Interacción Hombre-Computador
Facultad de Ingeniería en Ciencias de la Computación y Telecomunicaciones
Universidad Autónoma Gabriel René Moreno

Componentes Visuales en Java

Añez Vladimirovna Leonardo Henry
Ingeniería en Informática

DT	HI	HG	AE
0	12	0	100

Universidad Autónoma
Gabriel René Moreno

11 de febrero de 2020
Santa Cruz de la Sierra, Bolivia

Índice general

Introducción

Este trabajo comprende proyectos en el que la Interacción Hombre-Computador es el principal actor. Veremos el uso de los componentes graficos que ofrece el lenguaje de programacion `Java` utilizando principios de la programacion orientada a objetos, programación grafica y algoritmos aplicados en materias previas. Se trata de exponer la relación entre el usuario y la computadora, y como los programas desarollados deben de ser pensados de tal manera que la UX sea lo mas intuitivamente posible.

1. Novela Visual: 人間のコンピューター Interaction

1.1. Introducción

1.1.1. Definición

Una novela visual, a veces abreviada como VN (en inglés), es un género de juego interactivo, que se originó en Japón, presentando una historia basada en texto con estilo narrativo de literatura e interactividad ayudada por imágenes estáticas o basadas en sprites, la mayoría de las veces usando arte estilo anime u ocasionalmente imágenes fijas de acción en vivo (y a veces secuencias de video). Como su nombre podría sugerir, se parecen a las novelas de medios mixtos.

Si bien su descripción no es inexacta de ninguna manera (al menos para los juegos que describe), no captura la imagen completa. Las novelas visuales no son un objeto estático. Han cambiado mucho desde su inicio, por lo que limitarlos a una sola definición corre el riesgo de pasar por alto importantes acontecimientos históricos.

1.2. Antecedentes

1.2.1. Los primeros años: juegos de aventura y manga

Cualquier historia de novelas visuales debe comenzar con: *"The Portopia Serial Murder Case"* [2]. Yuji Horii (el diseñador del juego) en una entrevista con Retro Gamer, dice que los juegos de aventuras eran el género dominante para la narrativa en los juegos de la época. Así que creó Portopia como un intento de presentar juegos de aventuras estadounidenses al público japonés. Es por eso que el juego se inspira en juegos como *"King's Quest y Sam's Spade"* (y tal vez *"Mystery House"*).

Debido a que los juegos de aventura eran básicamente una serie de acertijos para que el jugador los resolviera, su diseño requería un enfoque holístico en el que cada aspecto del juego funcionara hacia el mismo

Figura 1.1: Portada de Portopia de 1983

fin. La narrativa, por ejemplo, tuvo que contextualizar los acertijos que el jugador encontró, definiendo la lógica de una manera lo suficientemente consistente como para que el jugador pudiera resolverlos razonablemente.

Las novelas visuales se desvanecerían lentamente en un diseño orientado a los rompecabezas a favor de la narrativa, todas sus otras estructuras permanecieron intactas, incluidas sus imágenes. Entonces, hasta ese momento, las primeras novelas visuales emplearon en gran medida una estrategia que evitó los sprites por completo. Contaban sus historias no a través de una colección de activos artísticos que se construyeron independientemente uno del otro, sino a través de paneles individuales. Al igual que el manga, cada panel se compuso en su conjunto, sus ángulos y formas y otros elementos visuales se unieron para crear una escena muy específica o transmitir un estado de ánimo particular.

1.2.2. Simuladores de citas y narración teatral

A pesar de que los simuladores de citas y las novelas visuales son términos distintos, es difícil negar la influencia que el primero ha tenido en el segundo. Parte de esto puede deberse a la renuencia a un género con fuertes raíces pornográficas. Sin embargo, también puede tener algo que ver con los simuladores de citas que carecen de una historia central. En estas categorias, los dos exponentes mas grandes son *"No Ri Ko"* y *"Tokimeki Memorial"*.

Figura 1.2: Tokimeki Memorial, 1994

El segundo, Tokimeki Memorial [5] no solo popularizó el género, sino que también se convirtió en el estándar que los futuros simuladores de citas buscarían. Algunas de sus influencias incluyen:

- Un enfoque en poner en primer plano escenarios y eventos comunes y relacionables sobre escenarios fantásticos.
- La idea de gestionar una colección de relaciones románticas a través de la actividad diaria.
- El juego que consiste en elecciones explícitas, como dónde ir o qué decir de un conjunto de opciones dadas.
- Esas elecciones que determinan con qué chica se vincula románticamente tu personaje y, por lo tanto, qué hilo narrativo sigue el jugador.

Estas características y otras son las que con el paso del tiempo se han mantenido como escencia en las VNs modernas, tal es el caso de: Clannad, Steins;Gate, Fate/stay night, Katawa Shoujo, Doki Doki Literature Club, Nekopara, por citar algunas.

