Método del Triangulo Para la Suma de Vectores

(Programa en java)

Andrea Robles Hernández 23SIC004

Física

Ing. Vanesa Tenopala Zavala

29/02/2024

Índice:

Tabla de Ilustraciones:

Introducción.

El método del triángulo para la suma de vectores es una técnica geométrica que permite encontrar la resultante de dos o más vectores. Este método se basa en la ley del paralelogramo, que establece que la suma vectorial de dos vectores se puede representar por la diagonal de un paralelogramo construido a partir de esos dos vectores.

Sin embargo, en situaciones donde solo se tienen dos vectores, el método del triángulo simplifica este proceso al considerar únicamente un triángulo. Este triángulo está formado por los dos vectores dados y la resultante, siendo uno de los vectores la hipotenusa y los otros dos lados del triángulo.

La aplicación del método del triángulo para la suma de vectores implica lo siguiente:

Representar gráficamente los vectores dados con una escala adecuada en un plano cartesiano. Los vectores se dibujan como segmentos de flecha con una dirección y magnitud específicas.

Colocar la punta de un vector al origen del otro vector, formando así un triángulo con ambos vectores como lados. La resultante es la línea que conecta el origen del primer vector con la punta del segundo vector.

Medir las magnitudes de los dos vectores y los ángulos entre ellos utilizando instrumentos adecuados o técnicas trigonométricas.

Aplicar la ley de los cosenos o la ley de senos, dependiendo de la información disponible, para encontrar la magnitud y dirección de la resultante. La magnitud se determina mediante la fórmula de la suma de los cuadrados de las magnitudes de los dos vectores y el doble del producto de sus magnitudes y el coseno del ángulo entre ellos.

El método del triángulo para la suma de vectores es una herramienta visual y geométrica que facilita la comprensión de las relaciones entre vectores y puede ser útil en la resolución de problemas prácticos que involucren movimientos o fuerzas concurrentes.

Suma de vectores “Método del triángulo”.

Descripción del programa.

Se plantea realizar un programa para la suma de vectores mediante el método del triángulo, dicho programa contara con interfaz grafica de usuario (GUI) para una mejor visualización por parte del usuario, dicha interfaz contara con paneles y graficas para el uso y trazado del método del triángulo.

El programa utilizará una paquetería adecuada la cual permita el trazado y medición de ángulos, además de que se pondrá a prueba con las medidas vistas en clase, para asegurar su correcto funcionamiento.

En éste caso para que fuese más fácil modificar en un futuro, se creó una interfaz gráfica en la cual podemos ir conectando futuros métodos y así poder ir complementado el programa realizado.

Lenguaje y entorno de programación.

El programa se realizó en lenguaje Java el cual cuenta con una amplia versatilidad para este tipo de operaciones y ejecuciones.

Al ser en lenguaje Java se eligió el programa JGrasp, el cual ofrece facilidades en la creación de programas, en conjunto con el JDK de Java, el cual contiene todas las librerías de Java, y simplemente se mandan a llamar en el código.

Código Java.

