# Borrador de turistas:

## Eliminación automática de objetos en movimiento de fotografias en Microsoft Windows

Daniel Guijarro-Sánchez

**Resumen**— Este proyecto se ha llevado a cabo para desarrollar una herramienta simple y sencilla pero rápida y eficaz que permita eliminar elementos no deseados de una serie de imágenes. La motivación inicial del proyecto ha sido el poder borrar turistas de imágenes tomadas en zonas turísticas debido a que el principal problema de este tipo de fotografías es la constante presencia de extraños en ellas. Se ha utilizado una metodología ágil y dinámica para obtener los mejores resultados, la cual se basa en la simplicidad, la comunicación y el *feedback*. El desarrollo de la aplicación se ha llevado a cabo en el lenguaje de programación C++, utilizando la biblioteca Microsoft Foundation Class para la interfaz, y las bibliotecas GIPL y GDI+ para el algoritmo central del programa. El trabajo ha transcurrido sin incidencias y sin apenas cambios desde la planificación inicial. En consecuencia, los resultados han sido satisfactorios.

**Palabras clave**— Algoritmo, Aplicación, Borrar, Fotografía, Imagen, Mediana, Turista

**Abstract**— This project has been carried out to develop a simple and easy but fast and effective tool to remove unwanted elements in a set of pictures. The motivation behind this project has been to be able to erase tourists from photographs taken in tourist destinations because the main problem with this kind of photography is the constant presence of strangers in them. It has been used an agile and dynamic methodology for best results, which is based on simplicity, communication and feedback. The development of the application is implemented in the C++ programming language, using the Microsoft Foundation Class library for the interface, and the GDI+ and GIPL libraries for the core algorithm of the program. The work has passed without incident and with little change from the initial planning. Consequently, the results have been satisfactory.

**Index Terms**— Algorithm, Application, Erase, Image, Median, Photography, Picture, Tourist

## 1 Introducción

El objetivo del proyecto es permitir a un usuario extraer el fondo o las partes estáticas de una serie de imágenes tomadas con trípode que contienen elementos no deseados y que ocultan parte del objetivo real de la fotografía. Utilizando un algoritmo basado en la mediana estadística, el programa extrae el fondo de las imágenes que el fotógrafo había pretendido captar.

Cuando alguien visita una ciudad desea hacer fotos de sus monumentos, pero casi siempre hay turistas delante que impiden sacar una buena foto. Una solución para obtener una foto sin turistas es hacer varias fotos con el mismo encuadre esperando que en cada foto haya una parte oculta del monumento diferente.

Por ejemplo, en una foto se encuentra un turista a la derecha que luego camina y sale en la izquierda de la siguiente foto. Entonces podemos juntar la parte izquierda de la primera foto con la parte derecha de la segunda foto para conseguir una foto completa del monumento sin turista.

El objetivo del proyecto sería hacer un programa que automáticamente seleccione las partes de las fotos donde no hay turistas y así pueda formar una foto completa sin ellos.

Ilustración 1: Ejemplo del problema de los turistas, en este caso en el arco del triunfo en Barcelona.

La motivación que ha llevado a realizar este proyecto ha sido poder trabajar en la visión por computador. La cámara capta todo lo que aparece en la escena, pero utilizando este método podemos obtener lo que el fotógrafo realmente visualizaba.

Actualmente existen algunas herramientas que permiten obtener estos resultados, pero son herramientas mucho más complejas y con toda una serie de características que puede que no sean de necesarias. Existen editores gráficos tales como Adobe Photoshop o GIMP, o herramientas web como Google Plus. Lo que se pretende conseguir con este proyecto es crear una herramienta simple y que permita de manera sencilla obtener dicho resultado.

En el presente artículo se describirá la metodología utilizada para llevar a cabo el proyecto, una descripción del proceso de desarrollo y finalmente los resultados obtenidos. Se incluirá un apéndice con una explicación de la interfaz de usuario e información técnica detallada.

## 2 METODOLOGÍA

La metodología aplicada durante el desarrollo del proyecto pretendía ser dinámica y adaptativa. Por ello contiene propiedades y características de las conocidas metodologías ágiles y más concretamente de la programación extrema (eXtreme Programming (XP)).

