

Primera práctica  
Funciones Matemáticas

Luis Blázquez Miñambres (i0910465)

12 de Noviembre de 2018

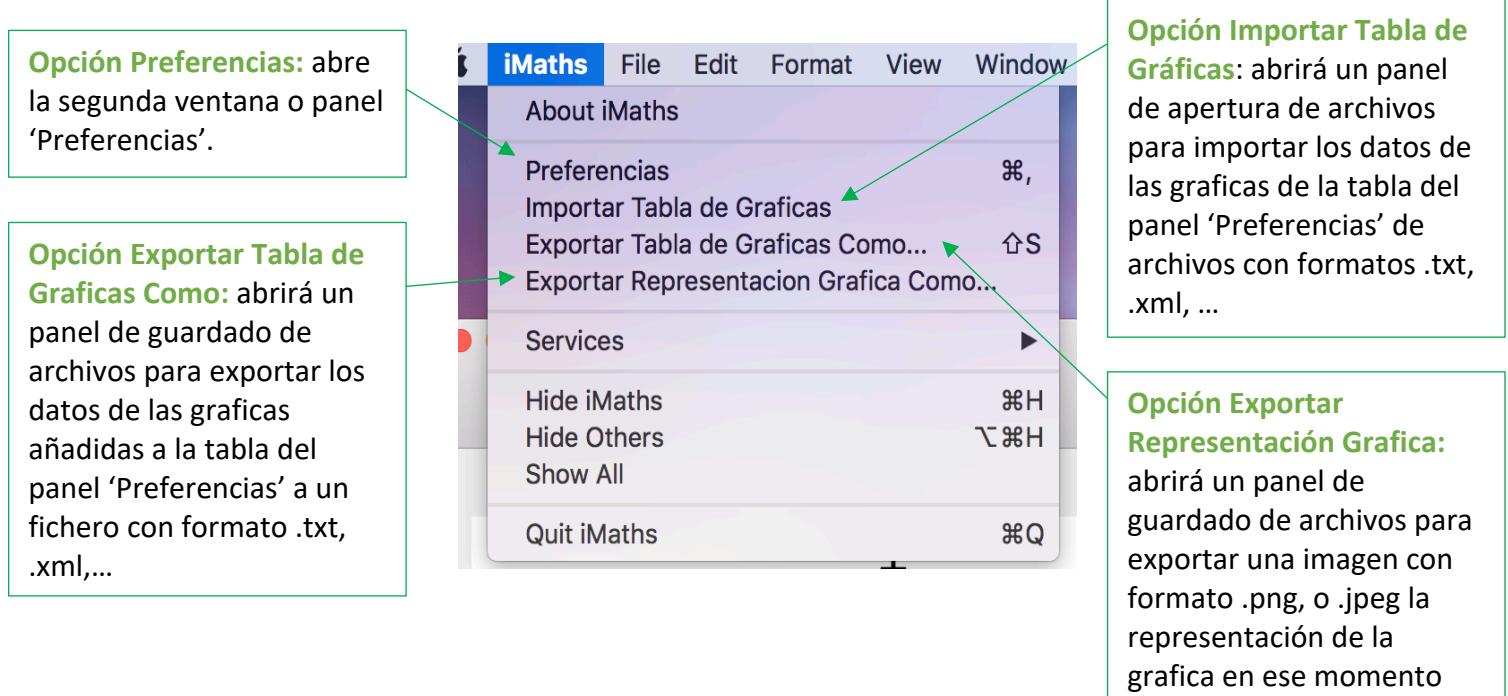
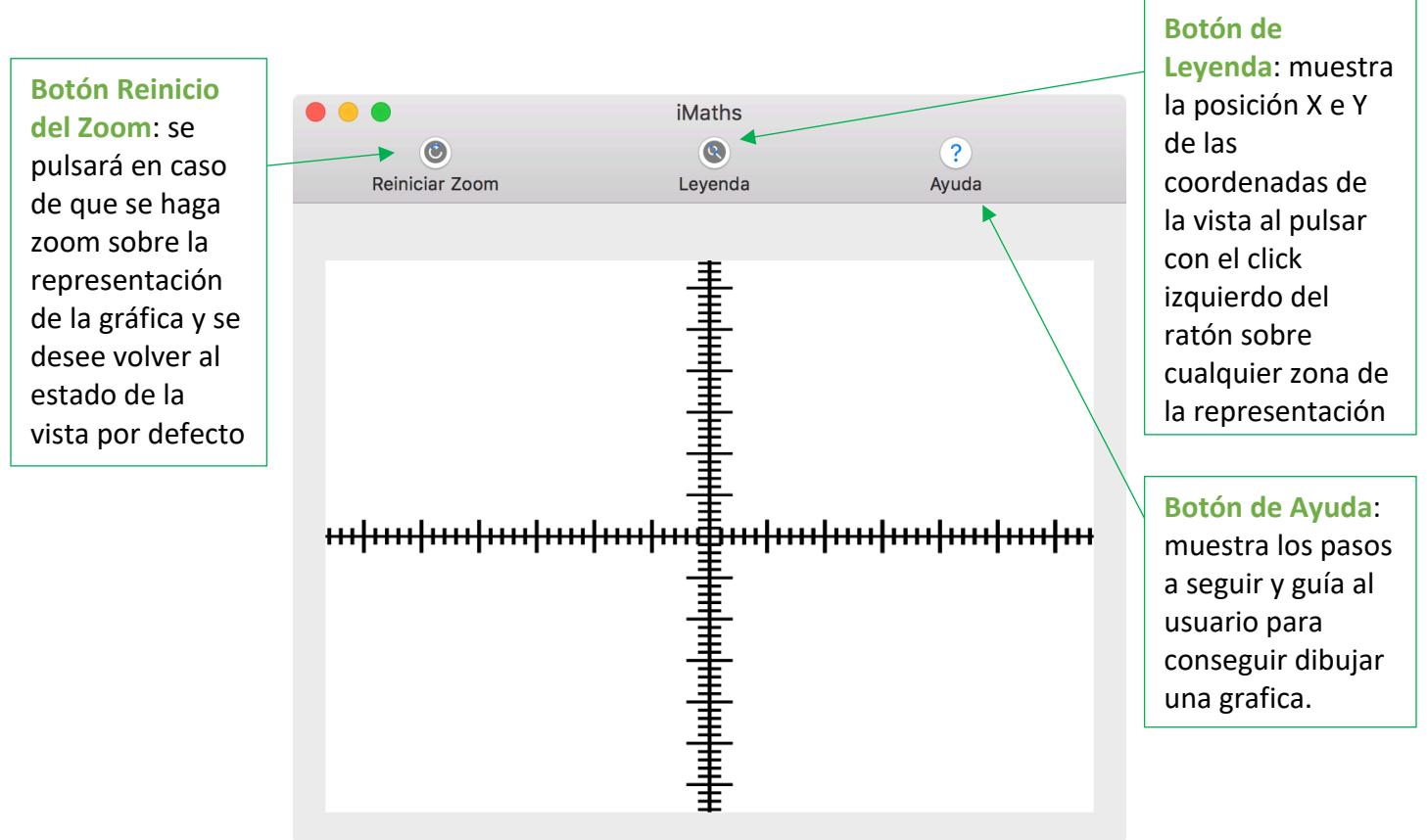
UNIVERSIDAD DE SALAMANCA

