



UNIVERSIDAD DE SALAMANCA
FACULTAD DE CIENCIAS

EQUATION DRAWER APPLICATION

Héctor Chamorro Álvarez

Interfaces gráficas (WPF)

Xamo1998@usal.es

Tabla de contenido

Introducción	3
Entorno de desarrollo	3
Lenguaje utilizado	3
Objetivos de la práctica	3
Manual de Usuario	4
Funcionamiento	4
Funciones	6
Personalizar	6
Exportar imagen	7
Manual del programador	8
Diagrama de clases.....	8
Glosario de clases.....	10
Paquete ViewModel.....	10
Paquete Converter	10
Paquete Commands	11
Paquete Helpers.....	11
Paquete principal	12
Dificultades pasadas.....	13
Posibles mejoras.....	13
Funcionalidades que destacar.....	13
Referencias.....	14

Introducción

Entorno de desarrollo

Esta práctica ha sido desarrollada con WPF (Windows Presentation Foundation), una API para crear interfaces de usuario (UI) para aplicaciones de escritorio con el framework .NET

Personalmente, WPF me ha parecido un API bastante potente, y poder utilizar archivos xaml me parece un gran punto positivo ya que puedes separar la lógica de la interfaz de una manera muy eficiente. Aunque estos archivos cuando aportan realmente una mejora muy grande son cuando se utiliza el patrón MVVM (Model View ViewModel) ya que nos permite que toda la lógica de datos recaiga en los xaml haciendo así que el código sea mas legible debido a que hay menos.

Lenguaje utilizado

El lenguaje utilizado en esta práctica es *C#*, un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET.

Personalmente *C#* me ha parecido un buen lenguaje, ya que es bastante parecido a java y tiene mucha potencia.

Objetivos de la práctica

El objetivo principal de esta práctica es construir una aplicación que permita la representación gráfica de funciones.