### 2.1 Características

Las carácteristicas principales de la metodología que se han adaptado de la programación extrema son las siguientes:

- **Desarrollo iterativo e incremental**: pequeñas mejoras, unas tras otras. E.g. Primero se desarrolló una interfaz básica con tan solo dos botones y después se fueron añadiendo funcionalidades y más botones.
- **Pruebas unitarias continuas**: se repiten frecuentemente para verificar los cambios y se realizan también pruebas de regresión. E.g. Después de construir cada uno de los módulos se testea éste y los que ya funcionaban anteriormente.
- **Integración del equipo de desarrollo con el cliente**: se ha simulado realizando reuniones semanales con el tutor del proyecto.
- **Corrección de errores** antes de añadir nuevas funcionalidades. Se realizan entregas periódicas, en las reuniones semanales.
- **Refactorización del código**: se reescriben partes del código sin cambiar su comportamiento pero mejorando la legibilidad y el rendimiento. Esto se ha aplicado sobretodo en el núcleo del programa, en el algoritmo de mediana, se ha logrado mejorar el rendimiento.
- **Simplicidad en el código:** es preferible realizar algo simple para poder mejorarlo posteriormente. E.g. La interfaz de la aplicación incluía lo básico al inicio del desarrollo para poder construir el núcleo y después se añadieron funcionalidades. Lo mismo para dicho núcleo, se diseñó de la forma más simple posible para después mejorarlo si era posible.

### 2.2 Roles

Al ser este un proyecto individual, la mayoría de roles han recaído sobre una misma persona. Los roles interpretados han sido básicamente los de:

- **Programador:** construye el código del programa y realiza las pruebas unitarias
- **Tester:** escribe las pruebas a realizar e informa de los resultados en las reuniones semanales.
- **Tracker:** encargado del seguimiento y del feedback, de nuevo en las reuniones semanales.
- **Gestor:** coordinador. Nexo entre cliente (tutor) y programador.

Mientras que el tutor ha colaborado realizando labores equivalentes a las de **Cliente** y **Consultor.**

### 2.3 Valores

Se mantienen los valores originales de la programación extrema:

- **Simplicidad:** es la base de esta metodología. Se simplifica el diseño para agilizar el desarrollo. Se necesita la refactorización de código para mantener esta simplicidad y es importante mantener el código autodocumentado para poder simplificar también la documentación necesaria. Para esto, en el código se han utilizado nombres autodescriptivos para las variables y métodos, insertado los comentarios necesarios y programado de forma ordenada, clara y coherente.
- **Comunicación:** la comunicación con el cliente ha de ser fluída para que éste pueda decidir sobre las prioridades y para solucionar dudas. Esto se ha podido llevar a cabo en las reuniones semanales con el tutor.


- *E-mail de contacto: daniel.guijarros@e-campus.uab.cat*
- *Mención realizada: Ingeniería de Computación.*
- *Trabajo tutorizado por: Francisco Javier Sánchez Pujadas(Centro de Visión por Computador)*
- *Curso 2013/14*

- **Feedback**: al contar con reuniones constantes con el tutor se tiene la semejanza de contar con el cliente dentro del proyecto. Al realizarse ciclos de desarrollo cortos el trabajo es más efectivo teniendo buen nivel de feedback.
- **Coraje:** reconstruir el código o desechar partes obsoletas requiere valentía. Además coraje significa persistencia: un problema complejo puede impedir el avance durante horas hasta que se consiga resolver. E.g. el módulo de lista de *thumbnails* requirió más horas de las previstas pero se logró llevar a cabo gracias a este empeño.

### 2.4 Planificación

Al comienzo del proyecto se realizó una planificación desglosando las tareas e intentando prever el trabajo que requeriría cada una.