---

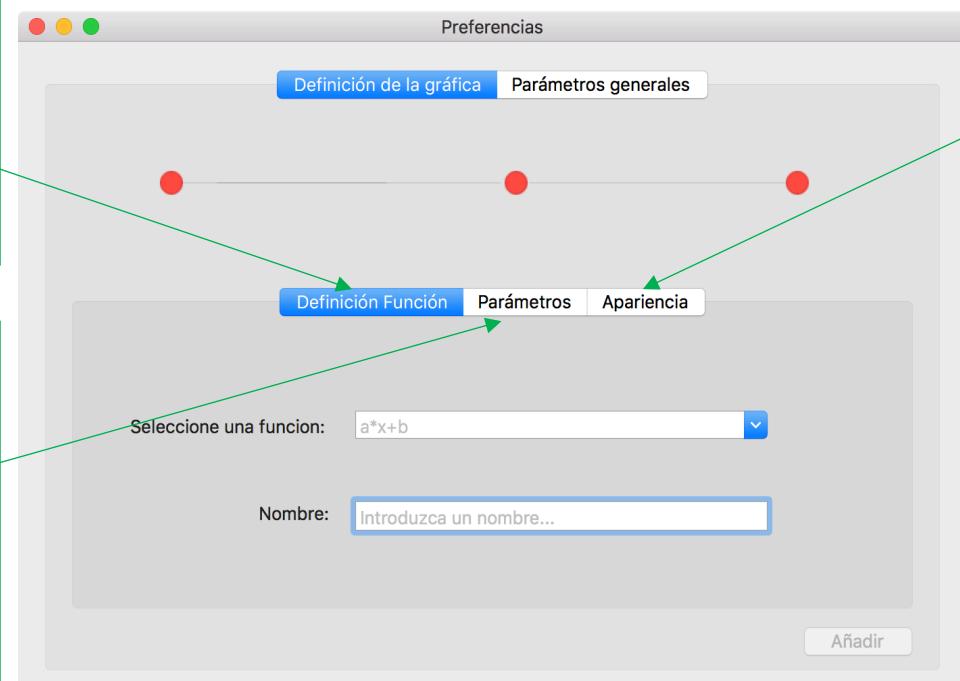
## 1. Manual de Usuario

La aplicación iMaths se compone de tres ventanas:

- **Menú principal:** aquí se mostrará la representación de las gráficas que se dibujen



- **Preferencias**: este menú incluye dos submenús dentro de él.
  - **Definición de la gráfica**: aquí se añadirán todos los parámetros necesarios para crear y añadir una nueva gráfica a la tabla

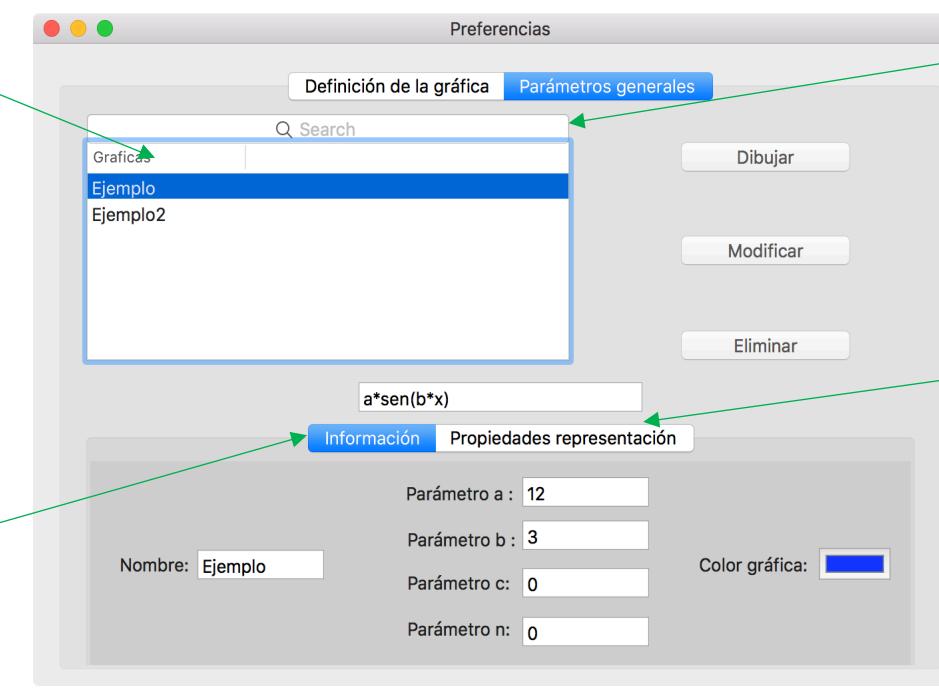


**Definición Función:** en este submenú se añadirán una de las funciones disponibles y un nombre para la gráfica

**Parámetros:** en este submenú se añadirán los valores para los parámetros a, b, c o n de cada función dependiendo que parámetros contenga la función seleccionada

**Apariencia:** en este submenú se seleccionará el color que tendrá la gráfica (Puede dejarse el que está por defecto)

- **Parámetros generales**: aquí se mostrará una tabla con todas las gráficas añadidas y listas para dibujar, así como la información de cada una, la posibilidad de modificar, eliminar o dibujar una de ellas o varias a la vez (manteniendo pulsado el botón *CMD+seleccionar* todas las que se deseen dibujar) y marcar los límites de dibujado.