Manual de Usuario

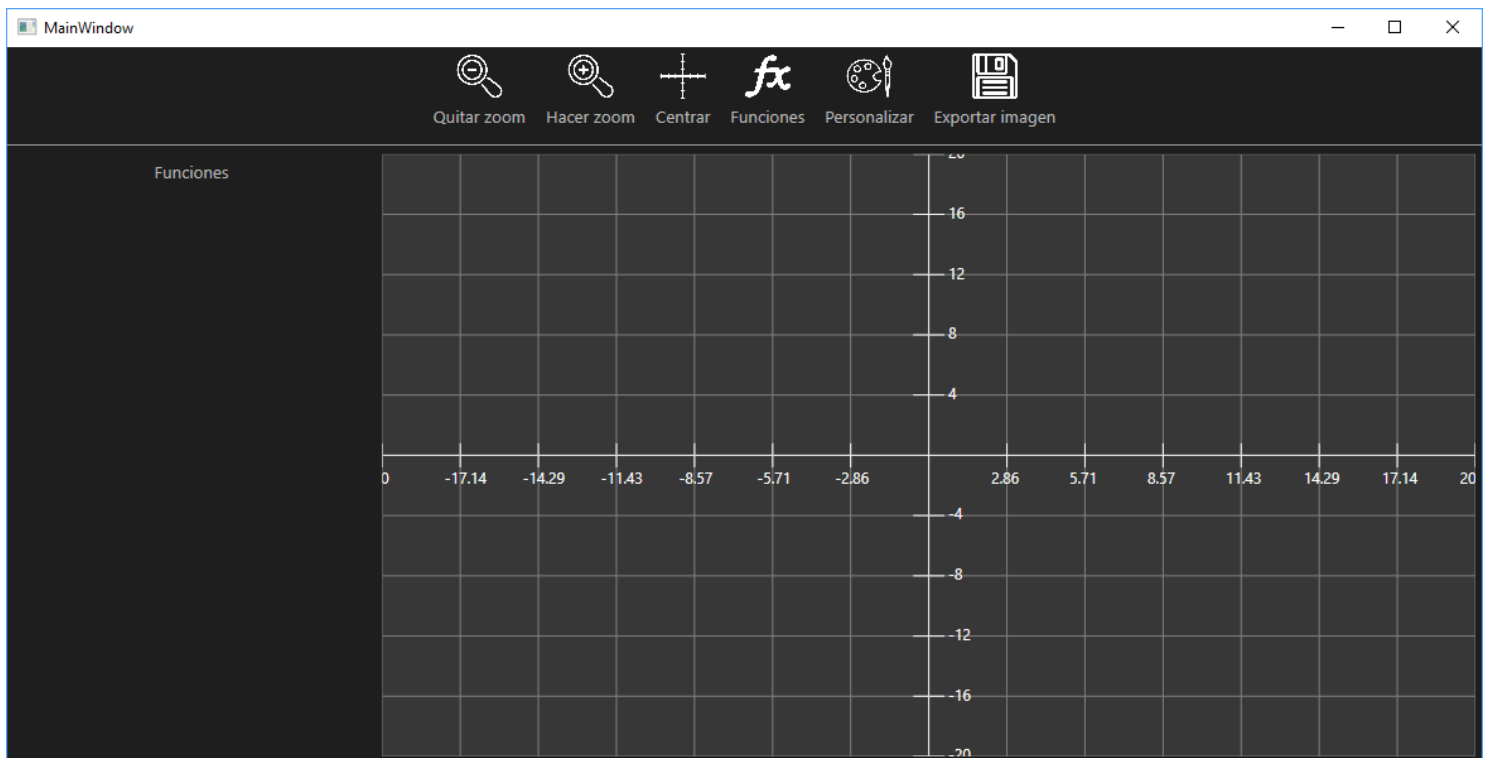
Funcionamiento

La funcionalidad mínima que posee la aplicación es representar los siguientes tipos de ecuaciones dados los valores de los parámetros:

- $a \cdot \sin(b \cdot x)$
- $a \cdot \cos(b \cdot x)$
- $a \cdot x^n$
- $a \cdot x + b$
- $a \cdot x^2 + b \cdot x + c$
- $a/(b \cdot x)$

Ya que se ha introducido un analizador de ecuaciones no es necesario escribir parámetros, tan solo basta con introducir la función a representar.

La pantalla principal es la siguiente:

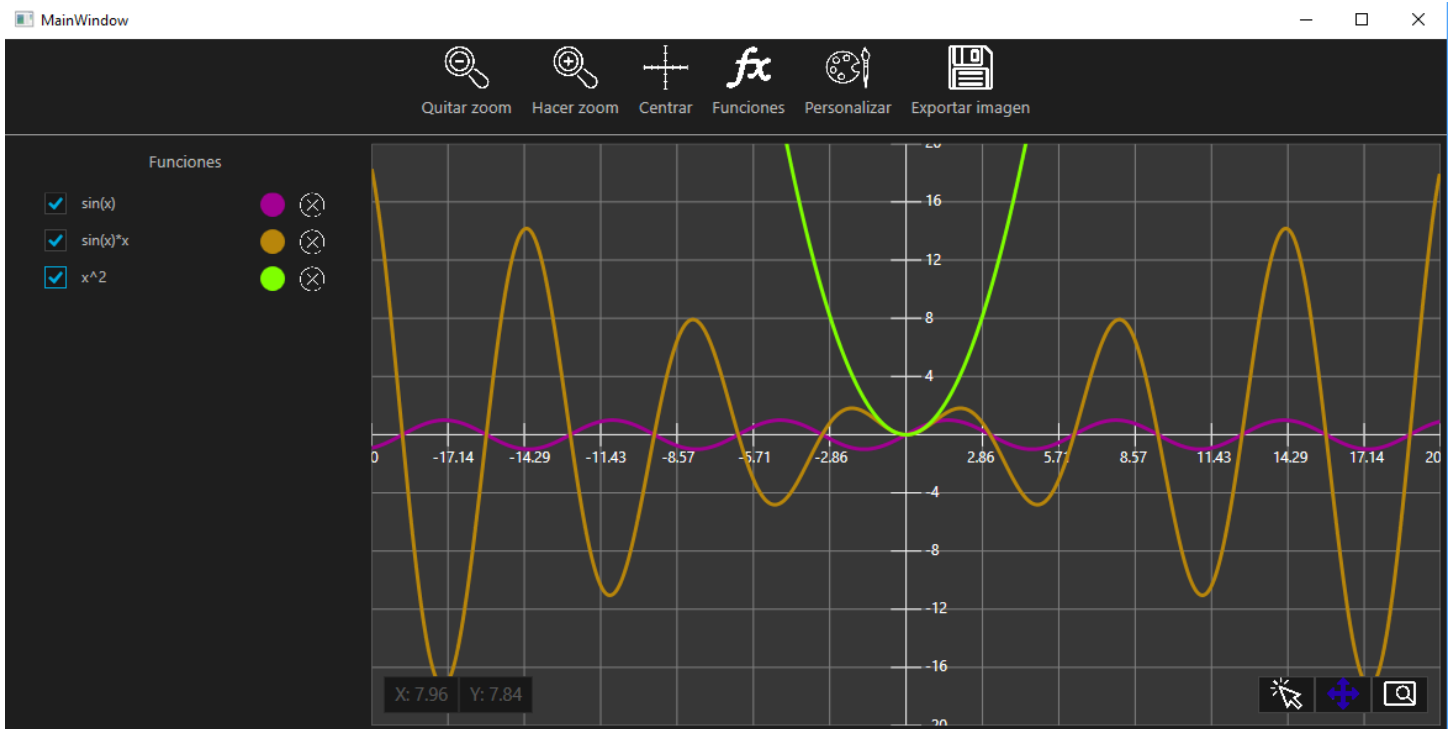


Las dos opciones más importantes del menú que se muestra son la de “funciones” y “personalizar” aunque procederemos a explicar todas las opciones:

- Quitar zoom: Aumenta el rango de representación al doble del valor actual.
- Hacer zoom: Reduce el rango de representación a la mitad del valor actual.
- Centrar: Muestra un rango de -20 a 20 en ambos ejes.
- Funciones: Pantalla que permite añadir funciones.

- Personalizar: Pantalla que permite personalizar tanto funciones como varios parámetros del lienzo donde se representan las funciones.
- Exportar imagen: Permite exportar el “lienzo” donde se representan las funciones a varios ficheros de imagen.

En la parte izquierda vemos las funciones que se están representando y si introducimos el ratón dentro del canvas podremos ver las coordenadas, así como la herramienta seleccionada:



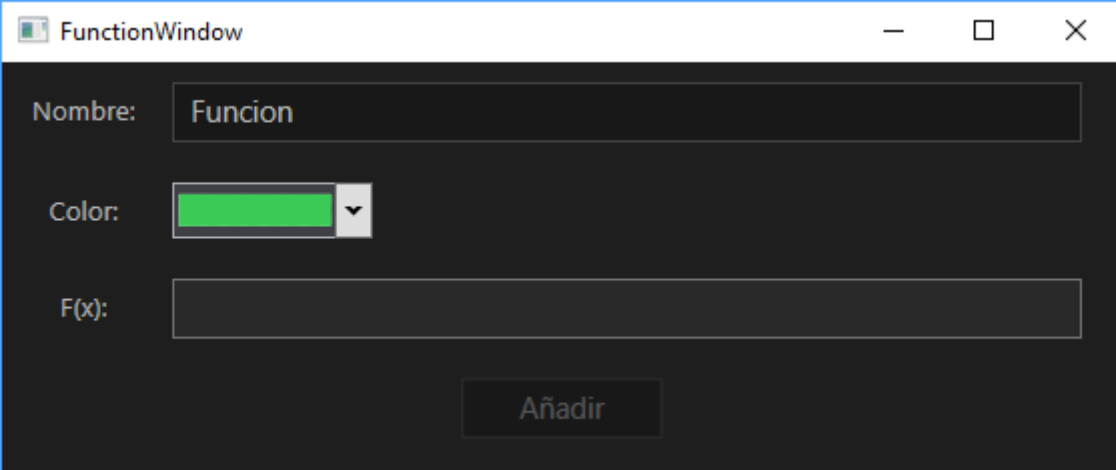
El icono seleccionado sirve para arrastrar y moverse dentro del canvas.

El icono a la derecha de este sirve para hacer zoom en una región específica:



Funciones

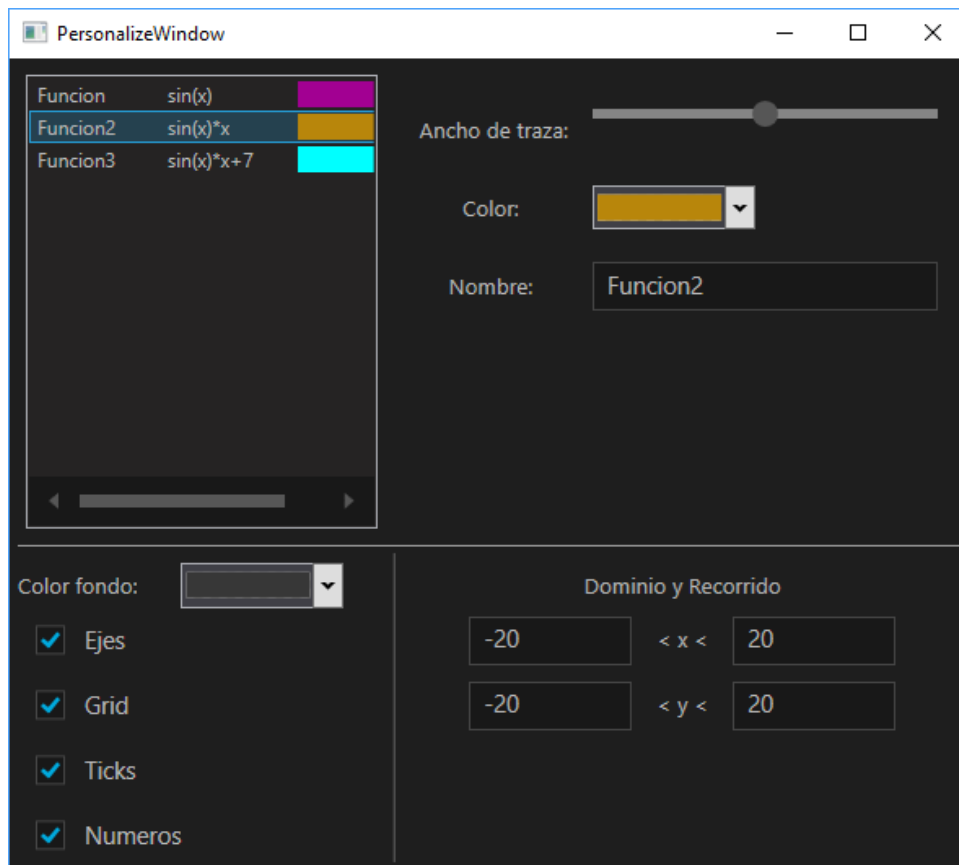
En esta ventana se podrá añadir cualquier función que desee el usuario. Primero se deberá seleccionar el nombre de la gráfica, el color y por último la función a representar. El botón añadir solo se activará cuando la función tenga un nombre y cuando la función escrita por el usuario pueda representarse.



Personalizar

En esta ventana podremos realizar varias funciones:

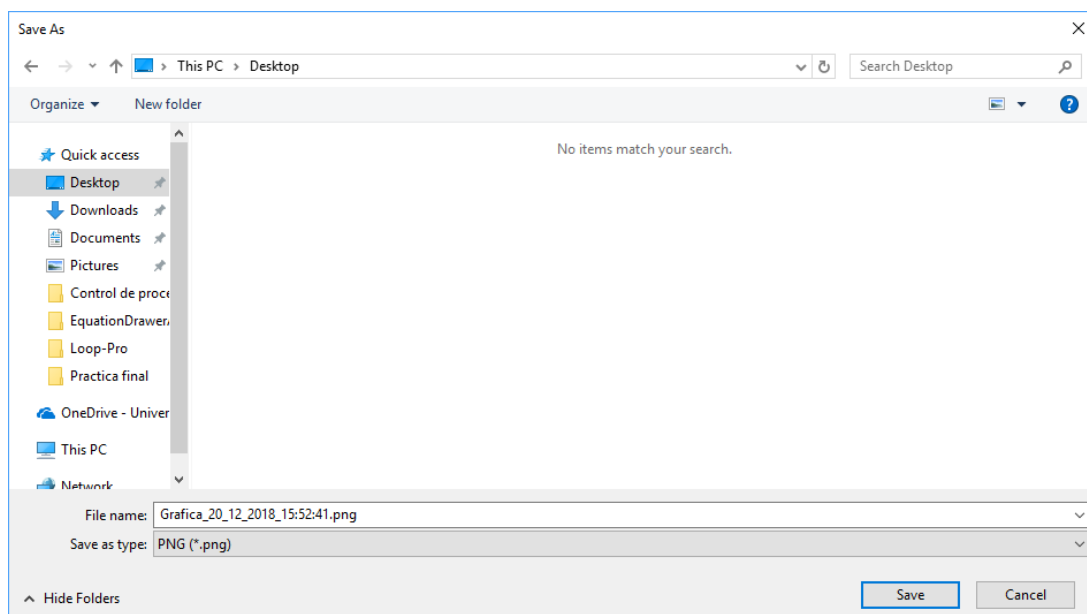
- Modificar atributos de cada ecuación:
 - Color
 - Ancho de traza
 - Nombre
- Modificar atributos de la vista principal:
 - Color de fondo
 - Ejes
 - Grid
 - Marcas numéricas
 - Números
- Modificar Dominio y recorrido de representación



Exportar imagen

En esta pestaña podremos exportar la imagen en varios formatos:

- PNG
- GIF
- JPEG
- JPG
- BMP

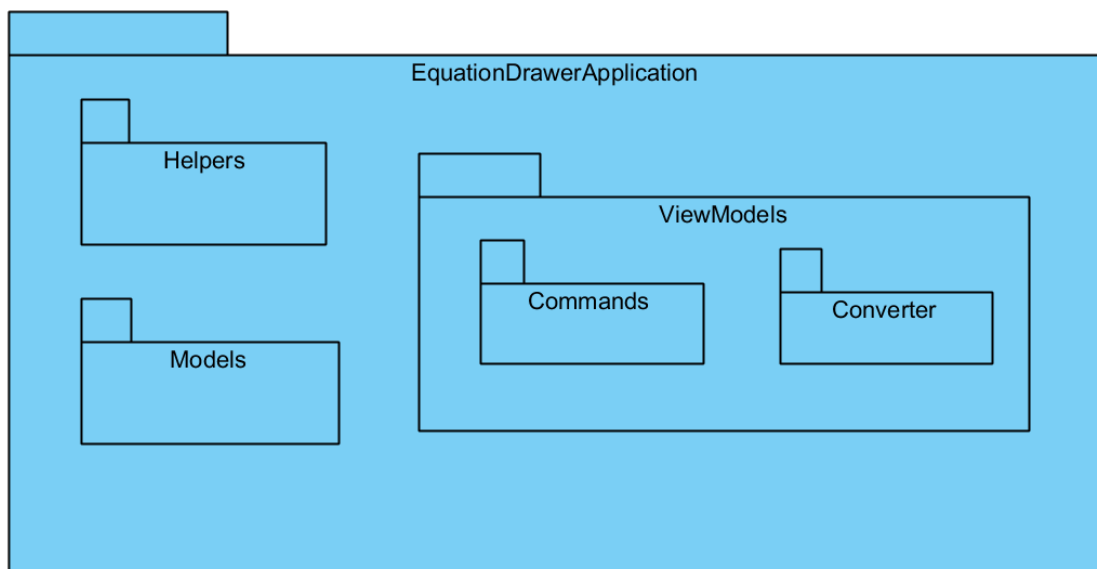


Manual del programador

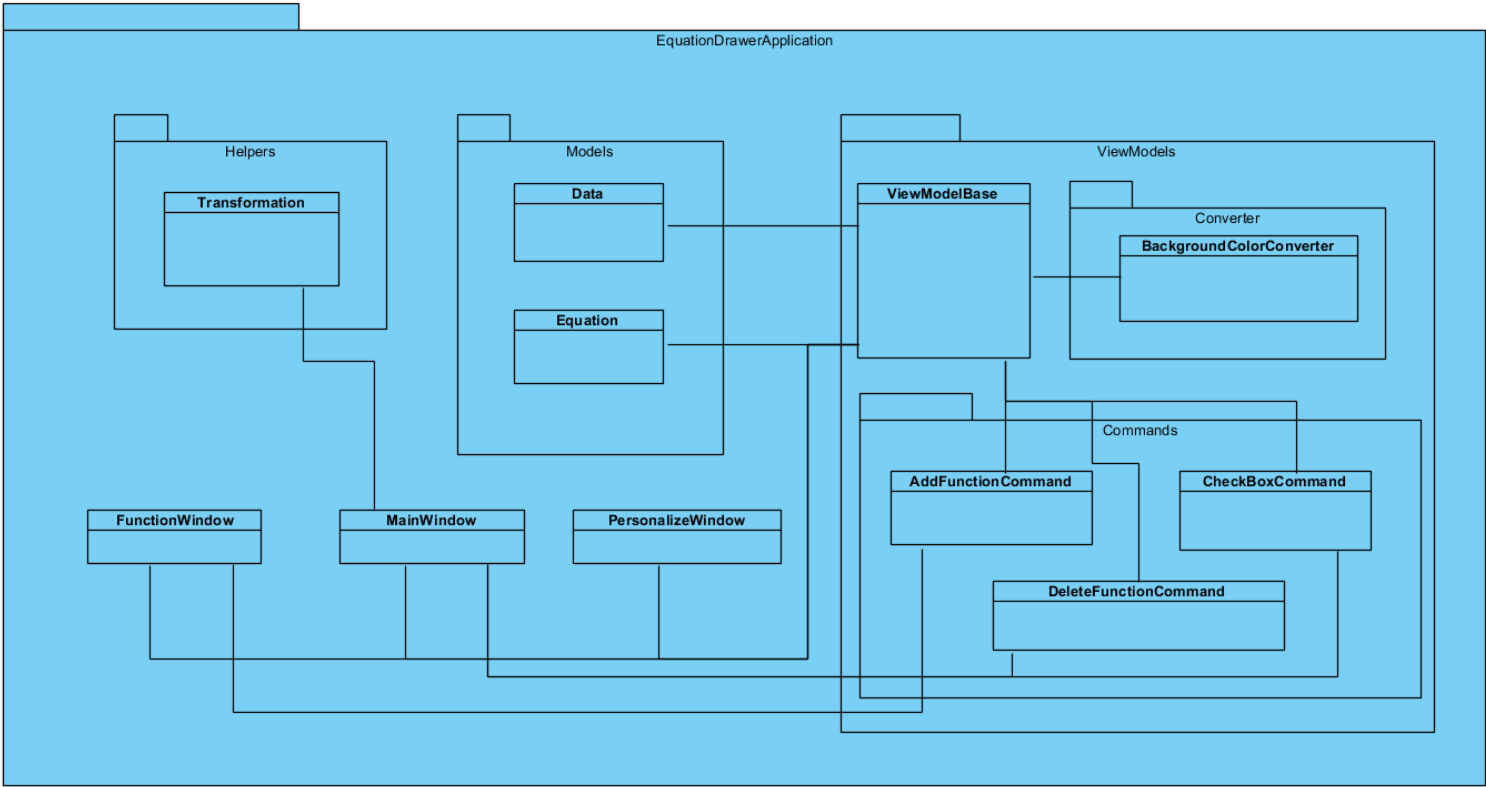
Diagrama de clases

Nuestro diagrama de clases consta de varios paquetes:

- EquationDrawerApplication: Paquete principal de la aplicación.
- Helpers: Paquete en que guardaremos clases de ayuda.
- Models: Paquete donde se guardará todos los datos de nuestra aplicación.
- ViewModels: Paquete en el cual guardaremos todo lo relacionado a los ViewModels y a su correcto funcionamiento:
 - Commands: Paquete para los comandos de la aplicación.
 - Converter: Paquete para los convertidores de la aplicación.