1.3. Componentes

1.3.1. Base

El proyecto tiene la idea base de un motor de juego, donde se cuenta con una clase base llamada `HCIGame.java`

```
public class HCIGame {
    public static void main(String[] args) {

         // Instancia del Juego
        final VNGame game = new VNGame();

        // Instanciamos el JFrame y colocamos sus configuraciones
        JFrame frame = new JFrame(VNGame.TITTLE);
        frame.add(game);
        frame.setSize(VNGame.WIDTH, VNGame.HEIGHT);
        frame.setResizable(false);
        frame.setFocusable(true);

        frame.addWindowListener(new WindowAdapter() {
            @Override
            public void windowClosing(WindowEvent e) {
                System.err.println("Bye!");
                game.stop();
            }
        });

        frame.setLocationRelativeTo(null);
        frame.setVisible(true);

        //  Iniciamos la ventana
        game.run();
    }
}
```

Esta clase, inicializa al JFrame y carga en el la clase `VNGame.java` esta clase extiende de `Canvas` y extiende la interfaz `Runnable`

```
public class VNGame extends Canvas implements Runnable{
  ...
}
```

1.3.2. Doble Buffer

Está tecnica[1] es utilizada en el renderizado de las imagenes, de tal manera que las graficas en pantalla sean fluidas. Por esto, es que en la clase `VNGame.java` se implementa usando un `BufferStrategy`.

[1]El almacenamiento en búfer múltiple es el uso de más de un búfer para contener un bloque de datos, de modo que un "lector"verá una versión completa (aunque quizás antigua) de los datos, en lugar de una versión parcialmente actualizada de los datos siendo creado por un “escritor”.

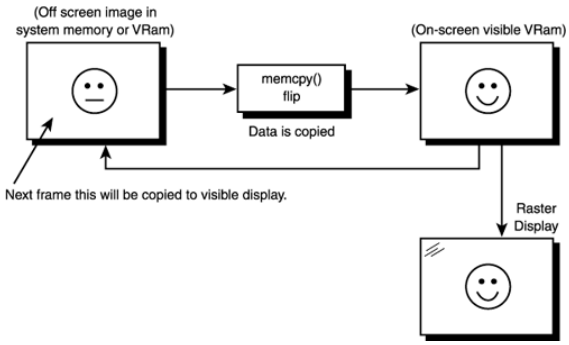


Figura 1.3: Esquema de Doble Buffer (Double Buffering and Page Flipping, The Java™ Tutorials)

```
private void render(){
    BufferStrategy bs = getBufferStrategy();
    if (bs == null) {
        createBufferStrategy(2);
        return;
    }

    Graphics g = bs.getDrawGraphics();
    Graphics2D g2d = (Graphics2D) g;
    ////////////////////////////////////////////////

    p.render(g2d,this);

    ////////////////////////////////////////////////
    g.dispose();
    bs.show();
}
```

Dentro de esta idea, mantenemos un hilo `thread` que es el encargado de la actualización de la entrada (`ticks/sec`) necesarios para la interacción.

La idea es realizar el calculo en base al tiempo entre imagenes renderizadas de tal manera que se obtiene movimientos fluidos (esto para no llevar el tiempo de actualizacion al minimo que ofrece el procesador). Realizamos los calculos en base a `System.nanoTime()`[2] y `System.currentTimeMillis()`[3].

```
double target = 60.0;
double nsPerTick = 1000000000.0 / target;
long lastTime = System.nanoTime();
long timer = System.currentTimeMillis();
// Delta
double unprocessed = 0.0;
int fps = 0;
int tps = 0;
```

[2]El método `java.lang.System.nanoTime()` devuelve el valor actual del temporizador del sistema más preciso disponible, en nanosegundos.

[3]El método `java.lang.System.currentTimeMillis()` devuelve la hora actual en milisegundos. La unidad de tiempo del valor de retorno es un milisegundo

Por lo que el método principal para la actualización queda de la siguente manera:

```
@Override
public void run() {

    running = true;
    requestFocus();
    double target = 60.0;
    double nsPerTick = 1000000000.0 / target;
    long lastTime = System.nanoTime();
    long timer = System.currentTimeMillis();
    // Delta
    double unprocessed = 0.0;
    int fps = 0;
    int tps = 0;
    boolean canRender = false;

    // Mientras    running    ejecutamos el run a ~60fps
    while (running) {
        long now = System.nanoTime();
        unprocessed += (now - lastTime) / nsPerTick;
        lastTime = now;

        if (unprocessed >= 0.5) {
            tick();
            p.update(this);
            unprocessed = 0;
            tps++;
            canRender = true;
        } else canRender = false;

        try {
            Thread.sleep(1);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        if (canRender) {
            render();
            fps++;
        }

        if (System.currentTimeMillis() - 1000 > timer) {
            timer += 1000;
            fps = 0;
            tps = 0;
        }

    }
    System.exit(0);
}
```

1.3.3. Interacción

Hasta ahora, todo lo referido a las clases implementadas, es netamente para propositos graficos. En esta sección se muestran las clases utilizadas para que la interacción se lleve a cabo.