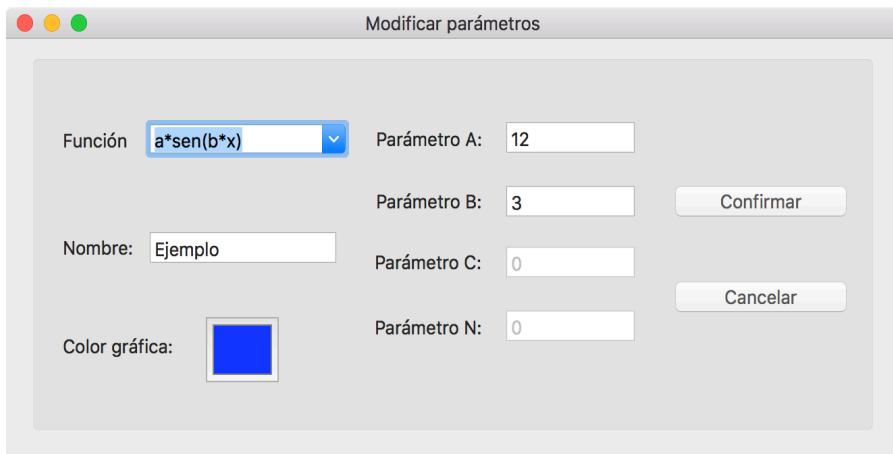
**Tabla de gráficas:** aquí estarán las graficas que se podrán dibujar modificar, dibujar o modificar. Así como ordenar por orden alfabético pulsando en la cabecera de la columna

**Buscador:** aquí se introducirá un nombre a buscar entre las gráficas añadidas. Sirve como un filtrador

**Información:** en este submenú se mostrará toda la información correspondiente a la grafica seleccionada en la tabla

**Propiedades representación:** en este submenú se introducirán, si se desea, los valores de los limites de representación X e Y

- **Panel de modificación de parámetros:** una vez seleccionada una gráfica de la tabla, y sucesivamente pulsado el botón ‘Modificar’ del panel ‘Preferencias’ se abrirá automáticamente un panel donde se podrá modificar cualquier casilla de información de la gráfica.



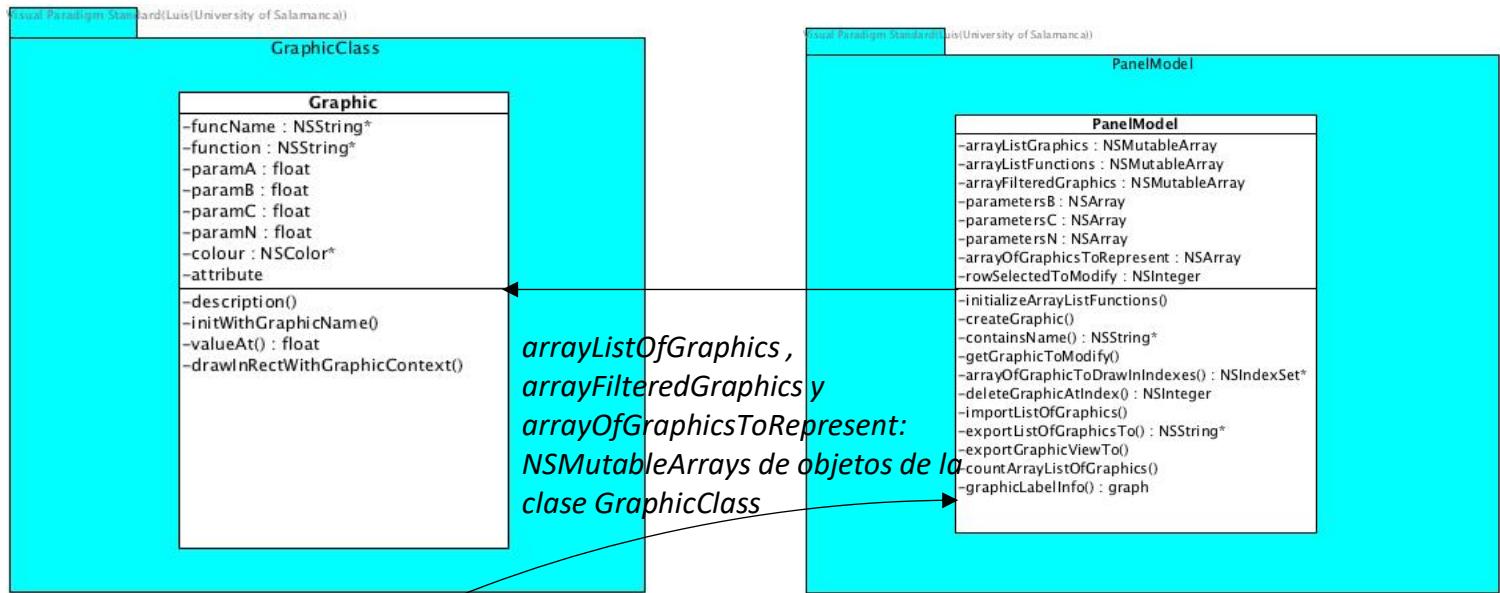
## **INFORMACIÓN ADICIONAL:**

- En las casillas de los parámetros A, B, C y N sólo podrán introducirse números reales positivos o negativos, cualquier otro carácter será rechazado.
- Las casillas de parámetros A, B, C y N se activarán según la función seleccionada. Por ejemplo: si seleccionamos la función  $a^*x^n$ , la aplicación solo activará las casillas A y N.
- Se podrá hacer zoom sobre la representación de la gráfica, haciendo clic con el botón izquierdo, arrastrando sobre la vista y soltando el botón del ratón. En caso de querer volver al estado de zoom por defecto de la representación gráfica, se pulsará el botón ‘Reiniciar Zoom’ anteriormente descrito.
- No se podrá exportar la tabla de gráficas hasta que no se añada al menos una gráfica en la tabla del panel ‘Preferencias’.
- Si se desea cambiar los límites de dibujado de la gráfica sobre los ejes X e Y se tendrá que introducir los nuevos valores de los límites y después pulsar el botón de dibujado ‘Dibujar’

