Interfaces Gráficas de Usuario

Aplicación para la representación gráfica de funciones en WPF

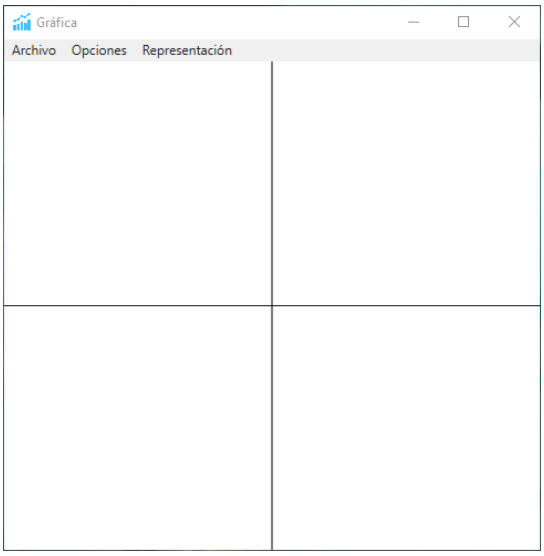
Pablo Jesús González Rubio

Curso 2020-21

Índice

Manual de usuario

Al abrir el programa se desplegará esta ventana:



Esta ventana se utilizará para la representación de los puntos respecto a sus ejes X e Y.

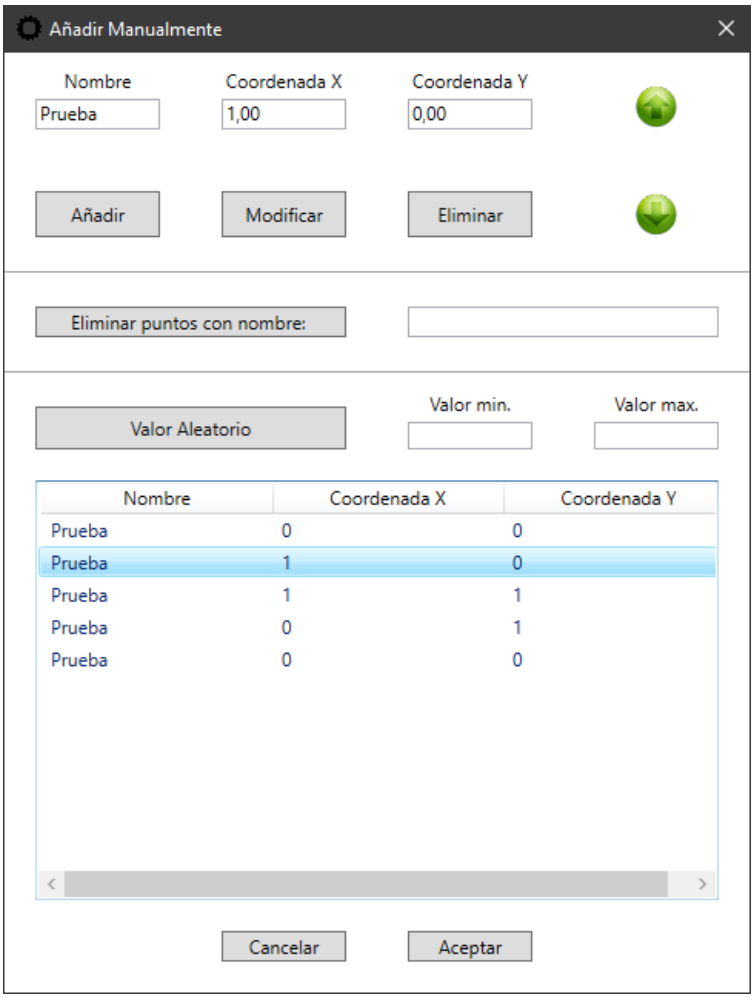
En la parte superior se encuentra el panel de navegación en el cual están distintos desplegables.

Archivo

En este apartado se explicarán los métodos de introducción de puntos y la opción de guardar la gráfica.

Introducción manual de puntos

Al seleccionar esta opción se abrirá una ventana como esta:



Se muestran varias casillas para la introducción de los datos:

El **nombre** se permite especificar para distinguir un punto de otro, pero si no se le asigna uno, se pondrá uno de la forma “Punto X” siendo X el número del orden en el que se ha creado.

La casilla “**Coordenada X”** y “**Coordenada Y”** admite valores enteros y decimales y conforman las coordenadas del punto, este se puede añadir con el botón “**Añadir**”.