1.3.3.1. `Player.java`

La clase principal es la clase `Player.java` que para este caso funciona como un intermediario entre la parte de renderizacion y la parte de interaccion ya que es la que contiene instancias de los botones y a la vez es a la que se le pasa el componente `Graphics` para mostrar las imagenes en pantalla. Los métodos principales que tiene esta clase son:

```
//Encargado de mostrar todo en Pantalla
public void render(Graphics g, VNGame game) {
  if (!go) {
      g.drawImage(image2, 0, 0, game);
  } else {
      g.drawImage(image, 0, 0, game);
      g.drawImage(girl.animaciones.get(girl.current), 0, 0, game);
      if (VNListButton != null) {
          for (VNButton var : VNListButton) {
              var.render(g);
          }
      }
  }
}

//Maneja la entrada por Mouse
public void mousePressed(MouseEvent e) {
    if (!clicked) {
        go = true;
        if (VNListButton != null) {
            for (VNButton var : VNListButton) {
                var.isClick(e, this, girl);
            }
        }
        clicked = true;
    }
}

//Maneja la posici n del Mouse
public void mouseMoved(MouseEvent e) {
    if (VNListButton != null && go) {
        for (VNButton var : VNListButton) {
            var.isInside(e);
        }
    }
}
```

1.3.3.2. `VNButton.java`

Para entender esta clase hay que ver la estructura de una novela visual, en este caso la que implementamos en este proyecto:

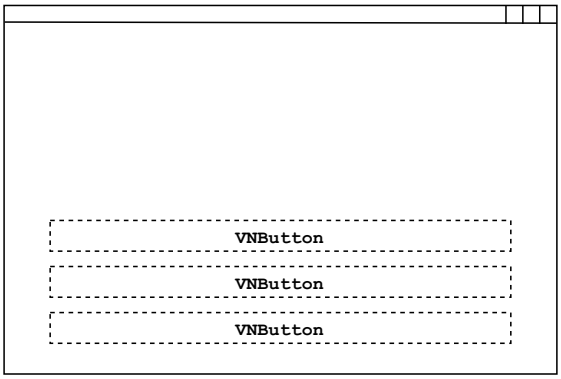


Figura 1.4: Esquema del Juego.

Donde en diferentes etapas del juego se muestran opciones a escoger, y estas llevan a otras, y así sucesivamente. Por esto es que la clase `VNButton.java` tiene entre sus atributos un listado de opciones a las que este lleva, es así que generamos el arbol de decisiones.

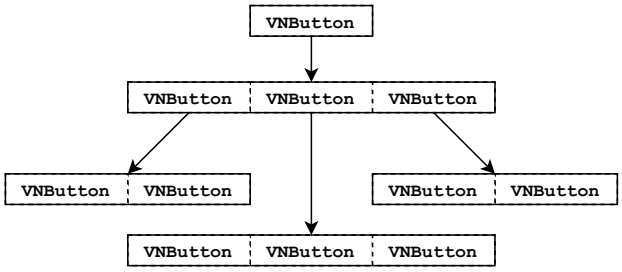


Figura 1.5: Arbol de Decisiones representado por `VNButtons`

Por esto, la parte principal de esta clase es el cambio de referencia a la lista del siguiente nivel en el arbol de decisiones:

```
public void isClick(MouseEvent e, Player player,NPC girl) {

    int xp = position.x + size.x;
    int yp = position.y + size.y;
    Point p = e.getPoint();

    if (p.x >= position.x && p.x <= xp && p.y >= position.y && p.y <= yp) {
        girl.current=mood;
    // Cambia a la siguiente kista
        player.VNListButton = Options;
    }
}
```

Respecto a la renderizacion, tambien se cuenta con los siguientes metodos:

```
public void render(Graphics g) {
    if (!inButton) {
        g.setColor(new Color(0.1f, 0.1f, 0.1f, 0.33f));
    } else {
        g.setColor(new Color(0.1f, 0.1f, 0.1f, 0.66f));
    }
    g.fillRect(position.x, position.y, size.x, size.y);
    g.setColor(new Color(1f, 1f, 1f, 1));
    drawString(g, text, position.x + 10, position.y + 5);
}
```

1.3.3.3. `NPC.java`

Esta clase es simplemente una abstraccion del personaje con el que interactuamos, y cuanta con un `map` compuesto de el nombre de la emocion y la imagen, ademas de un `string` que representa la emocion actual.

```
public class NPC {

    public static Map<String, Image> animaciones;
    public static String current = "normal";

    public NPC(){
        animaciones = new HashMap<>();
        current = "normal";
    }

}
```

Figura 1.6: NPC representado por la clase `NPC.java`

1.4. Resultados

En esta sección se expone el proyecto final. Los créditos de las imagenes van para Reitou Shijimi [4] y Liah Momiji.

2. Doodler: A Simple Paint Tool

2.1. Introducción

Este proyecto es una pequeña herramienta para dibujar, parecida al conocido **Paint**, solo que un poco mas simple, se puede dibujar a mano libre, y escoger el grosor del lapiz, ademas de borrar el dibujo hecho. La idea es hacer un programa interactivo donde el usuario aprecie cambios en tiempo real (el dibujo).

