

# GRAFICACIÓN

Doctor Norberto Castillo García

José Gustavo Zimbrón Lara

Nº de Control: 15820154

PRÁCTICA 10 – CURVA DE BEZIER

15 mayo de 2018

## OBJETIVOS DE LA PRACTICA

Realizar un programa el cual grafique la curva de Bezier dados n puntos de control.

## DESARROLLO DE LA PRÁCTICA

Comenzamos por importar las clases necesarias

```
] import java.awt.Color;
import java.awt.Graphics;
import java.awt.Polygon;
import java.awt.image.BufferedImage;
import java.util.ArrayList;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;
- import javax.swing.table.DefaultTableModel;
```

Declaramos las variables necesarias, es decir las coordenadas de los puntos en x y en y, estos serán almacenados e ArrayList, ancho y alto de elementos de la cuadrícula y la escala, que es el tamaño por pixel de cada unidad de la cuadrícula.

```
ArrayList<Double> px = new ArrayList();
ArrayList<Double> py = new ArrayList();
int ancho, alto, escala;
```

En el constructor de la clase mandamos llamar el método `pre_grafica()`; la cual dibuja la cuadrícula.

```
public CurvaBezier() {
    initComponents();
    pre_grafica();
}
```

### Método pre\_grafica()

Obtenemos los valores de ancho alto y escala del JFrame, convirtiendo el numero escrito en el jtextfield a un entero.

Instanciamos un objeto de tipo BufferedImage picture que medirá el ancho multiplicado por la escala y el alto por la escala, Instanciamos un objeto de tipo Graphics g y la obtenemos del objeto picture.

Utilizamos el método pinta\_cuadrícula() y pinta\_puntos() para graficar la rejilla y los puntos creados hasta ahora. Y se muestra en el jlabel labelshow.

```
private void pre_grafica() {
    ancho = Integer.valueOf(fancho.getText());
    alto = Integer.valueOf(falto.getText());
    escala = Integer.valueOf(fescala.getText());
    BufferedImage picture =
        new BufferedImage(ancho * escala,
            alto * escala, BufferedImage.TYPE_INT_RGB);

    Graphics g = picture.getGraphics();

    g = pinta_cuadrícula(g);
    g = pinta_puntos(g, true);
    labelshow.setIcon(new ImageIcon(picture));
}
```

### Método pinta\_cuadrícula()

Dos for que recorren el ancho y alto de la imagen, y van dibujando las líneas de la rejilla con los métodos fillRect.

```
private Graphics pinta_cuadrícula(Graphics g) {
    g.setColor(Color.WHITE);
    g.fillRect(0, 0, ancho * escala, alto * escala);
    g.setColor(new Color(0, 0, 0, 0.1f));

    for (int i = escala; i <= ancho * escala; i += escala) {
        g.fillRect(i, 0, 1, alto * escala);
    }
    for (int i = escala; i <= alto * escala; i += escala) {
        g.fillRect(0, i, ancho * escala, 1);
    }
    return g;
}
```

## Método pinta\_puntos()

Establece el color del cual dibujaremos los puntos y los agrega recorriendo el ArrayList de los px y py y agrega en la rejilla.