**TABLA 1**

**PLANIFICACIÓN INICIAL**

| Tarea | Esfuerzo (horas) | Previsión |
|---|---|---|
| Reuniones de seguimiento | 12 | Semanalmente |
| Interfaz MFC | 80 | Semanas 2 – 14 |
| Cargar imágenes | 8 | Semana 2 |
| Seleccionar imágenes | 12 | Semana 3 |
| Visualización imagen | 12 | Semana 4 |
| Creación máscara | 24 | Semana 14 |
| Módulo de ejecución | 12 | Semana 5 |
| Exportar imagen resultado | 12 | Semana 10 |
| Núcleo programa | 100 | Semanas 5 – 16 |
| Algoritmo de mediana | 60 | Semanas 5 – 16 |
| Auto-alineado de imágenes | 40 | Semanas 14 – 16 |

Respecto a esta planificación inicial se realizaron algunos cambios de prioridades a lo largo del proyecto y algunas tareas tuvieron que descartarse.
Respecto a la interfaz, el módulo de selección de imágenes (listado de *thumbnails*) se amplió y mejoró, por lo que tuvo que despriorizarse el módulo de creación de máscara y finalmente no se desarrolló.
Y en lo que al núcleo del programa concierne, se llevaron a cabo mejoras de rendimiento, uso de memoria y varias refactorizaciones del algoritmo de mediana, por lo que el auto-alineado de imágenes no se llevó a cabo.

## 3 DESARROLLO

El desarrollo del proyecto se ha llevado a cabo lo más acorde posible a la planificación inicial, y gracias a la metodología de trabajo utilizada se han podido gestionar e implementar los pocos cambios surgidos tanto en funcionalidades como en prioridades.

El programa se ha desarrollado en el lenguaje de programación C++, debido a su alto rendimiento frente a otras opciones como MATLAB, C# o Python.

### 3.1 Interfaz

Para el desarrollo de la interfaz de usuario y la comunicación de ésta con la parte interna del programa se ha utilizado la biblioteca Microsoft Foundation Class, la cual encapsula porciones de la API de Windows en clases de C++, y nos permite crear aplicaciones de escritorio para Windows. Para el módulo de visualización de imágenes se ha utilizado CVGraphBook, una clase que hereda de la clase CStatic de MFC y creada por el tutor Francisco Javier Sánchez Pujadas.

Ilustración 2: Interfaz de la aplicación recién abierta (arriba) y con imágenes cargadas (abajo).

La interfaz de usuario fue diseñada y desarrollada en primer lugar, y apenas sufrió cambios respecto al diseño inicial. Una descripción más detallada del diseño y funcionamiento de la interfaz de usuario aparece en el apéndice.

### 3.2 Núcleo del programa

Para las operaciones de manipulación y procesamiento de imágenes se han utilizado conjuntamente la API GDI+, de Microsoft Windows, y la biblioteca GIPL, desarrollada por el tutor Francisco Javier Sánchez Pujadas. GDI+ se ha utilizado para ejecutar el algoritmo central del programa, mientras que GIPL ha servido para las operaciones de entrada y salida de las imágenes, la carga y el guardado de éstas en los distintos formatos aceptados en la aplicación. Estos formatos son BMP, JPEG, PNG y TIFF (8 y 16 bits). El diseño inicial del algoritmo de mediana (núcleo del programa) se pensó de la manera más simple posible: se iteraría sobre cada uno de los píxeles de cada imagen y sobre todas las imágenes, calculando la mediana entre todos los píxeles equivalentes de cada imagen.

Después de este diseño inicial se realizó un prototipo en MATLAB, debido a la sencillez y rapidez de desarrollo de este lenguaje. Posteriormente se implementó el algoritmo en C++ utilizando GetPixel() y SetPixel() de GDI+. El programa funcionaba correctamente pero siendo esta la funcionalidad principal se optó por realizar mejoras en el algoritmo para optimizar el rendimiento [1]. Se utilizó la otra alternativa de la que dispone GDI+, LockBits(), más compleja pero con mejor rendimiento. Esta función nos permite trabajar directamente en memoria con punteros.
En el proceso de cálculo de medianas se ha utilizado el algoritmo QuickSelect [2], ya que era el idóneo para optimizar el rendimiento del programa, debido a que es algoritmo de ordenación con mejor velocidad de ejecución.

A lo largo del desarrollo se han realizado pequeños cambios en la planificación y han surgido algunas leves dificultades:

- Falta de conocimiento en el desarrollo con MFC. Se han consultado tutoriales online, libros y ejemplos [referencias] para aprender y mejorar el código.
- Biblioteca GIPL. Al comienzo del proyecto no se disponía de dicha biblioteca de modo que se optó por realizar un primer prototipo del núcleo del programa usando MATLAB. Una vez se disponía de ella se encontraron algunas dificultades para utilizarla en el programa y se optó por utilizar GDI+ en algunas de las partes.
- Dificultad en la optimización del núcleo del programa. El uso de GetPixel() y SetPixel() de GDI+ era sencillo, y el cambio a utilizar punteros a memoria supuso un número mayor de horas de trabajo de las previstas. Se despriorizó la tarea del autoalineado de imágenes para dar cabida a estas mejoras.