2.2. Antecedentes

2.2.1. La Historia temprana de Paint

Como antecedentes tenemos tal vez la historia de **Paint** [1]. La primera versión de Paint se introdujo con la primera versión de Windows, Windows 1.0, en noviembre de 1985 y solo admitía gráficos monocromos de 1 bit en un formato "MSP"patentado. Con las siguientes versiones de Windows Microsoft realizo actualizaciones de Paint con Windows 95 y Windows NT 4.0, que permite guardar y cargar un conjunto personalizado de pozos de color como paleta de colores. En versiones posteriores como Windows 98, Windows 2000 y Windows Me, Paint puede guardar imágenes en formatos JPEG, GIF y PNG cuando se instalan los complementos adecuados. Dichos complementos se incluyen con Microsoft Office y Microsoft PhotoDraw. Esto también permitia que Paint use fondos transparentes. Desde entonces, pasando por Windows 7, hasta hoy, Paint permanecióo igual hasta que en 2017 se le dió fin al soporte, y se lanzó **Paint 3D**. Que era como su predecesor, con la capacidad de cargar modelos en 3D.

Figura 2.1: Esta es la versión de ZSoft PC Paintbrush para Microsoft Windows que se ejecuta en Windows 1 y 2.

2.2.2. Pintura en AWT y Swing

En un sistema gráfico, un kit de herramientas de ventanas generalmente es responsable de proporcionar un marco para que sea relativamente sencillo que una interfaz gráfica de usuario (GUI) muestre los bits correctos en la pantalla en el momento adecuado. Cuando se desarrolló la AWT API original para JDK 1.0, solo existían componentes de peso pesado ("peso pesado"significa que el componente tiene su propia ventana nativa opaca). Esto permitió que AWT dependiera en gran medida del subsistema de pintura en cada plataforma nativa. [3]

2.3. Componentes

Los componentes usados son relativamente sencillos, se utilizo `JFrames, JButton,JRadioButton` y `JPanel` en la elaboración.

2.3.1. `Doodle.java`

Esta es la clase que extiende de `JFrame`, es la que carga los componentes, como ser los `Check` para escoger el grosor y el boton para limpiar el lienzo. Como podemos ver simplemente contiene una instancia de la clase `DrawPad.java`, necesaria para dibujar.

```
public class Doodler {
  // Ventana Principal
    JFrame frame = new JFrame("Paint");
  // DrawPad para dibujar
    final PadDraw drawPad = new PadDraw();
  // Otros componentes necesarios
    ImagePanel panel = new ImagePanel();
    JButton twoX = new JButton("2");
    JRadioButton rdbtnPx = new JRadioButton("3 px");
    JRadioButton rdbtnPx_1 = new JRadioButton("5 px");
    JRadioButton rdbtnPx_2 = new JRadioButton("12 px");
}
```

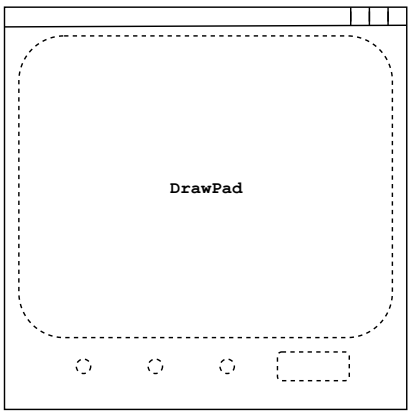


Figura 2.2: Diagrama, Doodler conteniendo al `DrawPad`

2.3.2. DrawPad.java

Esta es la clase mas importante, ya que es aca donde se lleva a cabo la lógica para dibujar. Esta clase extiende de `JComponent` (La clase base para todos los componentes Swing, excepto los contenedores de nivel superior). En los siguientes metodos es donde va toda la logica para dibujar.

```
addMouseListener(new MouseAdapter(){
    public void mousePressed(MouseEvent e){
        oldX = e.getX();
        oldY = e.getY();
    }
});

addMouseMotionListener(new MouseMotionAdapter(){
    public void mouseDragged(MouseEvent e){
        currentX = e.getX();
        currentY = e.getY();
        if(graphics2D != null)
        graphics2D.drawLine(oldX, oldY, currentX, currentY);
        repaint();
        oldX = currentX;
        oldY = currentY;
    }

});
```

