])	The enter point of this entire project.

Hole.java	
Properties	
x : int	X axis of the hole.
y : int	Y axis of the hole.

PlaySound.java	
Properties	
sourceDataLine : SourceDataLine	The pipeline of the speaker. Whenever a datum is sent to this pipeline, the speaker will process it, which is broadcast the sound.
audioInputStream : AudioInputStream	The pipeline which will undertake the input source, a sequence of data of the sound.
audioFormat : AudioFormat	The format of which the audioInputStream is going to input to the program.
dataLineInfo : Info	The variable that takes over the decoded data from audioInputStream with the format in audioFormat.
tempBuffer : byte[]	The temporary buffer between the dataLineInfo and the sourceDataLine. Data in here is waiting for being infused into sourceDataLine.
mySound : File	The variable linked to the sound file.
Methods	
PlaySound(String)	The constructor of the class. It will do some initialization while the object is generated.
run()	This method will invoke another thread to play the sound in order not to interfere the procedure in the main thread.
stopSound()	This method flushes the audioInputStream to empty in order to pause the sound.

Claims: Important designs in the project

The most outstanding part of our project that we are proud of is we put the **multi-thread** concept into practice. In the circumstance that we were adding the background sound into our project and played it when the game running, everything could not function harmoniously. When we focused on the fluency of the graphics, the background sound kept lagging. On the other hand, when we focused on the background sound, the mouse-click event listener worked awkwardly. Eventually, we decided to use multi-threading to dispatch the two functions to the two threads. Each thread can then focus on their main jobs intently. (First also the main thread focuses on the mouse-click event monitoring and GUI rendering. The second thread focuses on the playing the background.) And now, it works fine.

Results: Cases of Study

This design is used for stress relieving purposes. If you are very angry or stress due to any circumstances, this program has the very stress relieving factor so you do not go and punch a hole in your computer or laptop.

When given the chance to exercise on the program we made in action, we feel really stress relieved. On further studies, when students with extremely high stress such as big exams or project deadline are around the corner play the game, it is noted that their mood and emotions are calmed down.

We believe this small game is beneficial to all students who show the symptoms mentioned above. And we are proud to present the final rendition of our project that is: Whack-a-~~Paul~~ Mole.