Una vez añadido un punto, se puede modificar su posición reescribiendo las casillas de “Coordenada X” y “Coordenada Y”, y pulsando el botón de modificar.

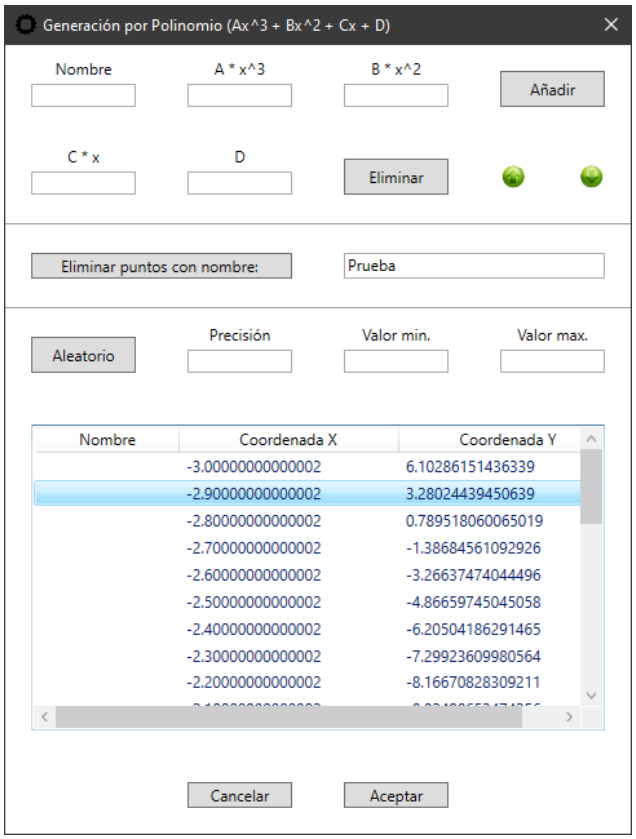
Este se puede reordenar hacia arriba o hacia abajo en la lista pulsando en las flechas “Arriba” y “Abajo”.

Si hubiera varios puntos, estos se pueden eliminar mediante su nombre, o bien seleccionando el primero a eliminar y manteniendo la tecla “Shift” hasta pulsar en el último, y pulsando el botón “Eliminar”.

Si se quisiera un valor aleatorio existe el botón “**Aleatorio**”, el cual genera un valor aleatorio de X e Y dado un rango mínimo de -9 y rango máximo de 9, los dos limitando la salida del punto Y a esos valores, aunque estos se pueden modificar en las casillas de “**Valor Min.**” y “**Valor Max.**”.

Para añadir los puntos a la gráfica y que dibuje la línea que siguiera esos puntos sólo habría que pulsar el botón “**Aceptar**”, en caso de que no se quisieran confirmar los cambios a “**Cancelar**”.

Introducción semiautomática mediante un polinomio



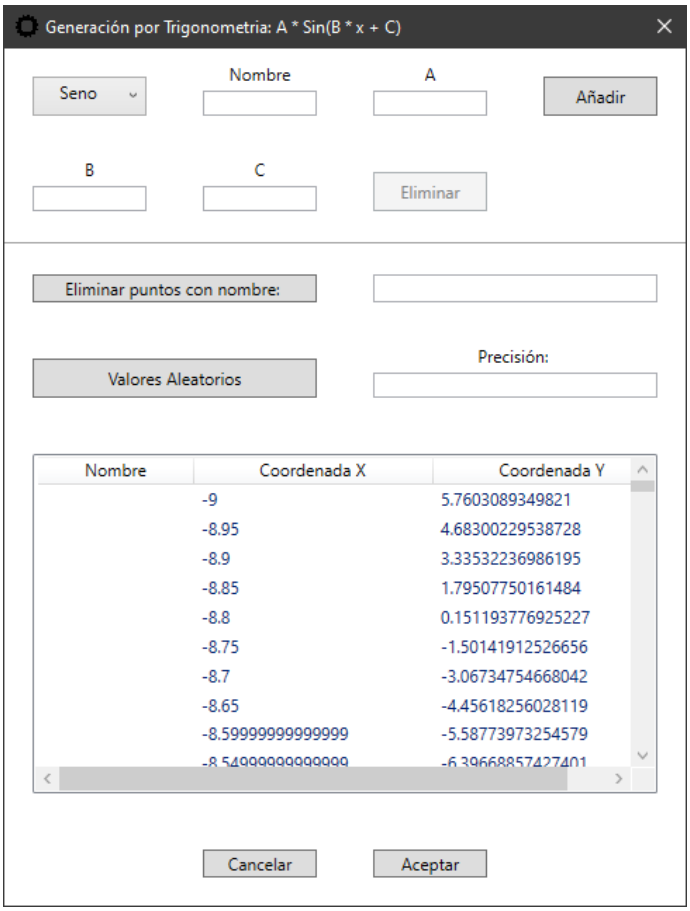
Esta ventana permite añadir puntos aleatoriamente o dados unos parámetros A, B, C y/o D que conforman el polinomio de 3º grado $Ax^3 + Bx^2 + Cx + D$.