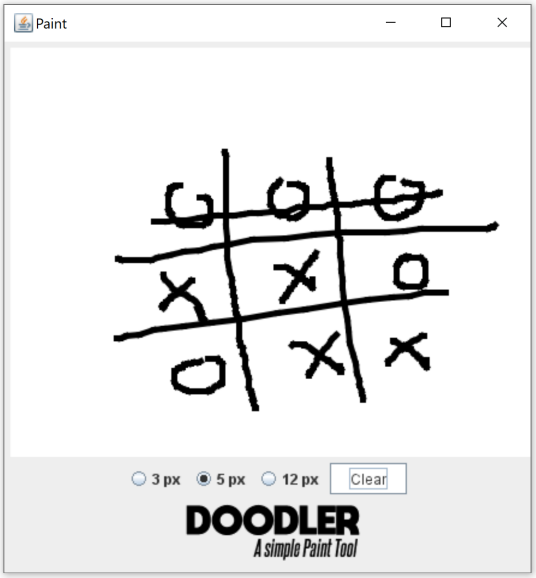
Las demas funciones de esta clase simplemente son setters y getters, a excepcion de `paintComponent` que es la encargada de setear la imagen en un principio.

```
public void paintComponent(Graphics g){
    if(image == null){
        image = createImage(getSize().width, getSize().height);
        graphics2D = (Graphics2D)image.getGraphics();
        graphics2D.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
            RenderingHints.VALUE_ANTIALIAS_ON);
        clear();

    }
    g.drawImage(image, 5, 5, null);
}
```

2.4. Resultados

En esta sección se encuentran capturas del proyecto terminado.



3. Conclusiones

Creo que ambos proyectos reflejan la Interacción Hombre-Computador que pretenden tener, por un lado el demo de una Novela Visual, hace que el usuario tenga que estar permanentemente interactuando ya que el esquema va a partir de la lectura del texto en pantalla, junto con la opcion de escoger el camino que quiera para la definicion de la historia, por el otro lado, el mini prototipo de Paint, creo que es util ya que es algo que deja al usuario una libertad de interaccion dentro de la usabilidad del probrama, plasmar lo que quiera en el programa, con total libertad. Creo que la parte mas intersante del trabajo es la estructura de cada proyecto individualmente, el uso de la OOP y algunos algoritmos y estructuras de datos ya conocidos. La complejidad no es mucha, pero creo que la union de los componentes gráficas y estos, crea una sinergia que da paso a la interacción con el usuario.

Bibliografía

[1] Samuel Banya. History of ms paint. *http://www.musimatic.net/FakeWebsites/TheHistoryOfMSPaint/index.html* .

[2] Brian Crimmins. A brief history of visual novels. *https://medium.com/mammon-machine-zeal/a-brief-history-of-visual-novels-641a2e6b1acb* , 2016.

[3] Amy Fowler. Painting in awt and swing. *https://www.oracle.com/technetwork/java/painting-140037.html* .

[4] Reitou Shijimi. しじみのおすまし, 背景素材(800x600) . *http://www7b.biglobe.ne.jp/~osumashi/*, 2019.

[5] Wikipedia. The portopia serial murder case. *https://en.wikipedia.org/wiki/The_Portopia_Serial_Murder_Case*, .

Anexos

VNGame.java

```
1
2  package hcigame;
3
4  import com.Player;
5  import java.awt.Canvas;
6  import java.awt.Graphics;
7  import java.awt.Graphics2D;
8  import java.awt.image.BufferStrategy;
9
10
11 public class VNGame extends Canvas implements Runnable{
12     public static String TITTLE = "
           Interaction";
13         public static final int WIDTH = 800;
14     public static final int HEIGHT = 600;
15     public static boolean running;
16     public static VNGame INSTANCE;
17     Player p;
18
19     public VNGame(){
20
21         p = new Player();
22         INSTANCE = this;
23     }
24
25     private void tick() {
26         //stateManager.tick();
27     }
28
29     private void render(){
30         BufferStrategy bs = getBufferStrategy();
31         if (bs == null) {
32             createBufferStrategy(2);
33             return;
34         }
35
36         Graphics g = bs.getDrawGraphics();
37         Graphics2D g2d = (Graphics2D) g;
```

```
38          ////////////////////////////////////////////////
39
40          p.render(g2d,this);
41
42          ////////////////////////////////////////////////
43          g.dispose();
44          bs.show();
45      }
46
47      @Override
48       public void run() {
49
50          running = true;
51          requestFocus();
52          double target = 60.0;
53          double nsPerTick = 1000000000.0 / target;
54          long lastTime = System.nanoTime();
55          long timer = System.currentTimeMillis();
56          // Delta
57          double unprocessed = 0.0;
58          int fps = 0;
59          int tps = 0;
60          boolean canRender = false;
61
62          // Mientras 'running' ejecutamos el run a ~60fps
63          while (running) {
64              long now = System.nanoTime();
65              unprocessed += (now - lastTime) / nsPerTick;
66              lastTime = now;
67
68              if (unprocessed >= 0.5) {
69                  tick();
70                  p.update(this);
71                  unprocessed=0;
72                  tps++;
73                  canRender = true;
74              } else canRender = false;
75
76              try {
77                  Thread.sleep(1);
78              } catch (InterruptedException e) {
79                  e.printStackTrace();
80              }
81
82              if (canRender) {
83                  render();
84                  fps++;
85              }
86
```

```
87              if (System.currentTimeMillis() - 1000 > timer) {
88                  timer += 1000;
89                  fps = 0;
90                  tps = 0;
91              }
92
93          }
94
95          System.exit(0);
96      }
97
98      public void stop() {
99          if (!running) return;
100         running = false;
101     }
102 }
```