## 2. Manual del programador

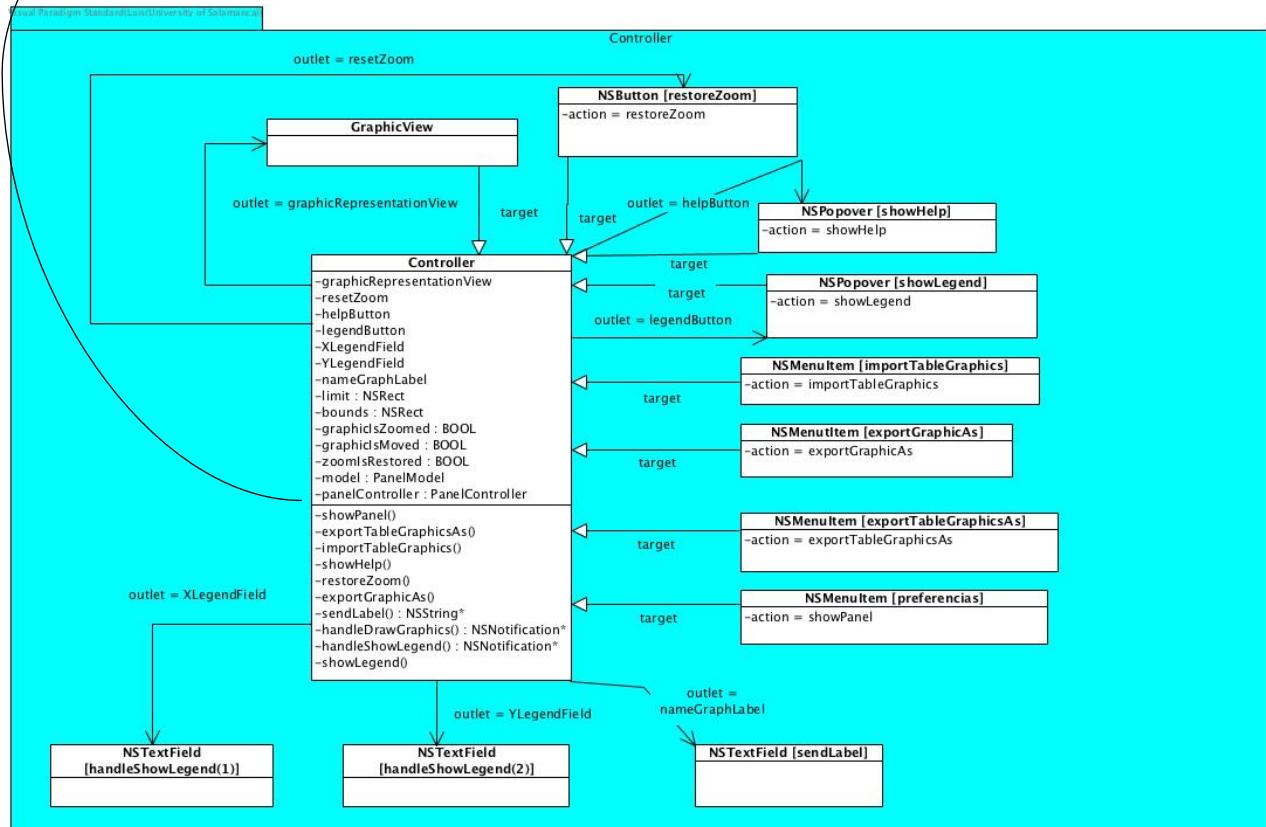
### CLASE GRÁFICA

### MODELO



*model : instancia de PanelModel*

### CONTROLADOR



### ❖ Métodos importantes (GraphicClass)

- ***drawInRect: withGraphicContext: andLimits: isZoomed : withMovement: w: h:***: éste método dibuja la gráfica o gráficas seleccionadas para dibujar recogiendo los datos necesarios para multiplicar la vista por la matriz de transformación afín correspondiente en cada caso, es decir, si se acaba de dibujar la gráfica las coordenadas X e Y de la vista se trasladaran al punto medio de los límites de la vista, tomando el (0,0) desde el punto central del CustomView con una escala predeterminada [1]. En otro caso, si se realiza la acción de zoom sobre la vista, haciendo clic con el botón izquierdo del ratón y arrastrando sobre la vista, se multiplicará por una matriz que trasladará las coordenadas X e Y al punto medio entre el punto de la vista donde se hizo clic y el punto en el que ,tras arrastrar, se soltó el botón izquierdo del ratón, con una escala fija de 0.2 puntos de acercamiento de Zoom cada vez que se realiza la acción [2].  
Además , el programa permitirá moverse a través de la vista con las flechas de dirección del teclado únicamente cuando se haya hecho zoom sobre la misma [3]; por lo tanto cada vez que se haga esta acción, la matriz de transformación deberá mantenerse constante. En cuanto a los límites, si se introducen valores distintos de 0 en el panel de preferencias, estos cambiarán los valores del array estático *funcBezier* [4] declarado globalmente arriba de la clase, que indica los valores de representación mínimos y máximos de X e Y por defecto (-10, -10, 20, 20).

```

if (!zoom){
    zoomQuant = 0;
}

NSLog(@"Matriz de transformación afín creada (GraphicClass)");
NSAffineTransform *tf = [NSAffineTransform transform];
tf = [NSAffineTransform transform];
// 2º Mult* por la matriz de Transformación Afín (Coloca la x e y en el (0,0) con respecto a la gráfica
[tf translateXBy:b.size.width/2;
     yBy:b.size.height/2];
// 1º Ancho y Alto / Escala (funcRect)
[tf scaleXBy:b.size.width/funcRect.size.width;
     yBy:b.size.height/funcRect.size.height];
[tf concat];

} else { // Si se realiza ZOOM sobre la vista
    if (!move)
        zoomQuant += 0.1; [3]

    NSLog(@"Matriz de transformación afín de ZOOM creada (GraphicClass)");
    NSAffineTransform *tfZoom = [NSAffineTransform transform];
    NSLog(@"Punto x: %f Punto y: %f ZoomQuant: %f", width, height, zoomQuant);
    [tfZoom translateXBy:width;
                 yBy:height];
    [tfZoom scaleXBy:width*zoomQuant
                 yBy:height*zoomQuant];
    [tfZoom concat];
}
}

limitOfDrawing.origin.x = funcRect.origin.x;
limitOfDrawing.origin.y = funcRect.origin.y;
limitOfDrawing.size.width = funcRect.size.width;
limitOfDrawing.size.height = funcRect.size.height;

NSLog(@"Limits: oX: %f oY: %f width: %f height: %f", limit.origin.x,
      limit.origin.y,
      limit.size.width,
      limit.size.height);
NSLog(@"Anterior FunRect: oX: %f oY: %f width: %f height: %f", funcRect.origin.x,
      funcRect.origin.y,
      funcRect.size.width,
      funcRect.size.height);

// Se cambia el funcRect si se introducen los límites [4] en el panel 'Preferencias'
if (limit.origin.x != 0)
    limitOfDrawing.origin.x = limit.origin.x;

if (limit.origin.y != 0)
    limitOfDrawing.origin.y = limit.origin.y;

if (limit.size.width != 0)
    limitOfDrawing.size.width = limit.size.width;

if (limit.size.height != 0)
    limitOfDrawing.size.height = limit.size.height;

```

### ❖ Métodos importantes (PanelModel)

- ***arrayOfGraphicToDrawIndexes:***: recoge un set con los índices de las gráficas del array de gráficas de la tabla del panel ‘Preferencias’ que se han seleccionado para dibujar, y los añade a otro array mutable para manejarlos más fácilmente.