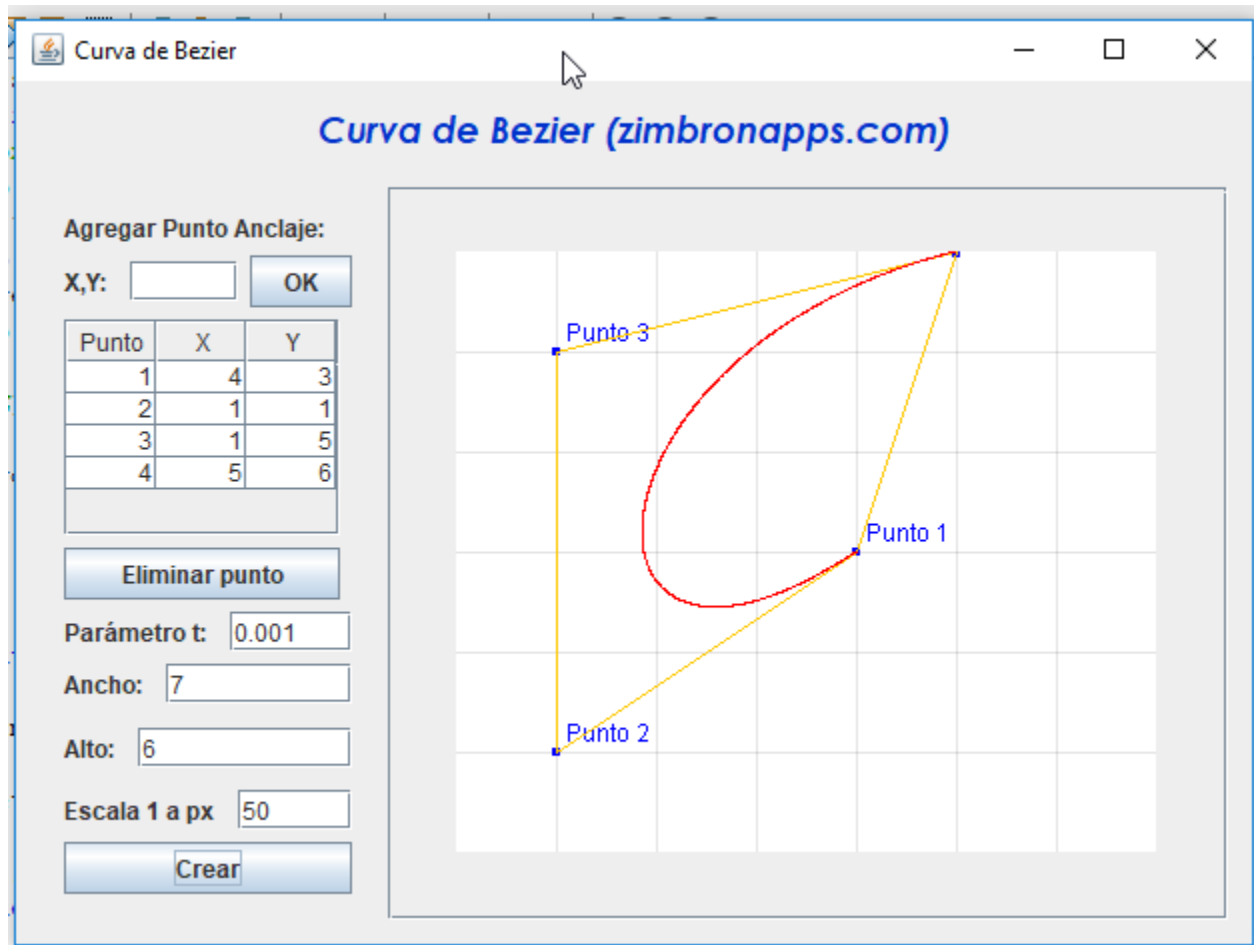
Si el parámetro **dibuja\_poligono** es verdadero, dibuja el polígono creado por los puntos de anclaje.

```
private Graphics pinta_puntos(Graphics g, boolean dibuja_poligono) {  
    // dibujando puntos de anclaje  
    g.setColor(Color.blue);  
    Polygon poligono = new Polygon();  
    for (int i = 0; i < px.size(); i++) {  
        poligono.addPoint((int) (px.get(i) * escala), (int) (alto * escala - py.get(i) * escala));  
        g.fillOval((int) (px.get(i) * escala - (0.05 * escala)),  
                (int) (alto * escala - py.get(i) * escala - (0.05 * escala)),  
                (int) (0.1 * escala), (int) (0.1 * escala));  
        g.drawString("Punto " + (i + 1),  
                (int) (px.get(i) * escala) + 5,  
                (int) (alto * escala - py.get(i) * escala) - 5);  
    }  
    g.setColor(Color.ORANGE);  
    if (dibuja_poligono) {  
        g.drawPolygon(poligono);  
    }  
    return g;  
}
```

Para la creación de puntos se escriben en el jTextField y se parsea con el método:

```
private void clic_agregar() {  
    // TODO add your handling code here:  
    String campo_add = tf_punto.getText().trim();  
    Pattern p = Pattern.compile("([0-9]+[.]?[0-9]?), ([0-9]+[.]?[0-9]?");  
    Matcher m = p.matcher(campo_add);  
    if (m.matches()) {  
        tf_punto.setText("");  
        String[] res = m.group(0).split(",");  
        px.add(Double.parseDouble(res[0]));  
        py.add(Double.parseDouble(res[1]));  
        rellenar_tabla();  
    } else {  
        JOptionPane.showMessageDialog(null, "Ingresa coordenadas válidas");  
    }  
}
```