Player.java

```
package com;

import hcigame.Game;
import hcigame.VNGame;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Image;
import java.awt.Point;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.MouseMotionListener;
import java.awt.image.BufferedImage;
import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.Arrays;
import javax.imageio.ImageIO;

public class Player implements MouseListener, MouseMotionListener {

    NPC girl;

    Point p1 = new Point(96, 440 - 32);
    Point p2 = new Point(96, 488 - 32);
    Point p3 = new Point(96, 536 - 32);

    Point s = new Point(608, 32);
    Point sz = new Point(608, 128);

    public ArrayList<VNButton> VNListButton;

    boolean clicked = false;
    private BufferedImage image;
    private BufferedImage image2;
    private BufferedImage imageChar;

    boolean pe = false;
    boolean go = false;

    public Player() {

        girl = new NPC();

        VNButton Text4 = new VNButton(p1, sz, "Bien~! <3 Me gusta la idea.
           Vamos, quiero aprender a usar Applets");
```

```
47          Text4.mood = "";
48
49          VNButton Text6 = new VNButton(p1, sz, "6 Horas Despues...");
50          Text6.mood = "normal";
51          Text4.Options.add(Text6);
52
53          VNButton Text7 = new VNButton(p1, sz, "Al fin! Creo que ya podemos
               hacer el informe. Ya revise el proyecto varias veces\nParece que
               esta todo bien. Le hice unas cuantas pruebas unitarias Y no hay
               fallas.\nUsar Applets parece algo muy facil y entretenido, pense
               que seria mas dificil. \nGracias por tu ayuda.
               (           \n\nVamos a comer un helado para celebrar! <3");
54          Text7.mood = "normal";
55          Text7.Options.add(Text7);
56          Text6.Options.add(Text7);
57
58          VNButton Text5 = new VNButton(p1, sz, "Hmmm... Yo pensaba ayudarte
               programando. (     _     ). Prefiero hacer la calculadora y luego
               hacemos\nel informe juntos.");
59          Text5.Options.add(Text6);
60
61          VNButton Text8 = new VNButton(p1, sz, "No me parece muy
               interesante, pero esta bien. \n\nSi tardamos mucho, podemos
               pedir unas pizzas mas tarde(?)");
62          VNButton Text9 = new VNButton(p1, sz, "10 Horas Despues...");
63          VNButton Text10 = new VNButton(p1, sz, "Vaya! tardamos mas de lo
               que pensaba... \n\nOh no! tengo que ir a mi casa antes de que
               anochezca. ");
64
65          VNButton Text11 = new VNButton(p1, sz, "Oh! Gracias <3");
66          VNButton Text12 = new VNButton(p1, sz, "Ara ara~ Podemos pedir las
               pizzas y terminamos de hacer el informe.");
67          Text12.mood = "normal";
68          Text12.Options.add(Text12);
69          VNButton B6 = new VNButton(p1, s, "Si quieres te acompa o a tu
               casa");
70          B6.mood = "embar";
71          B6.Options.add(Text11);
72          VNButton B7 = new VNButton(p2, s, "Puedes quedarte a dormir si
               quieres.");
73          B7.mood = "embar";
74          B7.Options.add(Text12);
75
76          Text9.mood = "surprised";
77          Text8.Options.add(Text9);
78          Text9.Options.add(Text10);
79          Text10.mood = "surprised";
80          Text10.Options.add(B6);
81          Text10.Options.add(B7);
```

```
82          Text11.mood = "embar";
83          Text11.Options.add(Text11);
84
85          VNButton B1 = new VNButton(p1, s, "Podriamos hacer una Graficadora
               de Fractales.");
86          B1.mood = "sad";
87          VNButton B2 = new VNButton(p2, s, "Podemos hacer algo f cil ,
               hagamos una calculadora.");
88          B2.mood = "normal";
89          B2.Options.add(Text8);
90          VNButton B3 = new VNButton(p3, s, "Hmm... No se. Quisiera jugar
               videojuegos un rato.");
91          B3.mood = "upset";
92          VNButton B4 = new VNButton(p1, s, "No te preocupes, te puedo
               explicar mientras pensamos como hacerlo.");
93          B4.mood = "surprised";
94          B4.Options.add(Text4);
95          VNButton B5 = new VNButton(p2, s, "Puedes ir haciendo el informe o
               si quieres puedes ir haciendo la calculadora");
96          B5.mood = "surprised";
97          B5.Options.add(Text5);
98
99          VNButton Text2 = new VNButton(p1, sz, "Hmm... No podemos vaguear
               >_<~! ");
100         Text2.mood = "normal";
101         Text2.Options.add(B1);
102         Text2.Options.add(B2);
103         Text2.Options.add(B3);
104         B3.Options.add(Text2);
105         VNButton Text3 = new VNButton(p1, sz, " No  te parece un poco
               complicado? No se mucho aun sobre Applets -_-\n"
106                 + " creo que no podria ayudarte mucho en este trabajo. *sad
                       noises");
107         Text3.mood = "normal";
108         Text3.Options.add(B4);
109         Text3.Options.add(B5);
110
111         B1.Options.add(Text3);
112
113         VNButton Text1 = new VNButton(p1, sz, "Oh~ Hola...! No te habia
               visto. Tenemos que terminar el proyecto de Interaccion
               Hombre-Computador\nDeberiamos pensar que hacer, no tengo muchas
               ideas pero... yo creo "
114                 + "que podemos terminarlo en 24 hrs. \n Primero deberiamos
                       pensar que proyecto hacer,  ya  haz pensado en algo?");
115         Text1.mood = "surprised";
116
117         Text1.Options.add(B1);
118         Text1.Options.add(B2);
```

```
119         Text1.Options.add(B3);
120 
121         try {
122             image =
                    ImageIO.read(getClass().getResourceAsStream("/background.jpg"));
123             image2 =
                    ImageIO.read(getClass().getResourceAsStream("/menu.jpg"));
124 
125             imageChar =
                    ImageIO.read(getClass().getResourceAsStream("/spr_NPC.png"));
126 
127             girl.animaciones.put("normal", imageChar);
128 
129             imageChar =
                    ImageIO.read(getClass().getResourceAsStream("/spr_NPC2.png"));
130 
131             girl.animaciones.put("surprised", imageChar);
132 
133             imageChar =
                    ImageIO.read(getClass().getResourceAsStream("/spr_NPC3.png"));
134 
135             girl.animaciones.put("upset", imageChar);
136 
137             imageChar =
                    ImageIO.read(getClass().getResourceAsStream("/spr_NPC4.png"));
138 
139             girl.animaciones.put("sad", imageChar);
140 
141             imageChar =
                    ImageIO.read(getClass().getResourceAsStream("/spr_NPC5.png"));
142 
143             girl.animaciones.put("embar", imageChar);
144 
145         } catch (IOException e) {
146 
147         }
148         VNButton total = new VNButton(0, 0, 800, 600, "");
149         total.mood = "normal";
150         total.Options.add(Text1);
151         VNListButton = new ArrayList<>(Arrays.asList(
152                 total
153         ));
154     }
155 
156     public void update(VNGame game) {
157         game.addMouseListener(this);
158         game.addMouseMotionListener(this);
159 
160     }
```

```
161
162     public void render(Graphics g, VNGame game) {
163         if (!go) {
164             g.drawImage(image2, 0, 0, game);
165         } else {
166             g.drawImage(image, 0, 0, game);
167 
168             g.drawImage(girl.animaciones.get(girl.current), 0, 0, game);
169             if (VNListButton != null) {
170                 for (VNButton var : VNListButton) {
171                     var.render(g);
172                 }
173             }
174         }
175 
176     }
177 
178     @Override
179     public void mouseClicked(MouseEvent e) {
180 
181     }
182 
183     @Override
184     public void mousePressed(MouseEvent e) {
185         if (!clicked) {
186             go = true;
187             if (VNListButton != null) {
188                 for (VNButton var : VNListButton) {
189                     var.isClick(e, this, girl);
190                 }
191             }
192 
193             clicked = true;
194         }
195 
196     }
197 
198     @Override
199     public void mouseReleased(MouseEvent e) {
200         clicked = false;
201     }
202 
203     @Override
204     public void mouseEntered(MouseEvent e) {
205     }
206 
207     @Override
208     public void mouseExited(MouseEvent e) {
209 
```

```
210     }
211
212     @Override
213     public void mouseDragged(MouseEvent e) {
214
215     }
216
217     @Override
218     public void mouseMoved(MouseEvent e) {
219         if (VNListButton != null && go) {
220             for (VNButton var : VNListButton) {
221                 var.isInside(e);
222             }
223         }
224     }
225 }
```