Source Code

HitMouse.java

```
import java.awt.*;
import java.awt.event.*;
import java.awt.image.BufferedImage;
import javax.swing.*;
import javax.swing.Timer;
import javax.sound.sampled.*;
import javax.imageio.ImageIO;
import java.io.*;
import java.util.*;

public class HitMouse extends JFrame implements ActionListener, MouseMotionListener, MouseListener {

  Container contentPane = getContentPane();
  JButton jButton_start = new JButton("Start");
  JButton jButton_stop = new JButton("Stop");
  JLabel jLabel_time = new JLabel("<= Press Start to Play");
  JLabel jLabel_score = new JLabel("Score: 0");
  JLabel jLabel_copyright = new JLabel("By 吳柏霆 鄒亞微 王薏婷 劉曉薇");

  Random random = new Random();
  Hole[] hole = new Hole[10];
  int mode = 0;
  int score = 0;
  double time_left;
  Timer timer_refresh_screen = new javax.swing.Timer(500, this);

  Cursor cursor_normal = Toolkit.getDefaultToolkit().createCustomCursor(
                         new ImageIcon("cursor@150.png").getImage(), new Point(0, 0), "custom cursor");
  Cursor cursor_down = Toolkit.getDefaultToolkit().createCustomCursor(
                         new ImageIcon("cursor@150_d.png").getImage(), new Point(0, 0), "custom cursor");

  BufferedImage image_normal;
  BufferedImage image_hit;

  PlaySound playSonud_hit = new PlaySound("surprise.wav");
  PlaySound playSonud_bg;

  public HitMouse() throws Exception {

    super("Hit Mouse");
    setResizable(false);
    setSize(650, 650);
    setLocationRelativeTo(null);

    setCursor(cursor_normal);

    try {
      image_normal = ImageIO.read(new File("hhw.png"));
      image_hit = ImageIO.read(new File("hhw_hit.png"));
    } catch (Exception e) {
      System.out.println("Load Picture Fail");
    }

    playSonud_hit.start();

    hole[0] = new Hole();

    for (int i = 1; i <= 3; i++) {
      for (int j = 1; j <= 3; j++) {
        hole[(i - 1) * 3 + j] = new Hole();
        hole[(i - 1) * 3 + j].x = 200 + (j - 1) * 150;
        hole[(i - 1) * 3 + j].y = 200 + (i - 1) * 150;
      }
    }

    jButton_start.setBounds(30, 50, 80, 50);
    jButton_stop.setBounds(30, 120, 80, 50);

    jLabel_time.setBounds(150, 27, 300, 100);
    jLabel_time.setFont(new Font("Serif", Font.BOLD, 24));

    jLabel_score.setBounds(450, 27, 200, 100);
    jLabel_score.setFont(new Font("Serif", Font.BOLD, 24));

    jLabel_copyright.setBounds(290, 550, 500, 100);
    jLabel_copyright.setFont(new Font("Serif", Font.BOLD, 24));

    jButton_stop.setEnabled(false);
```

```
    contentPane.setLayout(null);
    contentPane.add(jButton_start);
    contentPane.add(jButton_stop);
    contentPane.add(jLabel_time);
    contentPane.add(jLabel_score);
    contentPane.add(jLabel_copyright);

    jButton_start.addActionListener(this);
    jButton_stop.addActionListener(this);
    addMouseMotionListener(this);
    addMouseListener(this);

    setVisible(true);
  }

  public void paint(Graphics g) {
    super.paint(g);

    for (int i = 1; i <= 9; i++) {
      if (i != mode % 10)
        g.drawOval(hole[i].x - 50, hole[i].y - 50, 100, 100);
    }

    if (mode > 10) {
      g.drawImage(image_hit, hole[mode % 10].x - 50, hole[mode % 10].y - 100, this);
      try {
        Thread.sleep(500);
      } catch (Exception e) {
      }
    } else if (mode != 0)
      g.drawImage(image_normal, hole[mode].x - 50, hole[mode].y - 100, this);
  }

  public void actionPerformed(ActionEvent e) {

    if (e.getSource() == timer_refresh_screen) {
      time_left -= 0.5;
      mode = random.nextInt(10);
      jLabel_time.setText("Time: " + (int) time_left + " sec(s)");
      repaint();
    } else if (e.getSource() == jButton_start) {
      try {
        jButton_start.setEnabled(false);
        jButton_stop.setEnabled(true);
        playSonud_bg = new PlaySound("bg.wav");
        Thread.sleep(50);
        playSonud_bg.start();
        Thread.sleep(50);
        synchronized (playSonud_bg) {
          playSonud_bg.notify();
        }
      } catch (Exception e1) {
      }
      time_left = 60;
      score = 0;
      timer_refresh_screen.start();
      return;
    } else if (e.getSource() == jButton_stop) {
      time_left = -1;
    }

    if (time_left <= 0) {
      playSonud_bg.stopSound();
      jButton_start.setEnabled(true);
      jButton_stop.setEnabled(false);
      timer_refresh_screen.stop();
      jLabel_time.setText("<= Press Start to Play");
      mode = 0;
      repaint();
    }
  }

  public void mouseDragged(MouseEvent e) {
  };

  public void mouseMoved(MouseEvent e) {
  };

  public void mouseEntered(MouseEvent e) {
  };

  public void mouseExited(MouseEvent e) {
  };
```

```
  public void mousePressed(MouseEvent e) {
    setCursor(cursor_down);

    if ((hole[mode % 10].x - e.getX()) * (hole[mode % 10].x - e.getX())
        + (hole[mode % 10].y - e.getY()) * (hole[mode % 10].y - e.getY()) < 3000) {

      synchronized (playSonud_hit) {
        playSonud_hit.notify();
      }

      score++;
      jLabel_score.setText("Score: " + score);
      mode = mode * 10 + mode;
      repaint();
    }
  };

  public void mouseReleased(MouseEvent e) {
    setCursor(cursor_normal);
  };

  public void mouseClicked(MouseEvent e) {
  };

  public static void main(String args[]) throws Exception {
    HitMouse app = new HitMouse();
    app.addWindowListener(new WindowAdapter() {
      public void windowClosing(WindowEvent e) {
        System.exit(0);
      }
    });
  }
}
```

Hole.java

```
class Hole {
  int x;
  int y;
}
```

PlaySound.java

```
class PlaySound extends Thread {

  SourceDataLine sourceDataLine;
  AudioInputStream audioInputStream;
  AudioFormat audioFormat;
  DataLine.Info dataLineInfo;
  byte tempBuffer[] = new byte[32];
  File mySound;

  PlaySound(String mediaFileName) throws Exception {
    mySound = new File(mediaFileName);
    audioFormat = AudioSystem.getAudioInputStream(mySound).getFormat();
    dataLineInfo = new DataLine.Info(SourceDataLine.class, audioFormat, AudioSystem.NOT_SPECIFIED);
    sourceDataLine = (SourceDataLine) AudioSystem.getLine(dataLineInfo);
    audioFormat = new AudioFormat(AudioFormat.Encoding.PCM_SIGNED, audioFormat.getSampleRate(), 16,
        audioFormat.getChannels(), audioFormat.getChannels() * 2, audioFormat.getSampleRate(), false);
  }

  synchronized public void run() {
    while (true) {
      try {

        audioInputStream = AudioSystem.getAudioInputStream(mySound);

        // Open output device
        sourceDataLine.open(audioFormat);

        // Decode sound file to InputStream
        if (audioFormat.getEncoding() != AudioFormat.Encoding.PCM_SIGNED)
          audioInputStream = AudioSystem.getAudioInputStream(audioFormat, audioInputStream);

        sourceDataLine.start();

        wait();

        // Read InputStream and output to OutputStream
        int count;
        while ((count = audioInputStream.read(tempBuffer, 0, tempBuffer.length)) > 0)
          sourceDataLine.write(tempBuffer, 0, count);
```

```
        // Block until all the data is output
        sourceDataLine.drain();

      } catch (Exception e) {
      }
    }
  }

  void stopSound() {
    try {
      audioInputStream.skip(audioInputStream.available());
    } catch (IOException e) {
    }
  }
}
```

Conclusion: Thoughts of the Project (one page per member)