Utiliza REGEX para revisar si está bien escrito las coordenadas, si lo está agrega al arraylist px y py, y después manda llamar el método rellenar\_tabla()



### Método rellenar\_tabla()

Obtiene el DefaultTableModel de la tabla y con un for recorremos los arraylist px y py y los agregamos en la columna 1 y 2.

```
private void rellenar_tabla() {
    DefaultTableModel tabla_modelo
        = (DefaultTableModel) tablapuntos.getModel();
    tabla_modelo.setRowCount(0);
    for (int i = 0; i < px.size(); i++) {
        tabla_modelo.addRow(new Object[]{i + 1, px.get(i), py.get(i)});
    }
    pre_grafica();
}
```

### Métodos funcionParametrica(), factorial() y combinaciones()

Son métodos matemáticos que tratan ciertos datos y devuelven un valor de resultado.

```
private double funcionParametrica(ArrayList<Double> puntos, double t) {
    double restemp = 0;

    for (int i = 0; i < puntos.size(); i++) {
        restemp
            += puntos.get(i) * combinaciones(puntos.size() - 1, i)
               * Math.pow(t, i) * Math.pow((1 - t),
               (puntos.size() - 1 - i));
    }

    return restemp;
}

private int factorial(int numero) {
    if (numero > 1) {
        return numero * factorial(numero - 1);
    } else {
        return 1;
    }
}

private int combinaciones(int n, int m) {
    return factorial(n) / (factorial(m) * factorial(n - m));
}
```

## Método clic\_crear()

Verifica si hay al menos 2 puntos de anclaje, y si lo hay obtiene el ancho, alto y escala de la cuadrícula, el parámetro T y llama al método curvaBezier para dibujar la curva.

```
private void clic_crear() {
    if (px.size() < 2) {
        JOptionPane.showMessageDialog(null, "Se requieren al menos 2 puntos de anclaje");
    } else {
        int ancho, alto, escala;
        double parametrot;

        ancho = (fanchito.getText().isEmpty()) ? 0 : Integer.parseInt(fanchito.getText());
        alto = (faltito.getText().isEmpty()) ? 0 : Integer.parseInt(faltito.getText());
        escala = (fescala.getText().isEmpty()) ? 0 : Integer.parseInt(fescala.getText());
        parametrot = Double.parseDouble(pt.getText());
        curvaBezier(ancho, alto, escala, parametrot);
    }
}
```

## Método curvaBezier()

Instancia el método BufferedImage del tamaño ancho y alto multiplicados por la escala, y obtenemos su objeto Graphic de ella.

Pintamos cuadrícula y puntos con sus métodos específicos.

Establecemos el color de g en rojo, y con un while recorremos entre 0 y 1 sumando parametroT a cada iteración.

Dentro del while obtenemos la posición en x y en y de cada punto utilizando el método funcionParametrica() y lo multiplicamos por la escala.