VNButton.java

```java
package com;

import java.awt.Color;
import java.awt.Graphics;
import java.awt.Point;
import java.awt.event.MouseEvent;
import java.util.ArrayList;

public class VNButton {

    public ArrayList<VNButton> Options;
    public Point position;
    public Point size;
    boolean inButton = false;
    boolean isBox = false;
    String text;
    public String mood;

    public VNButton(int x, int y, int w, int h, String text) {

        Options = new ArrayList<>();
        position = new Point(x, y);
        size = new Point(w, h);
        this.text = text;
    }

    public VNButton(Point pos, Point size, String text) {

        Options = new ArrayList<>();
        position = pos;
        this.size = size;
        this.text = text;
    }

    public void render(Graphics g) {
        if (!inButton) {
            g.setColor(new Color(0.1f, 0.1f, 0.1f, 0.33f));
        } else {
            g.setColor(new Color(0.1f, 0.1f, 0.1f, 0.66f));
        }

        g.fillRect(position.x, position.y, size.x, size.y);

        g.setColor(new Color(1f, 1f, 1f, 1));

        drawString(g, text, position.x + 10, position.y + 5);
    }
```

```
48
49      void drawString(Graphics g, String text, int x, int y) {
50          for (String line : text.split("\n")) {
51              g.drawString(line, x, y += g.getFontMetrics().getHeight());
52          }
53      }
54
55      public void isClick(MouseEvent e, Player player, NPC girl) {
56
57          int xp = position.x + size.x;
58          int yp = position.y + size.y;
59          Point p = e.getPoint();
60
61          if (p.x >= position.x && p.x <= xp && p.y >= position.y && p.y <=
                yp) {
62              girl.current = mood;
63              player.VNListButton = Options;
64          }
65      }
66
67      public void isInside(MouseEvent e) {
68
69          if (!isBox) {
70              int xp = position.x + size.x;
71              int yp = position.y + size.y;
72              Point p = e.getPoint();
73
74              if (p.x >= position.x && p.x <= xp && p.y >= position.y && p.y
                    <= yp) {
75                  inButton = true;
76              } else {
77                  inButton = false;
78              }
79          }
80      }
81
82  }
```