Se muestran varias casillas para la introducción de los datos:

La casilla "**A",** "**B",** "**C"** y/o **"D"** admite valores enteros y decimales y conforman los parámetros del polinomio, al presionar el botón "**Añadir**" se generarán una serie de puntos que conforman ese polinomio.

Si se quisiera un polinomio aleatorio existe el botón "**Aleatorio**", el cual genera los puntos de la función polinómica dados unos parámetros "A", "B", "C" y "D" aleatorios.

Introducción semiautomática mediante funciones trigonométricas



Esta ventana permite añadir puntos aleatoriamente o dados unos parámetros A, B y/o C que conforman una función trigonométrica $A\ sen(Bx\ +\ C)$.

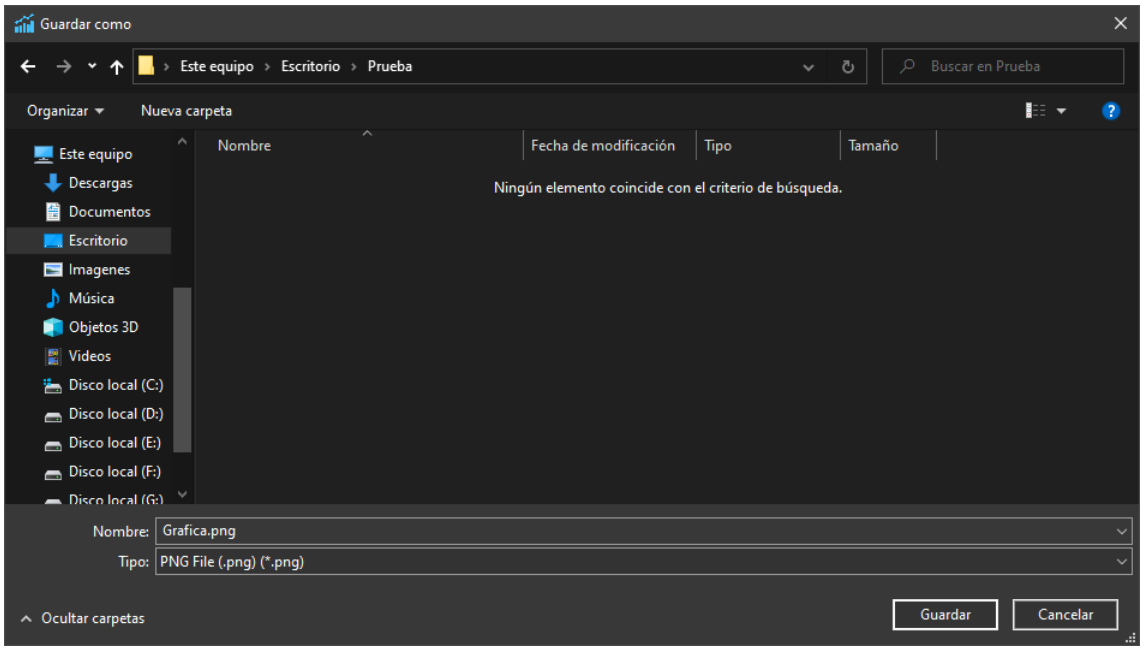
Se muestran varias casillas para la introducción de los datos:

El **tipo de función**, que puede ser “Seno”, “Coseno” o “Tangente”.

La casilla “**A”,** “**B”** y **“C”** admite valores enteros y decimales y conforman los parámetros de la función trigonométrica.

Si se quisiera una función aleatoria existe el botón “**Aleatorio**”, el cual selecciona “Seno”, “Coseno” o “Tangente” y genera los puntos de dicha función.