Primero visualizaremos el código de la ventana principal donde está la interfaz gráfica.

```
 1 import javax.swing.*;
 2 import java.awt.*;
 3 import java.awt.event.ActionEvent;
 4 import java.awt.event.ActionListener;
 5
 6 public class VectorCalculator extends JFrame {
 7
 8     public VectorCalculator() {
 9         initUI();
10     }
11
12     private void initUI() {
13         // Configurar ventana principal
14         setTitle("Calculadora de Vectores");
15         setSize(800, 600);
```

```
16          setLocationRelativeTo(null);
17          setDefaultCloseOperation(EXIT_ON_CLOSE);
18
19          // Crear menú
20          JMenuBar menuBar = new JMenuBar();
21          JMenu fileMenu = new JMenu("Archivo");
22          JMenuItem openItem = new JMenuItem("Abrir");
23          JMenuItem saveItem = new JMenuItem("Guardar");
24          JMenuItem closeItem = new JMenuItem("Salir");
25
26          // Agregar acciones
27          closeItem.addActionListener((event) ->
System.exit(0));
28
29          fileMenu.add(openItem);
30          fileMenu.add(saveItem);
31          fileMenu.add(closeItem);
32          menuBar.add(fileMenu);
33
34          // Submenú para operaciones con vectores
35          JMenu vectorMenu = new JMenu("Vectores");
36          JMenuItem sumVectorsItem = new JMenuItem("Sumar
Vectores");
37
38          sumVectorsItem.addActionListener(new
ActionListener() {
39              public void actionPerformed(ActionEvent e) {
40                  // Aquí iría el código para abrir el panel
de suma de vectores
41                  EventQueue.invokeLater(() -> {
42                  VectorSuma VS = new VectorSuma();
43                  VS.setVisible(true);
44                  });
45              }
46          });
47
48          vectorMenu.add(sumVectorsItem);
49          menuBar.add(vectorMenu);
50
51          setJMenuBar(menuBar);
52      }
53
54      private void openVectorSumPanel() {
55          // Implementar lógica para abrir el panel de suma de
vectores
56          JOptionPane.showMessageDialog(this, "Aquí se abriría
el panel para sumar vectores.");
```

```
57      }
58
59      public static void main(String[] args) {
60          EventQueue.invokeLater(() -> {
61              VectorCalculator vc = new VectorCalculator();
62              vc.setVisible(true);
63          });
64      }
65  }
```