Debido al gran tamaño de los módulos los veremos por separado desde más abstracto a más concreto, es decir, primero veremos las clases que contienen estos paquetes y las relaciones entre ellas y después veremos los atributos, métodos, eventos... de cada clase. (Ver páginas siguientes):



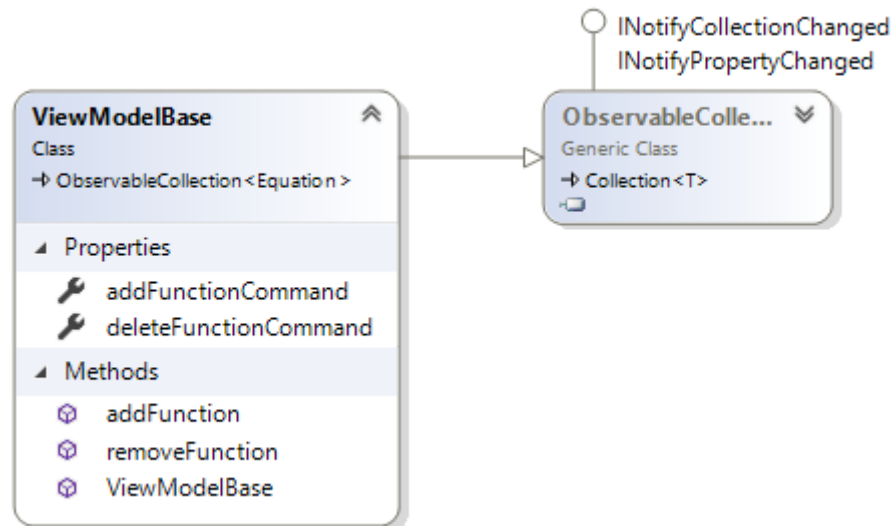
En las siguientes páginas procederemos a estudiar cada clase:

Glosario de clases

Paquete ViewModel

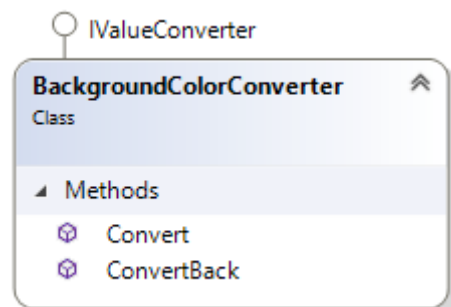
Este paquete contiene a su vez los paquetes Converter y Commands los cuales detallaremos mas adelante. Además de estos paquetes también tiene la clase ViewModelBase.

Esta clase sirve como enlace entre los objetos visuales y los datos del modelo.