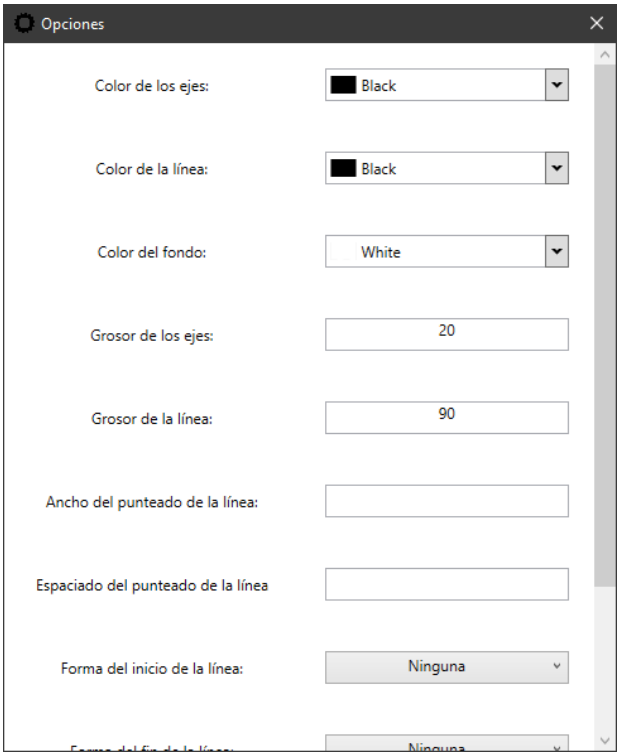
Guardar Imagen



Esta función permite guardar la imagen en pantalla, sea esta la polilínea o cualquiera de los otros métodos de representación.

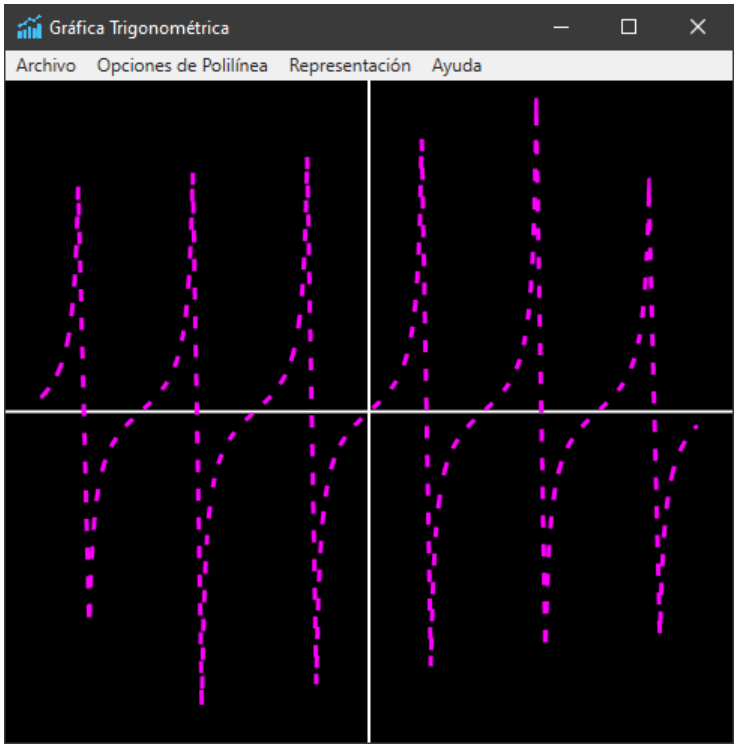
Opciones

Apariencia



Permite modificar el color de los ejes, de la polilínea y del fondo en el que se representa, así como de los grosores de los ejes y de la polilínea. También permite cambiar el trazado de esta, pudiéndose elegir la forma en que empieza y termina la polilínea (ninguna, triangular, cuadrada o redonda) y el punteado que puede tener.

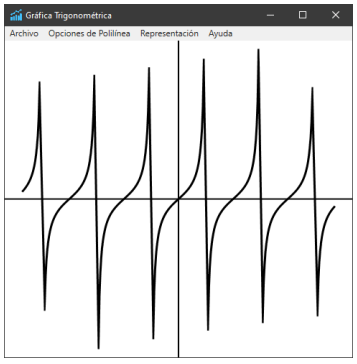
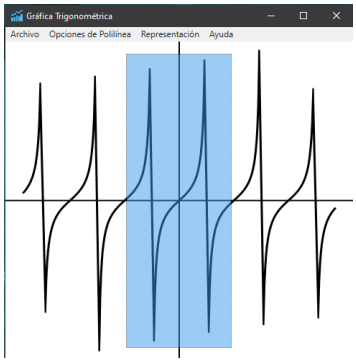
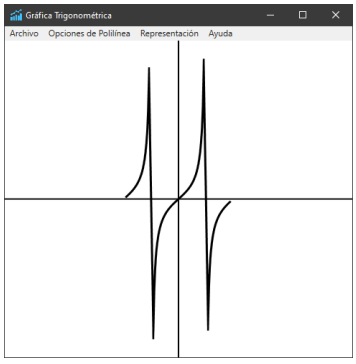
Un ejemplo de uso de todas las opciones es este:



Purgado

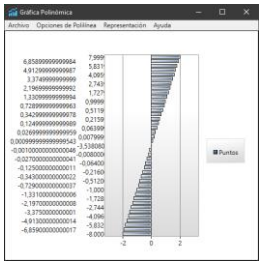
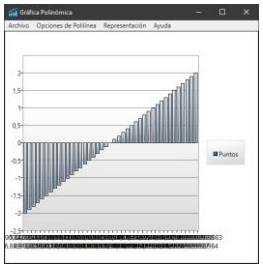
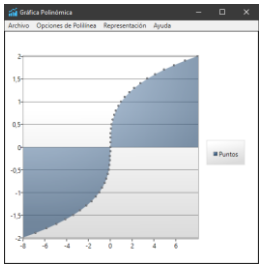
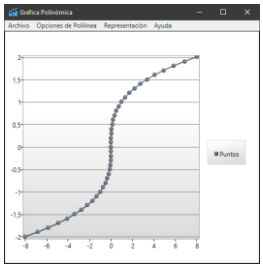
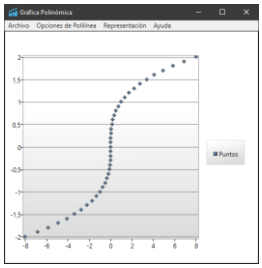
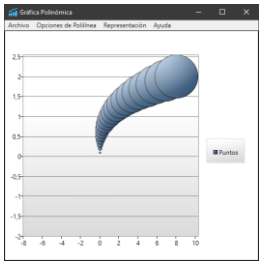
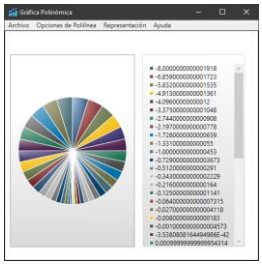
El purgado es una función especial que permite seleccionar un área en pantalla y borrar todo lo que no esté contenido en él.

Como mejor se entiende es con un ejemplo:

Inicial	Selección Purga	Purgado
		

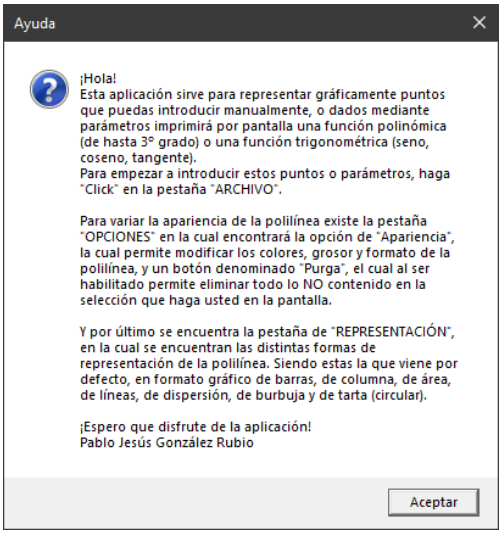
Representación

En la tabla inferior se pueden apreciar los distintos tipos de representación con una función polinómica x^3.

Polilínea	Gráfico de Barras	Gráfico de Columna	Gráfico de Área
			
Gráfico de Línea	**Gráfico de Dispersión**	**Gráfico de Burbuja**	**Gráfico de Tarta**
			