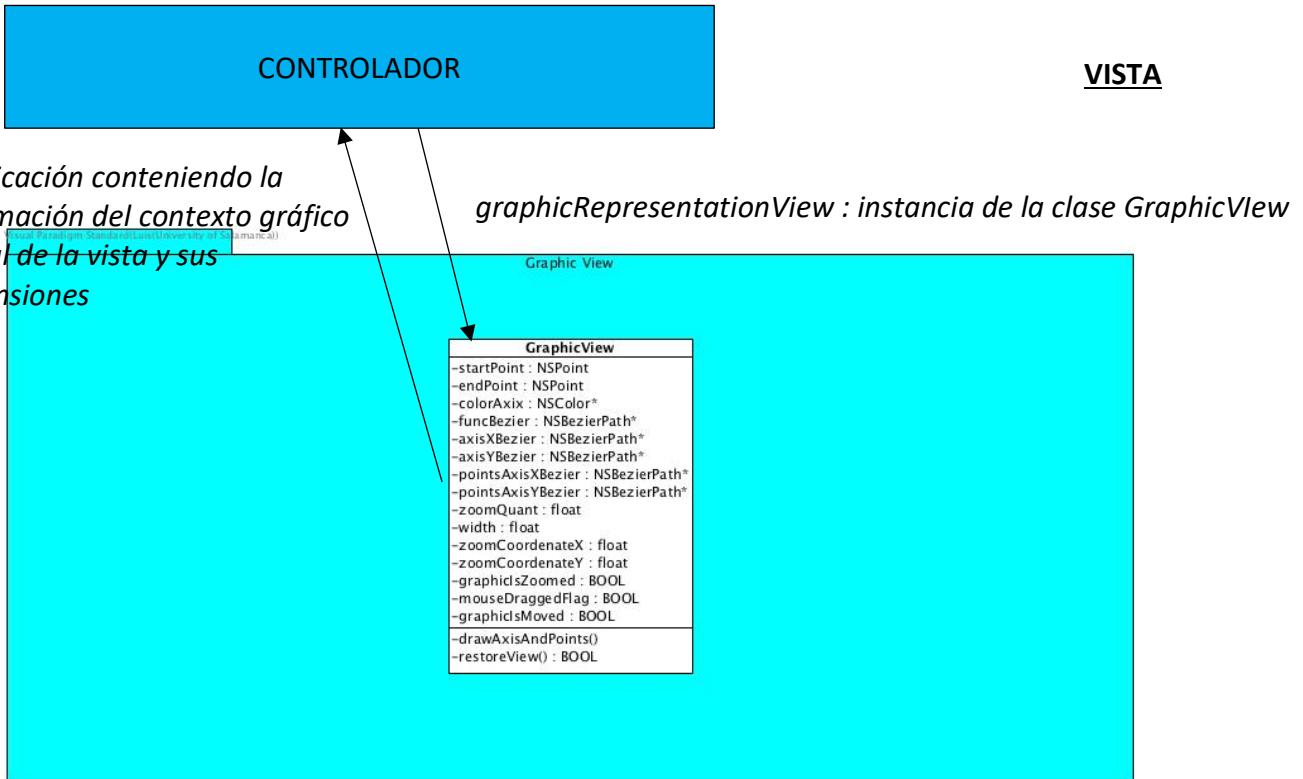
**IMPORTANTE SOBRE EXPORTAR E IMPORTAR:** no borrar .entitlements del proyecto pues para exportar e importar fue necesario que cambiase la propiedad de ficheros de Apple que viene por defecto en los proyectos de Xcode de com.apple.security.files.user-selected.read a com.apple.security.files.user-selected.read-write para permitir a la aplicación no solo la lectura de datos sino la escritura y sobreescritura de ficheros .

- ***exportListOfGraphicsTo:***: exporta la lista de objetos gráficas contenidas en la tabla del panel ‘Preferencias’ a un fichero de texto plano a elegir (txt, xml, csv o log). Se abrirá un panel de guardado (el de por defecto de macOS) para elegir la ruta en la que guardar el fichero; éste a su vez contendrá en cada fila una cadena con todos los atributos de un objeto gráfica separados por un delimitador '#'.
- ***exportGraphicView:***: a través de un panel de guardado se guardará en formato de imagen elegido (png, jpeg o bmp) el contenido de la vista pasada como parámetro en ese instante. Esto se consigue bloqueando la representación del CustomView para renderizar los bordes de la vista en su contexto gráfico actual y conseguir su contenido. Este contenido será tratado como un conjunto de bytes de la clase *NSData*, que será comprimida y reducida a imagen a través de la representación en formato TIFF a través de la representación de un mapa de bits de la clase *NSBitmapImageRep*.
- ***importListOfGraphics:***: importa o recoge la información contenida en un fichero de texto plano cuyas extensiones o formatos permitidos se definen un array que se incluirá en el ‘Open Panel’ o panel de apertura, de manera que se recoge de cada fila toda la información de un objeto gráfica y se devuelven todas las gráficas recogidas en un array mutable.

**NOTA:** al recoger la información de cada gráfica, el delimitador que he determinado para delimitar las distintas filas y, por tanto, los distintas gráficas recogidas es ‘\n’, de modo que tengo puesto una condición *if-else* dentro del bucle for que va recorriendo cada fila del fichero para que se salte el último objeto recogido (count = i+1) ya que recoge una línea vacía al hacer \n y provoca un error de ejecución.

#### ❖ **Métodos importantes (Controller)**

- ***showPanel:*** método a través del cual se manda la instancia del modelo al controlador del panel de preferencias, la clase ‘*panelController*’ , a través de un objeto de la clase *NSDictionary* y se lanza la ejecución de este.
- ***handleDrawGraphics:*** manejador que es llamado cuando se pulsa el botón ‘Dibujar’ del panel Preferencias y se manda una notificación al Controlador que recoge la información de los límites de representación de la gráfica para, una vez recibidos los valores de los límites, forzar la llamada al método de dibujado *drawRect con el método setNeedsDisplay*. Éste método , desde la vista, mandará una notificación al controlador con la información de las dimensiones y contexto gráfico de la vista en el momento de dibujar que recogerá este mismo método. Una vez recogida esta información, se pasará a llamar al método de dibujado de los ejes en la vista y al método de la clase ‘*GraphicClass*’ que contiene la funcionalidad de dibujado de cada gráfica.