## 4 RESULTADOS

Los resultados del proyecto han sido, a grandes rasgos, muy acordes a la planificación.
La interfaz de usuario es esencialmente tal y como se diseñó inicialmente. Las única diferencia es que se han añadido algunas características para mejorar la experiencia del usuario. Los módulos de la interfaz que se han mantenido intactos son los de visualizar imagen, lista de *thumbnails*, carga de imágenes y guardado de resultado. Se ha añadido una barra de progreso para permitir al usuario conocer en tiempo real cómo va la ejecución, también la característica de visualizar el último resultado obtenido sin tener que volver a ejecutar el programa de nuevo y se ha descartado el módulo de máscara, que permitiría mantener una zona de una imagen en el resultado final.

También podría decirse, de manera objetiva, que la interfaz es usable y sencilla, ya que se han realizado pruebas con usuarios de diversos perfiles y han sido capaces de utilizar la aplicación tras una simple explicación del funcionamiento de ésta.

En cuanto a la parte interna del programa, los cambios realizados han sido para mejorar y optimizar el código y el rendimiento. Se ha obtenido una diferencia muy notable entre los tiempos de ejecución tratando las imágenes píxel a píxel respecto a usando punteros a memoria.

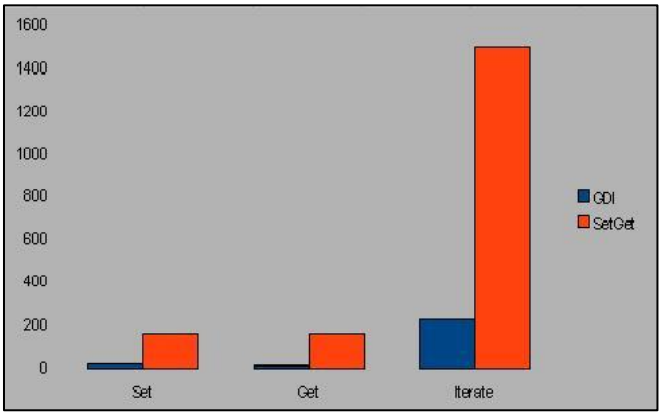



Ilustración 3: Comparación entre GDI+ Lockbits y GetPixel SetPixel **[1]**

Podemos comprobar que el tiempo de ejecución ha mejorado cerca aproximadamente un 75%. Posteriormente a la mejora del rendimiento también se refactorizó el código para reducir el uso y los accesos a memoria.

En general, los resultados han sido satisfactorios, la aplicación funciona correctamente y el rendimiento es apropiado.

### 4.1 Ejemplos

En esta sección se mostrarán algunos de los resultados del programa. Todas las imágenes han sido tomadas con una cámara réflex digital montada sobre un trípode.

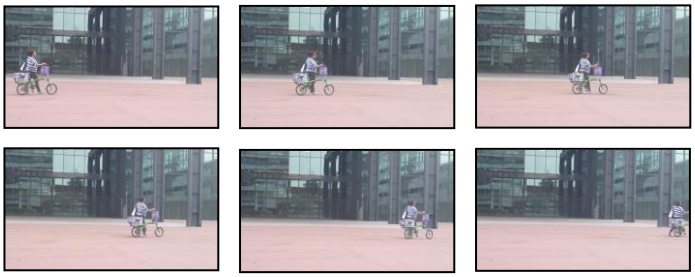


Ilustración 4: Fira de Sabadell. Un único objeto en movimiento borrado satisfactoriamente.

Ilustración 5: Passeig de Lluís Companys. Varios turistas atraviesan la escena. En este ejemplo puede comprobarse como el movimiento de las palmeras puede causar pérdida de nitidez en el resultado.

Ilustración 6: Las columnas de la UAB. Podemos apreciar como las nubes se mantienen debido a que apenas se desplazan durante la toma pero desaparecen los vehículos de la autopista así como las personas que pasean por la hierba.