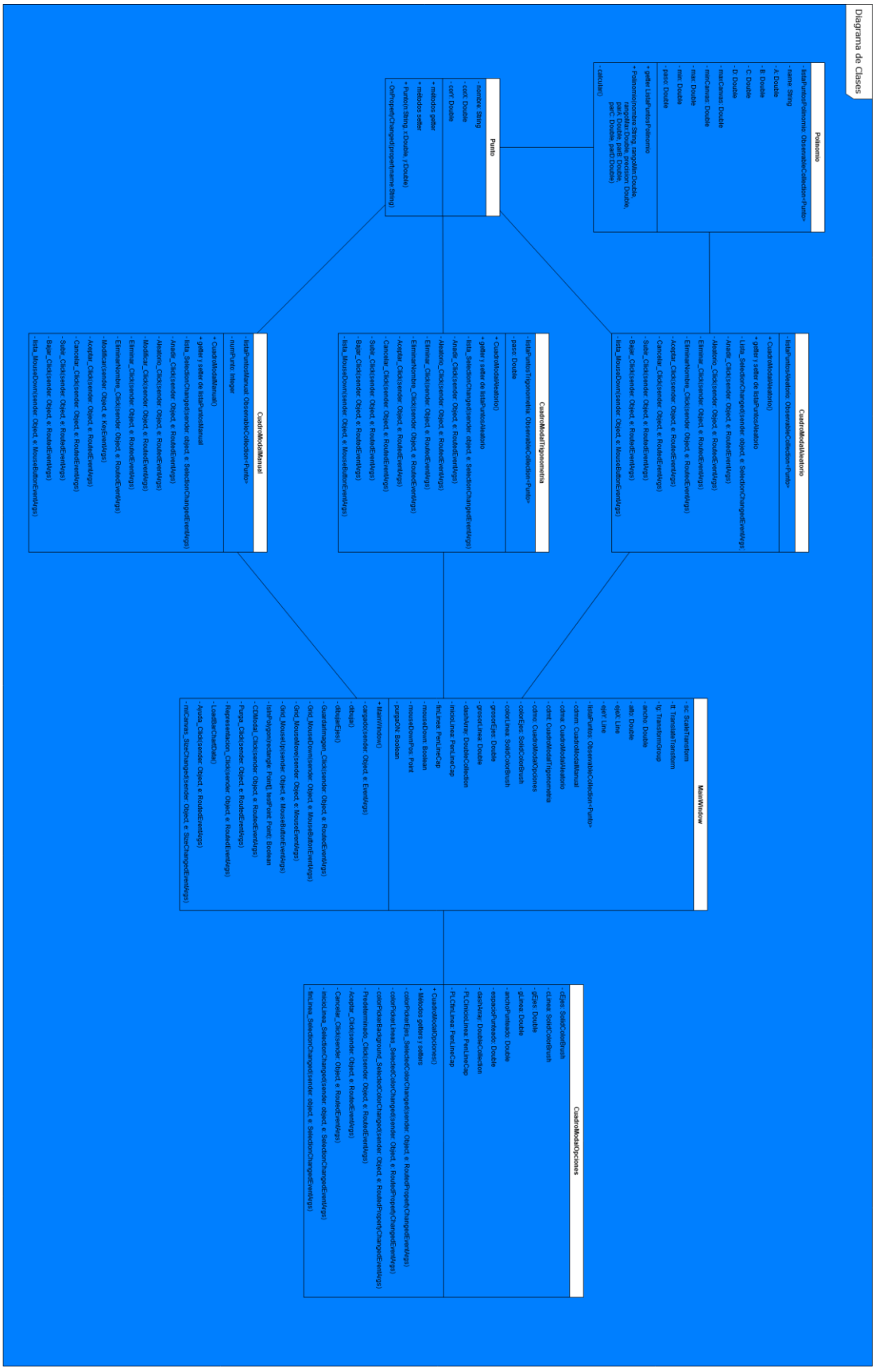
Ayuda

Esta opción muestra un texto con información de ayuda para el manejo del programa.



Manual de programador

En este apartado se explica cómo está desarrollada la aplicación.



MainWindow

Variables

Variables de redimensionado: se utilizan para gestionar la redimensión automática del canvas y sus elementos.

```
private ScaleTransform sc;

private TranslateTransform tt;

private TransformGroup tg;

private double ancho, alto;

private Line ejeX, ejeY;
```

Lista de puntos: Colección de elementos de la clase Punto que se utiliza para gestionar los puntos a dibujar, sus valores los obtiene mediante los métodos “Getter” de cada ventana modal.

```
private ObservableCollection<Punto> listaPuntos;
```

Variables de ventana modal: se declaran e instancian las ventanas modales para poder llamarlas desde la ventana principal.

```
private CuadroModalManual cdmm;

private CuadroModalAleatorio cdma;

private CuadroModalTrigonometria cdmt;

private CuadroModalOpciones cdmo;
```

Variables de aspecto: se utilizan por defecto para dibujar la polilínea, se pueden modificar desde la ventana modal Opciones para alterar las propiedades de la línea a dibujar.

```
private SolidColorBrush colorEjes = Brushes.Black;

private SolidColorBrush colorLinea = Brushes.Black;

private double grosorEjes = 20;

private double grosorLinea = 90;

private DoubleCollection dashArray = null;

private PenLineCap inicioLinea = 0;

private PenLineCap finLinea = 0;
```