Obtenidos estos datos, dibujamos el pixel.

```
private void curvaBezier(int ancho, int alto, int escala, double parametro_t) {
    BufferedImage picture
        = new BufferedImage(ancho * escala, alto * escala, BufferedImage.TYPE_INT_RGB);

    Graphics g = picture.getGraphics();

    g = pinta_cuadrícula(g);
    g = pinta_puntos(g, true);

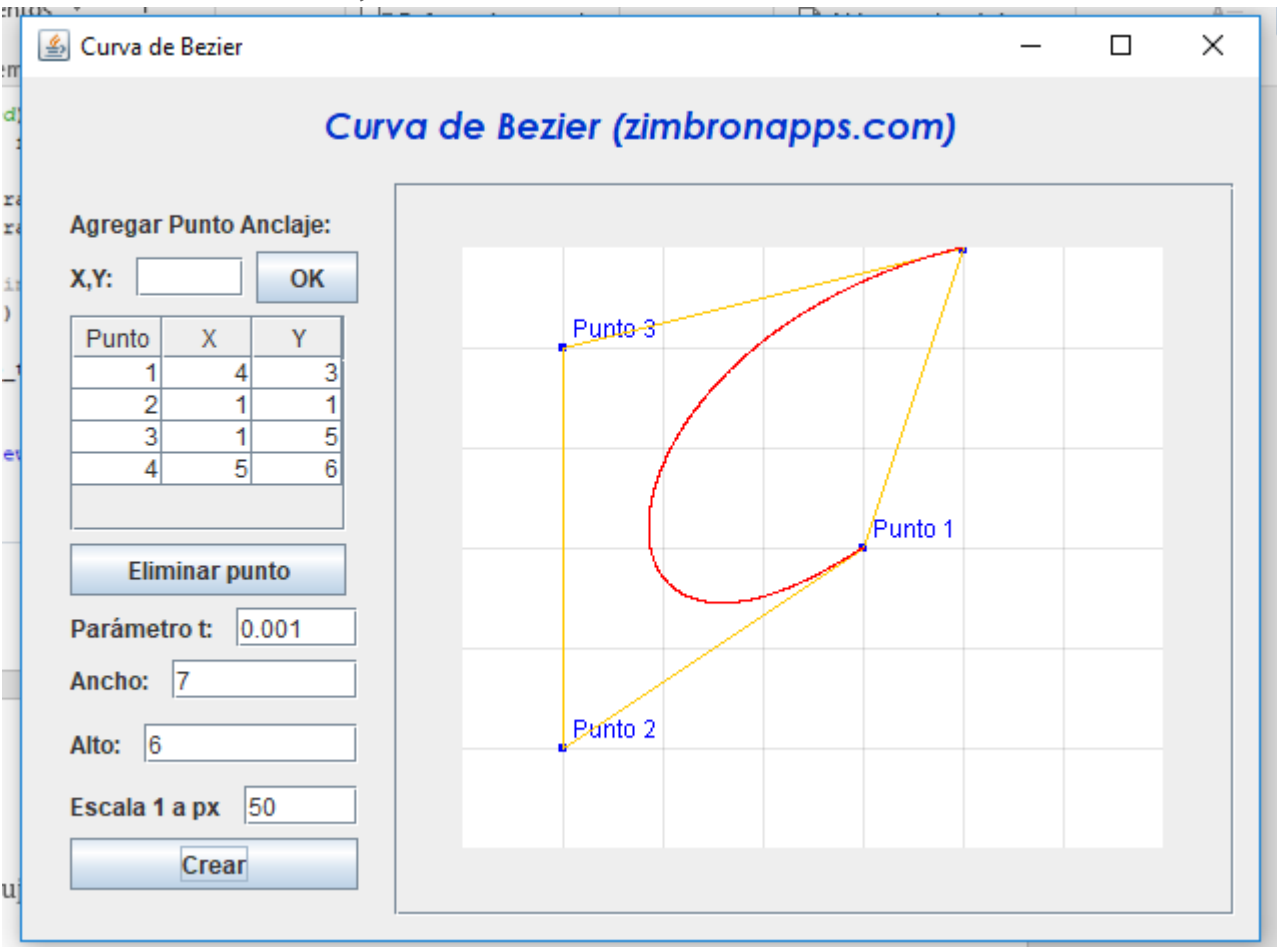
    g.setColor(Color.red);
    double it = 0, fpx, fpy;
    while (it <= 1) {
        fpx = funcionParametrica(px, it) * escala;
        fpy = funcionParametrica(py, it) * escala;

        //System.out.println("t: "+it+" x:"+fpx/escala+" y:"+fpy/escala);
        g.fillRect((int) fpx, (int) (alto * escala - fpy), 1, 1);

        it += parametro_t; // / escala;
    }

    labelshow.setIcon(new ImageIcon(picture));
}
```

Esto nos resulta en el dibujo de la curva de bezier:

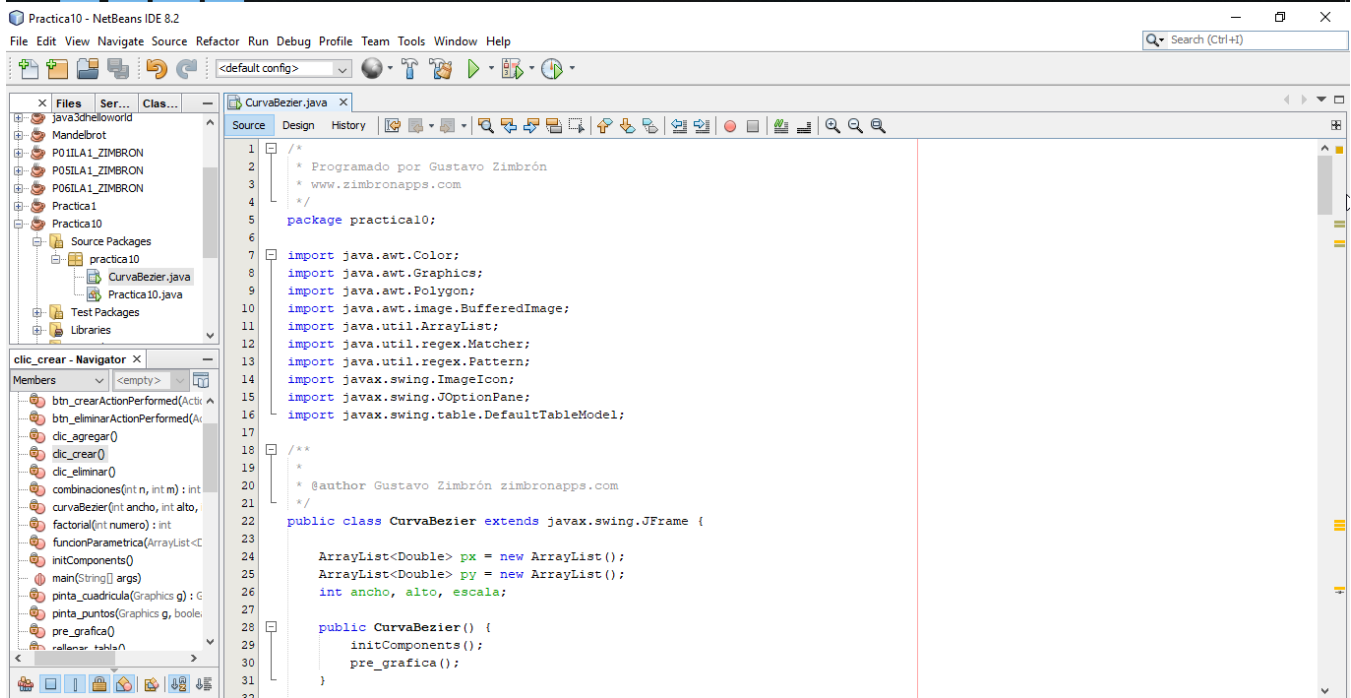
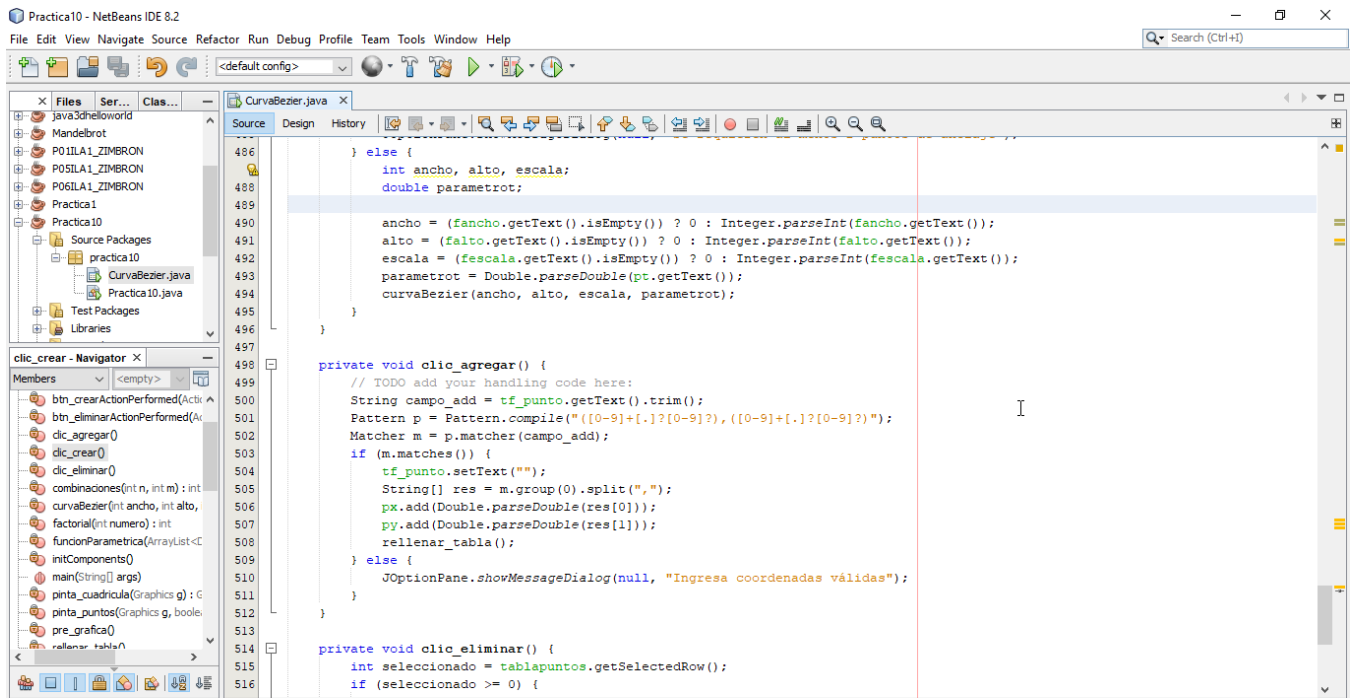