Posterior a esto se muestra el código del programa que permite el cálculo de la suma de vectores de manera lógica y gráfica.

```
  1 import javax.swing.*;
  2 import java.awt.*;
  3 import java.awt.event.ActionEvent;
  4 import java.awt.geom.Line2D;
  5
  6 public class VectorSuma extends JFrame {
  7
  8     private JPanel inputPanel;
  9     private JPanel drawPanel;
 10     private JTextField vector1xField;
 11     private JTextField vector1yField;
 12     private JTextField vector2xField;
 13     private JTextField vector2yField;
 14     private JButton calculateButton;
 15     private int sumX, sumY; // Estos almacenarán la suma de
los vectores
 16
 17     public VectorSuma() {
 18         initUI();
 19     }
 20
 21     private void initUI() {
 22         setTitle("Suma de Vectores");
 23         setSize(800, 600);
 24         setLocationRelativeTo(null);
 25         setDefaultCloseOperation(EXIT_ON_CLOSE);
 26         setLayout(new BorderLayout());
 27
 28         inputPanel = new JPanel(new FlowLayout());
 29
 30         vector1xField = new JTextField(5);
 31         vector1yField = new JTextField(5);
 32         vector2xField = new JTextField(5);
 33         vector2yField = new JTextField(5);
 34         calculateButton = new JButton("Calcular");
```

```
 35
 36          inputPanel.add(new JLabel("Vector 1 (x, y): "));
 37          inputPanel.add(vector1xField);
 38          inputPanel.add(vector1yField);
 39          inputPanel.add(new JLabel("Vector 2 (x, y): "));
 40          inputPanel.add(vector2xField);
 41          inputPanel.add(vector2yField);
 42          inputPanel.add(calculateButton);
 43
 44          drawPanel = new JPanel() {
 45              protected void paintComponent(Graphics g) {
 46                  super.paintComponent(g);
 47                  Graphics2D g2d = (Graphics2D) g;
 48                  drawCoordinateSystem(g2d);
 49
 50                  // Solo dibuja los vectores si se han
calculado (sumX y sumY tienen valor)
 51                  if (sumX != 0 || sumY != 0) {
 52                      drawVectors(g2d);
 53                  }
 54              }
 55          };
 56
 57          drawPanel.setBackground(Color.WHITE);
 58
 59          add(inputPanel, BorderLayout.NORTH);
 60          add(drawPanel, BorderLayout.CENTER);
 61
 62          calculateButton.addActionListener(e ->
calculateAndDraw());
 63      }
 64
 65      private void calculateAndDraw() {
 66          try {
 67              int v1x =
Integer.parseInt(vector1xField.getText());
 68              int v1y =
Integer.parseInt(vector1yField.getText());
 69              int v2x =
Integer.parseInt(vector2xField.getText());
 70              int v2y =
Integer.parseInt(vector2yField.getText());
 71
 72              // Suma de vectores
 73              sumX = v1x + v2x;
 74              sumY = v1y + v2y;
 75
```

```
 76              // Solicita a drawPanel que se repinte para
mostrar los nuevos vectores
 77              drawPanel.repaint();
 78          } catch (NumberFormatException e) {
 79              JOptionPane.showMessageDialog(this, "Por favor,
ingrese valores válidos.");
 80          }
 81      }
 82
 83      private void drawCoordinateSystem(Graphics2D g2d) {
 84
 85        int width = drawPanel.getWidth();
 86        int height = drawPanel.getHeight();
 87        int centerX = width / 2;
 88        int centerY = height / 2;
 89
 90        // Ejes
 91        g2d.drawLine(centerX, 0, centerX, height); // Eje Y
 92        g2d.drawLine(0, centerY, width, centerY); // Eje X
 93
 94        // Etiquetas para los ejes
 95        g2d.drawString("N", centerX - 10, 15);
 96        g2d.drawString("S", centerX - 10, height - 5);
 97        g2d.drawString("E", width - 15, centerY + 15);
 98        g2d.drawString("O", 5, centerY + 15);
 99
100        // Etiquetas para x y y
101        g2d.drawString("X", width - 20, centerY - 10);
102        g2d.drawString("Y", centerX + 5, 10);
103
104        // Marcas en los ejes
105        int axisStep = 10; // Cambia esto según la escala que
desees
106        for (int i = centerX; i < width; i += axisStep) {
107          g2d.drawLine(i, centerY - 5, i, centerY + 5);
108          g2d.drawLine(width - i, centerY - 5, width - i,
centerY + 5);
109          // Dibujando etiquetas de escala en el eje X
110          if (i != centerX) { // Evita sobreponer el "0"
111              g2d.drawString(Integer.toString((i - centerX) /
axisStep), i - 5, centerY + 20);
112              g2d.drawString(Integer.toString(-(i - centerX)
/ axisStep), width - i - 5, centerY + 20);
113          }
114      }
115      for (int i = centerY; i < height; i += axisStep) {
116          g2d.drawLine(centerX - 5, i, centerX + 5, i);
```

```
117             g2d.drawLine(centerX - 5, height - i, centerX + 5,
height - i);
118             // Dibujando etiquetas de escala en el eje Y
119             if (i != centerY) { // Evita sobreponer el "0"
120                 g2d.drawString(Integer.toString(-(i - centerY)
/ axisStep), centerX + 10, i + 5);
121                 g2d.drawString(Integer.toString((i - centerY) /
axisStep), centerX + 10, height - i + 5);
122             }
123         }
124     }
125
126     private void drawVectors(Graphics2D g2d) {
127         int centerX = drawPanel.getWidth() / 2;
128         int centerY = drawPanel.getHeight() / 2;
129         int v1x =
Integer.parseInt(vector1xField.getText());
130         int v1y =
Integer.parseInt(vector1yField.getText());
131
132         g2d.drawLine(centerX, centerY, centerX + v1x,
centerY - v1y);
133         g2d.drawLine(centerX + v1x, centerY - v1y, centerX
+ sumX, centerY - sumY);
134         g2d.setColor(Color.RED);
135         g2d.drawLine(centerX, centerY, centerX + sumX,
centerY - sumY);
136     }
137
138     public static void main(String[] args) {
139         EventQueue.invokeLater(() -> {
140             VectorSuma VS = new VectorSuma();
141             VS.setVisible(true);
142         });
143     }
144 }
```

Podemos observar que es un código con algunas deficiencias pues, aunque conecta y funciona como uno solo, no logra abarcar todos los requerimientos solicitados con anterioridad, por lo que es un programa que esta aun en una etapa de mejora.

Capturas de ejecución.

Al cargar el programa en primera instancia se mostrará la interfaz gráfica y su distribución. Tal como podemos observar en la siguiente imagen.

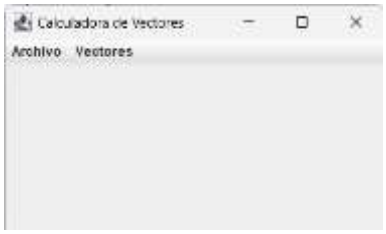

Ilustración 1 Ventana Principal.