Variables de purgado: se utilizan para gestionar el cuadro de selección para la purga de puntos.

```
private bool mouseDown = false;

private Point mouseDownPos;

private bool purgaON = false;
```

Métodos

- Se utiliza para inicializar la ventana principal, instanciar las variables de redimensionado y gestionarlas.
    - `public MainWindow()`
- Inicializa las variables de ventana modal a “NULL” y llama a la función de dibujado de ejes.
    - `private void cargado(object sender, EventArgs e)`
- Añade las coordenadas contenidas en la lista de puntos como puntos de tipo “Point” en una línea de tipo “Polyline”. A la línea le da las propiedades de las variables de aspecto (color, grosor, punteado y forma de inicio y final de línea) y la añade al canvas.
    - `private void dibuja()`
- Genera dos variables de tipo “Line” especificándoles la altura y anchura del canvas y el color y grosor que se obtiene de las variables de aspecto, y las añade al canvas.
    - `private void dibujarEjes()`
- Gestiona la obtención de la pantalla actual y la renderiza como un Bitmap, por último, la exporta como un fichero de imagen con extensión “.png”.
    - `private void GuardarImagen_Click(object sender, RoutedEventArgs e)`
- Gestionan la selección del área para el purgado de los puntos no contenidos en esa área. Sólo se realiza el purgado si la variable “purgaON” tiene valor verdadero.
    - `private void Grid_MouseDown(object sender, MouseButtonEventArgs e)`
    - `private void Grid_MouseMove(object sender, MouseEventArgs e)`
    - `private void Grid_MouseUp(object sender, MouseButtonEventArgs e)`
- Verifica si los puntos si los puntos de la lista de puntos están contenidos en el rectángulo que el usuario ha creado, y si es así devuelve verdadero, por lo que negando la condición se obtienen los NO contenidos.
    - `private bool IsInPolygon(Point[] rectangle, Point testPoint)`
- Gestiona la instanciación de las ventanas modales y de sus valores mediante métodos “Getters” y “Setters”, modificando a su vez la lista de puntos y las variables de aspecto.
    - `private void CDModal_Click(object sender, RoutedEventArgs e)`
- Modifica el valor de la variable “purgaON”.
    - `private void Purga_Click(object sender, RoutedEventArgs e)`
- Permite elegir el tipo de representación de los puntos.
    - `private void Representacion_Click(object sender, RoutedEventArgs e)`
- Asigna los puntos de la lista de puntos a la gráfica elegida como pares de valores a su “ItemSource”.
    - `private void LoadBarChartData()`
- Imprime un “MessageBox” con información de ayuda.
    - `private void Ayuda_Click(object sender, RoutedEventArgs e)`
- Gestiona la redimensión de la ventana.
    - `private void miCanvas_SizeChanged(object sender, SizeChangedEventArgs e)`

Ventanas modales

En este apartado se utilizarán abreviaturas para especificar qué métodos existen en cada tipo de ventana modal pues muchos son los mismos.

Las ventanas modales existentes son:

- Manual (M): Se utiliza para la gestión de los puntos por introducción manual.
- Aleatorio (A): Se utiliza para la gestión de los puntos por introducción por parámetros de la función polinómica.
- Trigonométrica (T): Se utiliza para la gestión de los puntos por introducción por parámetros de las funciones Seno, Coseno y Tangente.
- Opciones (O): Se utiliza para la gestión de la apariencia de la línea, los ejes y el fondo del canvas.

```
private void Lista_SelectionChanged(object sender, SelectionChangedEventArgs e)
```

(M)(A)(T) - Gestiona la visibilidad de los elementos y si están habilitados los botones de ordenar, modificar y eliminar.

```
private void Anadir_Click(object sender, RoutedEventArgs e)
```

(M)(A)(T) – Gestiona los parámetros Nombre, A, B, C (y D), rangoMin y rangoMax y añade el/los punto(s) a la lista de puntos local.

```
private void Aleatorio_Click(object sender, RoutedEventArgs e)
```

(M)(A)(T) – A los parámetros se les asigna un valor aleatorio y junto con los valores introducidos en rangoMin y rangoMax se añaden los puntos a la lista de puntos local.

```
private void Modificar_Click(object sender, RoutedEventArgs e)
```

(M) – Permite modificar el valor del punto seleccionado en la lista.

```
private void Eliminar_Click(object sender, RoutedEventArgs e)
```