Paquete Converter

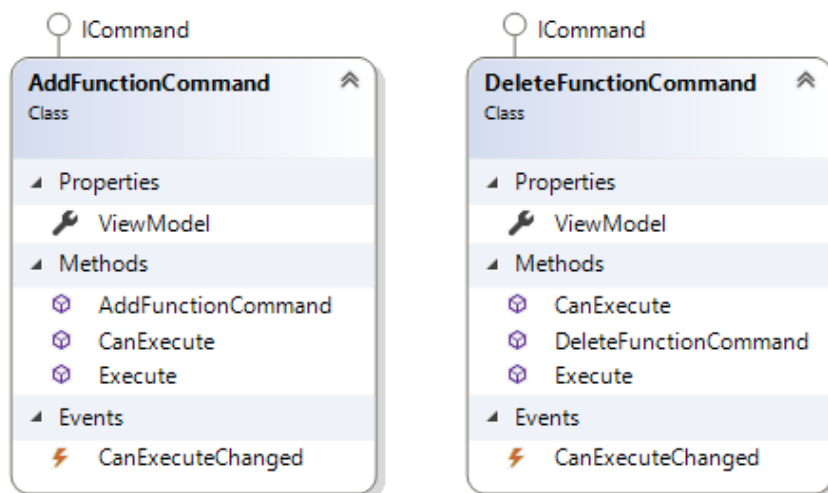
En este paquete nos encontramos con la clase *BackgroundColorConverter* la cual es utilizada para convertir el color que obtenemos del colorPicker a un color que le podamos asignar al color de fondo del canvas.



Paquete Commands

En este paquete nos encontramos 2 clases las cuales implementan la interfaz *ICommand*. Estas dos clases son utilizadas desde el código XAML para poder hacer un buen uso del patrón MVVM. Las clases son:

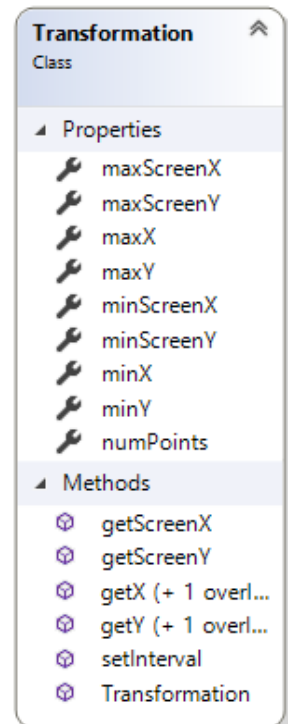
- **AddFunctionCommand**: Este comando es utilizado para decir si el botón de añadir función está habilitado o no. La manera en la que el método *CanExecute* decide es la siguiente:
 - Comprobamos que el atributo nombre es distinto de nulo o vacío.
 - Comprobamos que la ecuación introducida por el usuario es correcta.
- **DeleteFunctionCommand**: Este comando es utilizado para poder eliminar una ecuación cuando se llama a este comando.



Paquete Helpers

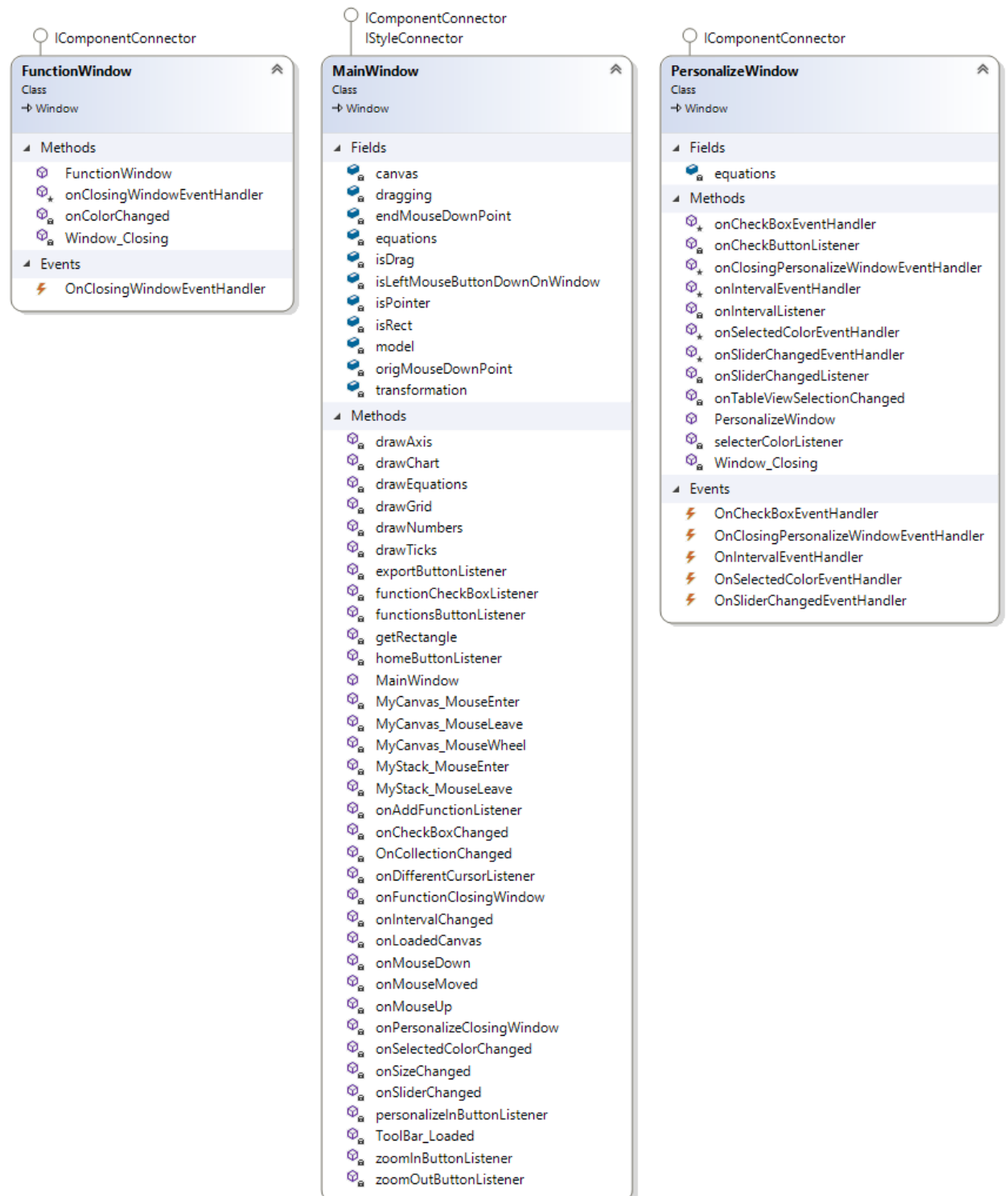
Nos encontramos la clase *Transformation* la cual es utilizada para la representación de puntos en la pantalla. Dado un punto en coordenadas de pantalla nos lo convierte a puntos de nuestro canvas en concreto y, al contrario.