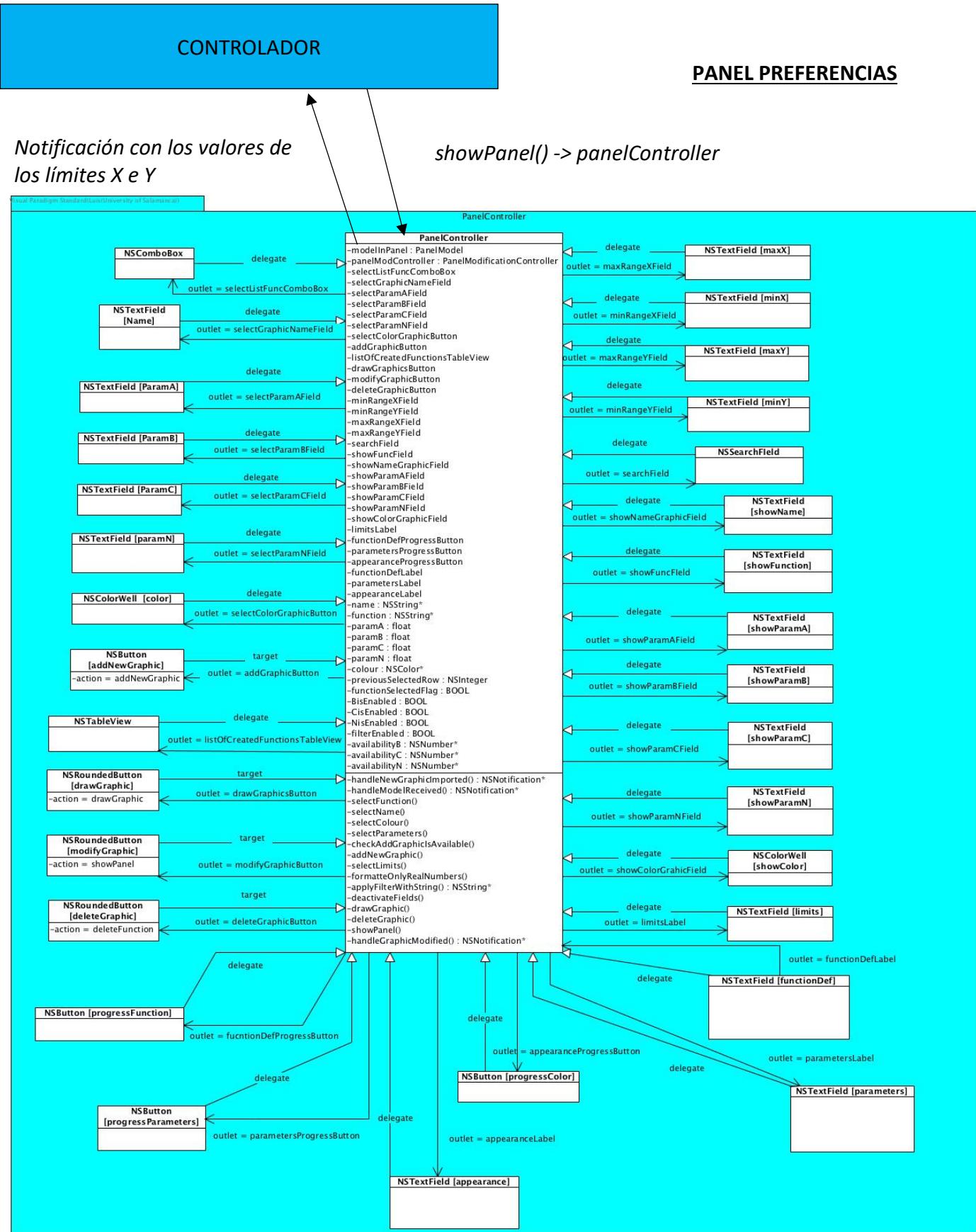
#### ❖ Métodos importantes (GraphicView)

- **isFlipped:** método de la clase NSView que indica si el sistema de coordenadas se ha invertido o no. Si se devuelve YES, como es mi caso, el sistema de coordenadas se invierte , de manera que se invierte a su vez el origen de coordenadas de la esquina inferior izquierda de la vista a la esquina superior izquierda, permitiendo que en el método *mouseDown* pueda multiplicar las coordenadas donde el usuario hace clic con el botón izquierdo del ratón sobre la vista por la matriz de transformación afín, pero en vez de concatenarla como en los casos anteriores, la invierto [*tf invert*] logrando sacar el punto con las coordenadas tomando como referencia el origen del sistema de coordenadas una vez multiplicada la matriz (es decir el punto 0,0 está en el centro de la vista) y no el origen tomando como referencia la vista (punto 0,0 está en la esquina inferior izquierda).

**NOTA:** en el manejador del Controlador que recibe las coordenadas de este punto para mostrarlas en la leyenda, *handleShowLegend* , tuve que negar el valor de la coordenada Y porque al invertir la matriz de transformación afín los valores de X se mantienen constantes independientemente de la matriz, pero las coordenadas Y se invierten (se pasa del punto 0,0 en la esquina inferior izquierda a la esquina superior izquierda, es decir, cambia el valor de Y); de manera que un punto que visualmente representa un valor Y positivo , al invertir la matriz , se representa como negativo y viceversa.

- **restoreView:** método que es llamado cuando se pulsa el botón '*Restaurar Zoom*' en el menú principal y devuelve la vista a su representación por defecto (antes de aplicarle zoom) definiendo los *bounds* o límites de la vista a su estado por defecto, con el origen de coordenadas siendo 0,0 en la esquina inferior izquierda.

- **keyDown:** método que es llamado cuando se produce un evento con una tecla del teclado. En esta aplicación he registrado dentro del método, las flechas de dirección del teclado de manera que el usuario pueda moverse alrededor de la vista con las teclas de dirección **únicamente** si se ha aplicado zoom sobre la vista.



❖ **Métodos importantes (PanelController)**

- ***formatterOnlyRealNumbers***: método llamado desde el método *selectLimits* que formatea la entrada de los textFields en los que sólo quiero que se introduzcan números reales positivos o negativos y rechace cualquier otro carácter. Añado un *NSCharacterSet* que toma la entrada de cada uno de los textField y separa cada uno de los componentes de la cadena introducida siguiendo el patrón del *NSCharacterSet*, si la cadena introducida no sigue el patrón, este valor es rechazado y eliminado. En caso contrario, si la cadena es válida, muestro la cadena en el textField.
- ***selectParameters***: método llamado desde el método *controlTextDidChange* que activa o desactiva los campos de los parámetros B,C y N dependiendo de si uno de estos parámetros está contenido o no en la función escogida del comboBox. El funcionamiento es el siguiente, se pasa la función por 3 bucles for que recorren 3 arrays estáticos inicializados en el *modelo* conteniendo cada uno de ellos una serie de subcadenas(\*b,b\*,+b,b+,...) en las que pueden encontrarse los parámetros B,C o N y comparándola con la función seleccionada para comprobar si alguna de esas subcadenas está contenida en la función, en caso afirmativo se activa el campo y se habilita la recogida de datos de ese campo. En caso contrario se mantiene desactivado el campo.
- ***tableView:\_sortDescriptorsDidChange***: aplica el descriptor de ordenamiento (según el nombre de la función en nuestro caso) que he aplicado a la columna de la tabla en el método *awakeFromNib* pero en este caso al array de gráficas creadas y añadidas a la tabla cada vez que el usuario pulsa sobre la cabecera de la columna ‘Gráficas’ para ordenar en orden ascendente o descendente el nombre de las gráficas.
- ***selectLimits***: método llamado desde el método *controlTextDidChange* que comprueba que si se cambia el valor de los campos de los límites de representación de los ejes X y Y (por defecto están a 0) se cumplan varias condiciones. En primer lugar que el valor de X e Y no sean el mismo (excepto en el caso de que sean 0) porque la gráfica no se representaría; y una vez cumplida esta condición que el valor máximo de X o Y sea mayor que el valor mínimo de X e Y porque provocaría una incoherencia en la representación de la gráfica a dibujar.
- ***controlTextDidChange***: método que es llamado cada vez que se produce un cambio dentro de los textField que delegan la clase ‘panelController’. En mi caso, cada uno de los parámetros necesarios para crear una gráfica y añadirla a la tabla serán controlados desde este método para no habilitar el botón ‘Añadir gráfica’ hasta que se cumplen estrictamente todos los requisitos requeridos para la creación de la gráfica (función seleccionada, nombre correcto no repetido, valores de los parámetros numéricos correctos).
- ***applyFilterWithString***: método llamado desde el método *controlTextDidChange* que filtra en la tabla, los nombres de las gráficas que coincidan con la cadena introducida en la barra de búsqueda. Si no se introduce ninguna cadena en la barra de búsqueda se muestran todas las gráficas añadidas; en caso contrario, he