(M)(A)(T) – Permite eliminar de la lista de puntos local los puntos seleccionados.

```
private void EliminarNombre_Click(object sender, RoutedEventArgs e)
```

(M)(A)(T) – Permite eliminar todos los puntos que coincidan con el Nombre escrito por el usuario en la casilla adyacente al botón.

```
private void Aceptar_Click(object sender, RoutedEventArgs e)
```

(M)(A)(T)(O) – Cierra la ventana modal con valor en “DialogResult” verdadero, por lo que añadirá los puntos a la lista de puntos de la “MainWindow” y redibujará la polilínea o la gráfica. En el caso de la ventana modal de opciones, obtiene los valores de las casillas y las asigna a las variables locales para luego poder ser recogidas mediante los métodos “Getter” en la “MainWindow”.

```
private void Cancelar_Click(object sender, RoutedEventArgs e)
```

(M)(A)(T)(O) – Cierra la ventana modal con valor en “DialogResult” falso.

```
private void Subir_Click(object sender, RoutedEventArgs e)
```

```
private void Bajar_Click(object sender, RoutedEventArgs e)
```

(M)(A)(T) – Permite reordenar los valores de la lista. Si el punto elegido es el primero sólo permitirá reordenar hacia abajo y si es el último, sólo hacia arriba.

```
private void lista_MouseDown(object sender, MouseButtonEventArgs e)
```

(M)(A)(T) – Permite deseleccionar el punto elegido si se presiona en cualquier parte de la "ListView" que no sea otro punto.

```
private void colorPickerEjes_SelectedColorChanged(object sender,
RoutedPropertyChangedEventArgs<Color?> e)
```

```
private void colorPickerLineas_SelectedColorChanged(object sender,
RoutedPropertyChangedEventArgs<Color?> e)
```

```
private void colorPickerBackground_SelectedColorChanged(object sender,
RoutedPropertyChangedEventArgs<Color?> e)
```

(O) – Permiten elegir el color de la línea, los ejes y el fondo de pantalla.

```
private void Predeterminado_Click(object sender, RoutedEventArgs e)
```

(O) – Permite poner al valor inicial (por defecto) todas las casillas.

```
private void inicioLinea_SelectionChanged(object sender, SelectionChangedEventArgs e)
```

```
private void finLinea_SelectionChanged(object sender, SelectionChangedEventArgs e)
```

(O) – Permiten elegir el inicio o fin de la línea, si es Ninguna, Triangular, Redondeada o Cuadrada.

Clases

Punto

La clase Punto hereda de “INotifyPropertyChanged” y tiene 3 campos: Nombre de tipo “String” ,y corX y corY de tipo “Double” con sus métodos “Getters” y “Setters”, teniendo este último un “Binding” o enlace con los campos de las “ListView” de adición de puntos.

Tiene un constructor y un método `private void OnPropertyChanged(string propertyname)` que se utiliza para asignar cada valor al campo de la “ListView”.

Polinomio

La clase Polinomio tiene los mismos campos que el usuario puede introducir en la ventana modal Aleatorio. Contiene el método “Getter” de la lista de puntos que se devolverá cuando se ejecute el método public void calcular() dentro del constructor.

Al constructor se le pasan por parámetro los campos debajo adjuntados para poder realizar el cálculo y obtener la lista de puntos.

```
private string name;

private double A;

private double B;

private double C;

private double D;

private double min;

private double max;

private double minCanvas = -9.0;

private double maxCanvas = 9.0;

private double paso;
```

Referencias y bibliotecas utilizadas

Bibliotecas utilizadas

Se han utilizado 2 bibliotecas para realizar este trabajo. Ambas instaladas con el gestor de paquetes NuGet de Visual Studio.

WPF Extended Toolkit: Esta biblioteca se utiliza para el control "ColorPicker", el cual ha se ha utilizado para usar seleccionar el color de las funciones en la tabla de funciones. Página de Github.

System.Windows.Controls.DataVisualization.Toolkit: Esta biblioteca se utiliza para representar los distintos tipos de gráficas con el control "Chart" y sus respectivas subvariantes. Página de NuGet.

Recursos bibliográficos

Microsoft – Documentación sobre las funciones de WPF y C#.

Stackoverflow – Algunas páginas sobre las cuestiones principales de funcionamiento:

- Eliminar múltiples elementos de una ListView
- Deseleccionar un elemento de una ListView
- Crear un área de selección con el ratón
- Verificar si un punto está contenido en un polígono