Esta clase es usada por la ventana principal para dibujar las funciones, realizar el zoom, el scrolling y todo lo que conlleve calcular puntos.



Paquete principal

En el paquete principal donde tenemos el resto de paquete también tenemos las 3 ventanas mencionadas anteriormente:



Dificultades pasadas

1. Implementación del patrón MVVM
 - Debido a la poca información dada acerca de este potente patrón tuve que ver tutoriales de como funcionaba este patrón para poder implementarlo correctamente.
2. Exportar la imagen
 - Me costó bastante trabajo dar con un código que permitiera exportar un canvas a un fichero de imagen ya que lo más próximo que encontré era una imagen de toda la ventana en blanco salvo el canvas.

Posibles mejoras

Dadas las dificultades mencionadas anteriormente, esta práctica podría haberse mejorado aún más, pero esto no ha sido posible por falta de tiempo.

A continuación, enumero algunas de las posibles mejoras a realizar:

- Limpieza de código
- Documentación de código
- Portabilidad a tablets
- Mas funcionalidades
- Guardar funciones en ficheros
- Intersecciones de funciones

Funcionalidades que destacar

Dado que se trata de un trabajo con una funcionalidad mínima vamos a destacar algunas funcionalidades especiales que se han implementado de manera opcional para incrementar el potencial de este representador gráfico de funciones:

1. Intérprete de ecuaciones
2. Hacer zoom dibujando un rectángulo
3. Hacer zoom con la ruleta del ratón
4. Rastreador de posición del ratón
5. Modificar intervalos en ambos ejes, así como el centro
6. Exportar a varios tipos de ficheros
7. Interfaz limpia, profesional y sencilla

Referencias

- Tema Metro oscuro:
 - <http://brianlagunas.com/free-metro-light-and-dark-themes-for-wpf-and-silverlight-microsoft-controls/>
- Eliminar flecha de Toolbar:
 - <https://stackoverflow.com/questions/4662428/how-to-hide-arrow-on-right-side-of-a-toolbar>
- Extended Toolkit del que se ha utilizado el ColorPicker:
 - <https://github.com/xceedsoftware/wpftoolkit>
- Guardar canvas como imagen:
 - <http://tutorialgenius.blogspot.com/2014/12/saving-window-or-canvas-as-png-bitmap.html>
 - <https://stackoverflow.com/questions/21411878/saving-a-canvas-to-png-c-sharp-wpf>
- Interprete de ecuaciones:
 - <https://github.com/mariuszgromada/MathParser.org-mXparser>
- Iconos:
 - <https://www.flaticon.com/authors/smashicons>