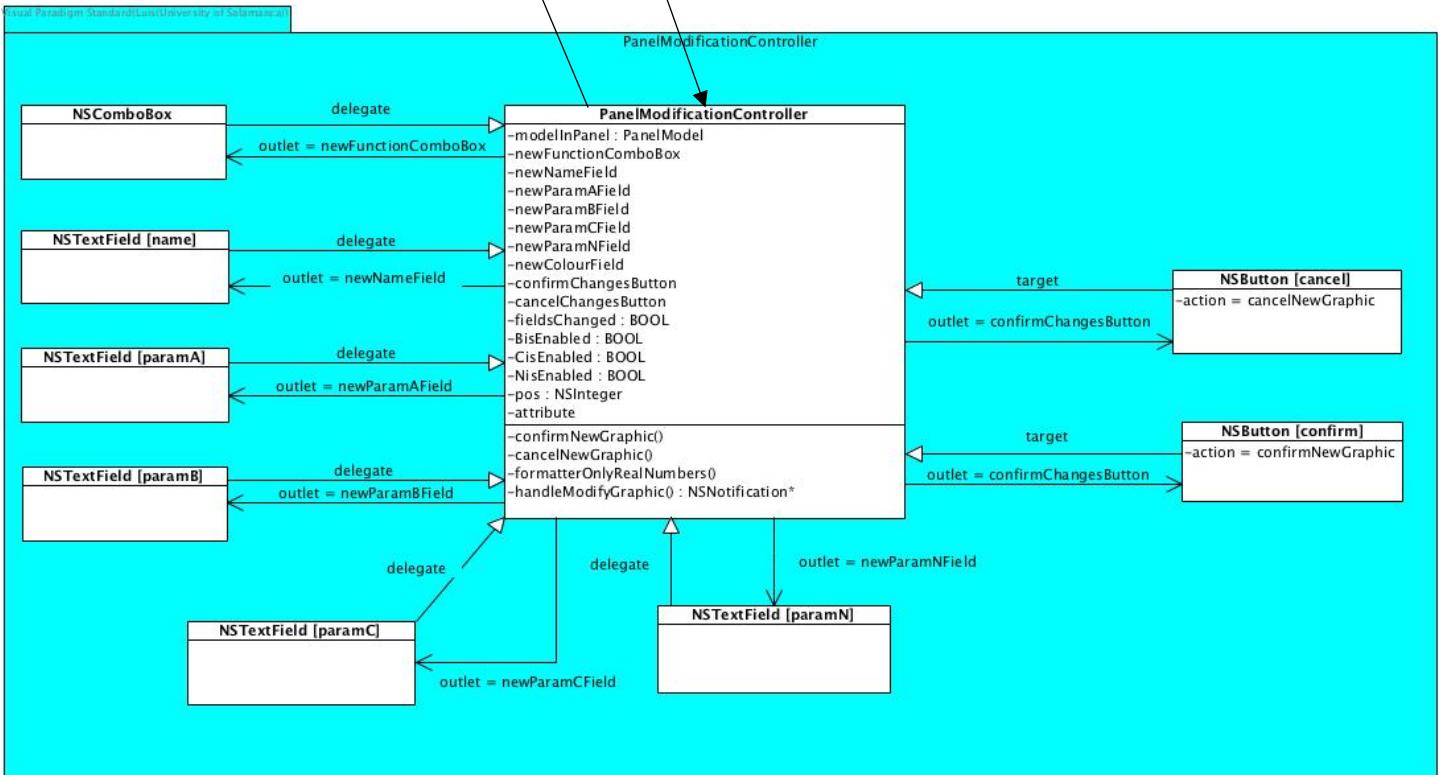
utilizado una expresión regular que aplico al array de gráficas de la tabla que devuelve un array que contiene las gráficas de la tabla que cumplen con esa expresión regular, que serán las gráficas que se muestren como resultado de haber hecho el filtrado.

- ***drawGraphic:*** método llamado cuando se pulsa el botón ‘Dibujar’ para representar una gráfica. Comprueba y toma los valores de los límites de X e Y por si el usuario ha introducido nuevos valores y envía una notificación conteniendo estos valores al controlador principal , la clase ‘Controller’, en un objeto de la clase NSDictionary.
- ***showPanel:*** si se ha seleccionado una gráfica de la tabla, y se pulsa el botón ‘Modificar’, se llama a este método que se encarga de lanzar y ejecutar el panel de modificación de gráficas, además de guardar en una variable de instancia del modelo el índice en el array de gráficas de la gráfica que se va a modificar (para no tener que pasar la gráfica entera a través de la notificación y obtenerla a través del modelo y conseguir menor acoplamiento y más flexibilidad). También se envían la instancia del modelo y los flags que indican si los campos de los parámetros B,C y N están habilitados y habilitarlos de manera equivalente en el panel de modificaciones.

## PANEL PREFERENCIAS

## PANEL MODIFICACIONES

Notificación de modificación de gráfica



### ❖ Métodos importantes (PanelModificationController)

- **handleModifyGraphic**: manejador que es llamado cuando se recibe la notificación desde el Panel ‘Preferencias’ para modificar una gráfica de la tabla. En este caso, se recoge de la notificación la instancia del modelo para obtener la grafica que se desea modificar, ya que en el método *showPanel* de la clase *panelController* no solo se manda la notificación que es aquí recibida sino que se guarda en una variable del modelo, la fila o índice del objeto grafica del array que se ha seleccionado para ser modificado, y también se recoge el valor de los flag de los campos B,C y N que indican que campos de la función de la gráfica que se desea modificar están habilitados.