NPC.java

```
1
2  package com;
3
4  import java.awt.Image;
5  import java.util.AbstractMap;
6  import java.util.HashMap;
7  import java.util.Map;
8
9  public class NPC {
10
11     public static Map<String, Image> animaciones;
12     public static String current = "normal";
13
14     public NPC(){
15         animaciones = new HashMap<>();
16         current = "normal";
17     }
18
19 }
```

Doodler.java

```
package p2;

import java.awt.Graphics;
import java.awt.*;
import java.awt.event.*;
import java.awt.image.BufferedImage;
import javax.swing.*;

public class Doodler {

    public static void main(String[] args) {

        JFrame frame = new JFrame("Paint");

        Container content = frame.getContentPane();

        content.setLayout(new BorderLayout());

        final PadDraw drawPad = new PadDraw();

        content.add(drawPad, BorderLayout.CENTER);

        ImagePanel panel = new ImagePanel();

        panel.setPreferredSize(new Dimension(100, 96));
        panel.setMinimumSize(new Dimension(100, 96));
        panel.setMaximumSize(new Dimension(100, 96));

        JButton twoX = new JButton("2");
        twoX.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {
                drawPad.clear();
            }
        });

        content.add(panel, BorderLayout.SOUTH);

        JRadioButton rdbtnPx = new JRadioButton("3 px");
        panel.add(rdbtnPx);

        JRadioButton rdbtnPx_1 = new JRadioButton("5 px");
        panel.add(rdbtnPx_1);

        JRadioButton rdbtnPx_2 = new JRadioButton("12 px");
        panel.add(rdbtnPx_2);

```

```
        ButtonGroup bg = new ButtonGroup();
        bg.add(rdbtnPx);
        bg.add(rdbtnPx_1);
        bg.add(rdbtnPx_2);

        rdbtnPx.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                drawPad.small();
            }
        });
        rdbtnPx_1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                drawPad.medium();
            }
        });
        rdbtnPx_2.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                drawPad.big();
            }
        });

        JButton clearButton = new JButton("Clear");
        clearButton.setBackground(new Color(255, 255, 255));
        clearButton.setFont(UIManager.getFont("TextArea.font"));

        clearButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                drawPad.clear();
            }
        });
        panel.add(clearButton);

        frame.setSize(454, 480);

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        frame.setVisible(true);

    }
}
```

DrawPad.java

```
package p2;
import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Image;
import java.awt.RenderingHints;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.MouseMotionAdapter;
import javax.swing.JComponent;


class PadDraw extends JComponent{

private Image image;
private Graphics2D graphics2D;
private int currentX , currentY , oldX , oldY ;

public PadDraw(){
    setDoubleBuffered(false);
    addMouseListener(new MouseAdapter(){
        public void mousePressed(MouseEvent e){
            oldX = e.getX();
            oldY = e.getY();
        }
    });

    addMouseMotionListener(new MouseMotionAdapter(){
        public void mouseDragged(MouseEvent e){
            currentX = e.getX();
            currentY = e.getY();
            if(graphics2D != null)
            graphics2D.drawLine(oldX, oldY, currentX, currentY);
            repaint();
            oldX = currentX;
            oldY = currentY;
        }

    });

}

public void paintComponent(Graphics g){
    if(image == null){
        image = createImage(getSize().width, getSize().height);
        graphics2D = (Graphics2D)image.getGraphics();
```

```
48          graphics2D.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
                RenderingHints.VALUE_ANTIALIAS_ON);
49          clear();
50
51      }
52      g.drawImage(image, 5, 5, null);
53  }
54
55  public void clear(){
56      graphics2D.setPaint(Color.white);
57      graphics2D.fillRect(0, 0, getSize().width, getSize().height);
58      graphics2D.setPaint(Color.black);
59      repaint();
60  }
61
62  public void small(){
63      graphics2D.setStroke(new BasicStroke(1));;
64  }
65  public void medium(){
66      graphics2D.setStroke(new BasicStroke(5));;
67  }
68  public void big(){
69      graphics2D.setStroke(new BasicStroke(12));;
70  }
71
72  }
```