En la parte de archivo, al seleccionar se visualizan tres opciones, comúnmente utilizadas abrir, guardar, y salir, tal como se muestra en la siguiente imagen.

Ilustración 2 Menú.

En la segunda sección podemos observar donde dice Vectores, al presionar podemos observar una opción de suma de vectores por el método del triángulo, como se muestra a continuación.

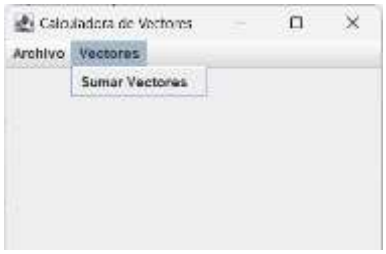

Ilustración 3 Opciones.

Como podemos observar a continuación, al hacer Crick en esa opción, agregue una ventana con etiquetas para poder visualizar mejor la dimensión de los vectores, estas etiquetas se colocaron por medio de un contador pues agregarlas manualmente una por una aunque sería un poco tedioso e innecesario pues ocuparíamos bastantes líneas de código innecesarias, además de agregar lo que se pretende graficar, así mismo etiquetas para de X, Y así como también de N, S, E, O para que sea más entendible y practico de utilizar, pes estas al modificar el tamaño de la ventana se ajustan automáticamente.

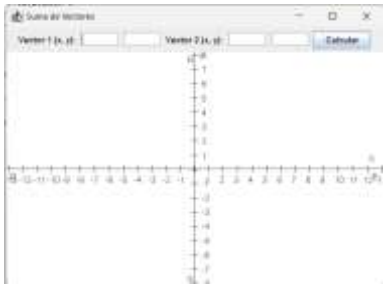

Ilustración 4 Organización de Elementos.

En esta ventana es donde podemos ingresar dos vectores, los cuales va a graficar y si cumplen con las especificaciones va a crear un triángulo con una tercera recta la cual representa al desplazamiento, la cual tiene una escala de que 20 = 1 km u 20 = 1 n. Tal como se muestra a continuación.

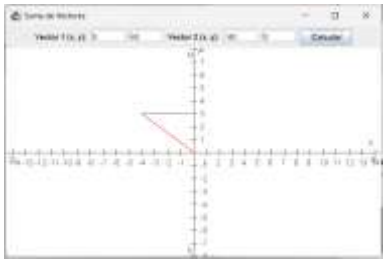

Ilustración 5 Ejemplo 1.

En la siguiente imagen se puede apreciar otro ejemplo de su funcionamiento.

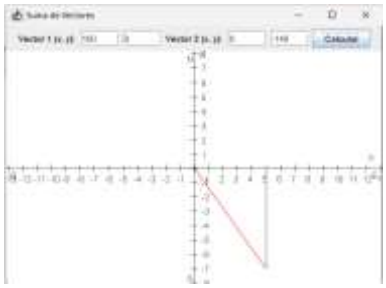

Ilustración 6 Ejemplo 2.

Este programa es una ayuda para representar sumas de vectores, por medio del método del triángulo.

Conclusión.

En conclusión, el Método del Triángulo para la suma de vectores es una técnica geométrica eficaz y visualmente intuitiva para encontrar la resultante de dos vectores. Su simplicidad y enfoque en la construcción de un triángulo con los vectores dados permiten una representación clara de las magnitudes y direcciones involucradas en la suma vectorial.

A través de este método, es posible visualizar y comprender fácilmente la relación entre los vectores, aprovechando la geometría del triángulo formado. Esto simplifica la resolución de problemas relacionados con fuerzas, desplazamientos u otras magnitudes vectoriales en situaciones bidimensionales.

Sin embargo, es importante destacar que el método del triángulo puede volverse menos práctico a medida que se enfrenta a un mayor número de vectores o situaciones tridimensionales, donde otras técnicas algebraicas, como la descomposición de vectores en componentes rectangulares, pueden resultar más eficientes.

En resumen, el Método del Triángulo es una herramienta valiosa para introducir y comprender la suma de vectores, proporcionando una base visual sólida que facilita el análisis y la resolución de problemas vectoriales en un plano bidimensional.

Referencias bibliográficas.

Java Platform SE 8. (s. f.-b). https://docs.oracle.com/javase/8/docs/api/