### 3. Bibliografía o referencias

El material que principalmente he utilizado para buscar información sobre las distintas funcionalidades que permite mi aplicación las he encontrado en diversas páginas como repositorios públicos del servicio web GitHub acerca del lenguaje de programación Swift (que luego he intentado en ocasiones transcribir a Objective-C) , distintos libros de programación en macOS como '[Building Cocoa Applications: A Step by Step Guide](#)' , el canal de guías sobre programación en Cocoa: [AppleProgramming](#) , así como distintos foros de ayuda a programadores como StackOverFlow o AppCoda , entre otros , y la página de documentación de Apple <https://developer.apple.com/>.

- NSTableView y sus características principales

<https://www.youtube.com/watch?v=p5U94-uRCOo>

<https://www.raywenderlich.com/830-macos-nstableview-tutorial>

<https://christiantietze.de/posts/2016/04/edit-nstableview-header-cell/>

- Documentación del código y formato de comentarios Objective-C

<https://www.appcoda.com/documenting-source-code-in-xcode/>

- Guardar información de la aplicación y exportarla a un fichero

<https://www.youtube.com/watch?v=yfiOGQmUVjE&list=PLE83F832121568D36&index=22>

<https://stackoverflow.com/questions/31852949/how-to-write-json-file-in-ios/31853362>

<https://stackoverflow.com/questions/17779655/objective-c-create-text-file-to-read-and-write-line-by-line-in-cocoa/17779710>

<https://stackoverflow.com/questions/1286212/how-to-convert-nsarray-to-nsdata>

<https://stackoverflow.com/questions/1828665/convert-nsarray-to-nsstring-in-objective-c>

<https://github.com/davedelong/CHCSVParser/blob/master/CHCSVParser/CHCSVParser/CHCSVParser.m>

<https://developer.apple.com/documentation/foundation/nsscanner/1412801-atend?language=objc>

- Importar información de un fichero

<https://stackoverflow.com/questions/6108924/how-can-i-convert-a-nsdata-to-nsarray>

- Uso de los paneles 'Open and Save'. Paneles de guardado y de apertura de archivos de macOS

<https://developer.apple.com/library/archive/documentation/FileManagement/Conceptual/FileSystemProgrammingGuide/UsingtheOpenandSavePanels/UsingtheOpenandsavePanels.html>

<http://www.extelligentcocoa.org/nsopenpanel-nssavepanel/>

<https://stackoverflow.com/questions/6673386/how-can-i-preset-the-filename-in-nssavepanel>

[http://juliuspaintings.co.uk/cgi-bin/paint\\_css/animatedPaint/009-NSSavePanel.pl](http://juliuspaintings.co.uk/cgi-bin/paint_css/animatedPaint/009-NSSavePanel.pl)

<https://stackoverflow.com/questions/47902995/read-and-write-permission-for-user-selected-folder-in-mac-os-app>  
<https://stackoverflow.com/questions/10708845/nssavepanel-is-not-saving-a-file-after-sandboxing-an-app>

- Eventos de ratón

[https://www.youtube.com/watch?v=Q-2W5o\\_znLU&list=PLE83F832121568D36&index=27](https://www.youtube.com/watch?v=Q-2W5o_znLU&list=PLE83F832121568D36&index=27)  
<https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/EventOverview/HandlingMouseEvents/HandlingMouseEvents.html>  
<https://developer.apple.com/documentation/appkit/nsevent/1529068-locationinwindow>

- Múltiples ventanas

<https://www.youtube.com/watch?v=Z1Erw7aP0EQ&index=18&list=PLE83F832121568D36>

- NSView y sus principales características

<https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/CocoaViewsGuide/WorkingWithAViewHierarchy/WorkingWithAViewHierarchy.html>  
<https://stackoverflow.com/questions/4723199/overlay-nsview-over-nsscrolleview/4764367>

- Representación y dibujado

[https://developer.apple.com/library/archive/documentation/AudioVideo/Conceptual/HTML-canvas-guide/CreatingChartsandGraphs/CreatingChartsandGraphs.html#/apple\\_ref/doc/uid/TP40010542-CH11-SW10](https://developer.apple.com/library/archive/documentation/AudioVideo/Conceptual/HTML-canvas-guide/CreatingChartsandGraphs/CreatingChartsandGraphs.html#/apple_ref/doc/uid/TP40010542-CH11-SW10)  
<https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/CocoaDrawingGuide/DrawingEnviron/DrawingEnviron.html>

- Paneles 'PopUp' y 'PopOver'

<https://stackoverflow.com/questions/18417432/how-to-show-alert-pop-up-in-in-cocoa>  
<https://www.youtube.com/watch?v=P6HX0K7ulSY&t=0s&list=WL&index=56>

- Guardar el contenido de un CustomView en una imagen

<https://stackoverflow.com/questions/32619372/how-to-render-a-layer-hosting-nsview-to-file>  
<https://discussions.apple.com/thread/2749575>  
<https://stackoverflow.com/questions/5491403/saving-an-nsview-to-a-png-file>  
<https://stackoverflow.com/questions/3251261/how-do-i-take-a-screenshot-of-an-nsview>  
<https://gist.github.com/shpakovski/7696268>  
<https://stackoverflow.com/questions/3038820/how-to-save-a-nsimage-as-a-new-file>

- NSNumberFormatter y restricción de los NSTextField a solo números

<https://forums.macrumors.com/threads/custom-nsnumberformatter-restricting-input-to-numeric-with-decimal.1083359/>

<https://stackoverflow.com/questions/12161654/restrict-nstextfield-to-only-allow-numbers>

<https://stackoverflow.com/questions/6337464/nsnumberformatter-doesnt-allow-typing-decimal-numbers>

- Zoom

<https://stackoverflow.com/questions/2444587/zooming-in-an-nsview>

<https://github.com/Darkwonder/BoundsAndFramesCroppingAndScalling/blob/master/LearningBoundsFrameCropping/ViewController.swift>

<https://stackoverflow.com/questions/1210047/cocoa-whats-the-difference-between-the-frame-and-the-bounds>

<https://eternalstorms.wordpress.com/2015/04/29/opensource-zoom-transition-between-nsviews/>

<http://downloads.goldensoftware.com/guides/Grapher13UsersGuidePreview.pdf>

<https://github.com/glfw glfw/issues/90>

<https://stackoverflow.com/questions/13990102/zoom-feature-like-preview-mac-app-for-nsview>

<https://stackoverflow.com/questions/6663028/how-to-perform-zoom-in-effect-on-nsview>

<http://blog.interfacevision.com/design/example-zoom-in-out/>

- Uso de NSIndexSet para representación de varías gráficas en la misma vista

<https://stackoverflow.com/questions/3773180/how-to-get-indexes-from-nsindexset-into-an-nsarray-in-cocoa>