## CONCLUSIONES

Con esta práctica aprendimos y comprendimos el método matemático para crear las curvas de Bezier, utilizando la función paramétrica, utilizando fillRect y fillOval para dibujar los puntos y los pixeles.

## ANEXOS



Curva de Bezier

*Curva de Bezier (zimbronapps.com)*

Agregar Punto Anclaje:

X,Y:

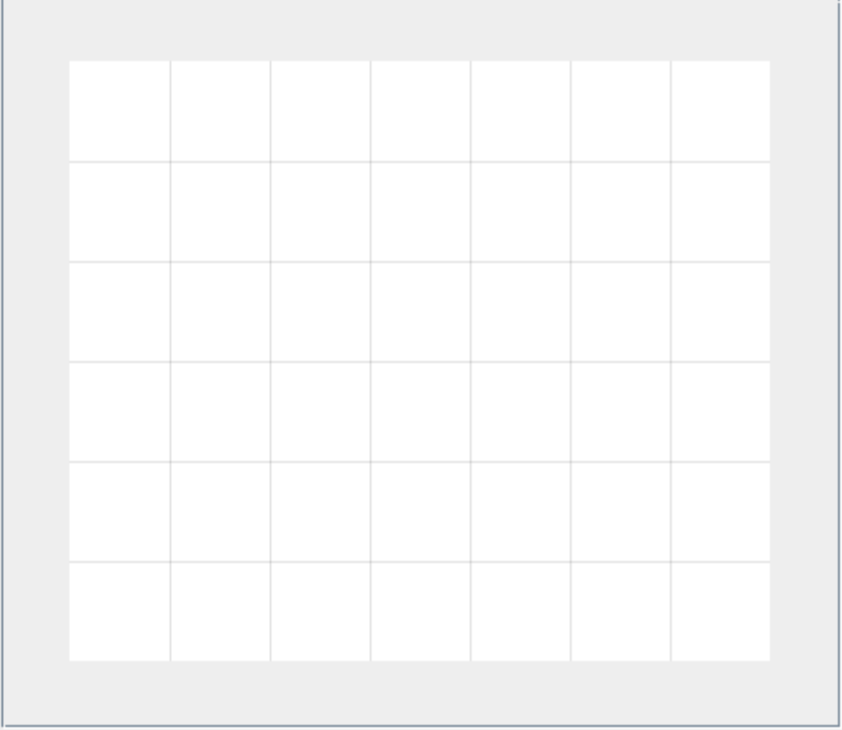
Punto	X	Y

Parámetro t:

Ancho:

Alto:

Escala 1 a px



Curva de Bezier

*Curva de Bezier (zimbronapps.com)*

Agregar Punto Anclaje:

X,Y:

Punto	X	Y
1	1	1

Parámetro t:

Ancho:

Alto:

Escala 1 a px

Punto 1

Curva de Bezier

Curva de Bezier (zimbronapps.com)

Agregar Punto Anclaje:

X,Y:

Punto	X	Y
1	1	1
2	5	1

Parámetro t:

Ancho:

Alto:

Escala 1 a px

Página 13