## 5 CONCLUSIÓN

Ha resultado un proyecto exitoso ya que se han cumplido los objetivos principales. Además ha sido un trabajo interesante y fructífero con el que se han podido pronfundizar los conocimientos en MFC, C++ y en tratamiento de imágenes en general.

Cada una de las partes del proyecto es actractiva en sí misma y eso permite tener motivación durante el trabajo.

Otro punto importante a destacar es la realización del proyecto en sí, crear una aplicación completa jugando los papeles de los distintos roles que hay en un equipo de desarrollo en el mundo profesional.

### 5.1 Trabajos futuros

Como posible trabajo a llevar a cabo en el futuro, algunas de las funcionalidades que se planificaron como opcionales y que han tenido que descartarse por cambios de prioridades podrían implementarse como mejoras de la aplicación. Éstas serían el módulo de máscara y el autoalineado de imágenes.

También podrían realizarse mejoras en otros aspectos de la aplicación como optimizar aún más el núcleo del programa para mejorar el rendimiento o cambiar la interfaz de usuario adaptando la aplicación a una interfaz más moderna con estilo Windows 8.

## AGRADECIMIENTOS

Debe agradecerse la ayuda prestada al tutor Francisco Javier Sánchez Pujadas, tanto por los conocimientos enseñados, la contribución con clases MFC de ejemplo, como CVGraphBook, y la biblioteca GIPL. También a Alba Guijarro Sánchez por su ayuda en la toma de fotografías para pruebas y demostración de resultados.

## BIBLIOGRAFÍA

[1] M. Franc, "LockBits vs Get Pixel Set Pixel - Performance," 22 November 2009. [Online]. Available: http://www.mfranc.com/programming/operacje-na-bitmapkach-net-1/. [Accessed 12 2013].

[2] SourceTricks, "Programming Tutorials by SourceTricks," 22 June 2011. [Online]. Available: http://www.sourcetricks.com/2011/06/quick-select.html. [Accessed December 2013].

[3] Y. Ren, "Using thumbnail images in a list control," 21 July 2006. [Online]. Available: http://www.codeproject.com/Articles/969/Using-thumbnail-images-in-a-list-control. [Accessed 12 2013].

[4] cgermany77, "MFC Interfaces - Part 1 of 2 - Creating Simple Hand-crafted (hacked) MFC Interfaces for C++ Programs," January 2012. [Online]. Available: https://www.youtube.com/watch?v=Su1x9EA5Njk. [Accessed 10 2013].

[5] The MathWorks, "Matlab Documentation Center," [Online]. Available: http://www.mathworks.es/es/help/matlab/. [Accessed 26 11 2013].

[6] moah, "Thumbnails Viewer using ListCtrl," 27 January 2006. [Online]. Available: http://www.codeproject.com/Articles/4990/Thumbnails-Viewer-using-ListCtrl. [Accessed November 2013].

[7] C. Germany, «Visual Studio MFC 2008 Bare Bones CDialog GUI Template,» 2010. [En línea]. Available: http://networkingprogramming.com/1024x768/2008_Bare_Template.html. [Último acceso: October 2013].

[8] A. Peters, "MFC Step-by-Step Guide, Event Programming Tutorial," 2006. [Online]. Available: https://depts.washington.edu/cmmr/biga/chapter_tutorials/1.C++_MFC_D3DOGL/1.StepByStepGuide/index.html. [Accessed November 2013].

[9] M. Weagle, "Using the List Control," 15 November 2002. [Online]. Available: http://www.codeproject.com/Articles/608/Using-the-List-Control. [Accessed November 2013].

[10] J. Wood, "Quickly Create Simple Applications Using MFC," 2010. [Online]. Available: http://msdn.microsoft.com/en-us/vstudio/cc507088.aspx. [Accessed November 2013].

[11] R. W. Powell, "Using the LockBits method to access image data," 2003. [Online]. Available: http://bobpowell.net/lockingbits.aspx. [Accessed December 2013].

[12] "Using LockBits in GDI+," 12 December 2009. [Online]. Available: http://supercomputingblog.com/graphics/using-lockbits-in-gdi/. [Accessed December 2013].

[13] D. Wells, "Extreme Programming: A gentle introduction," 8 October 2013. [Online]. Available: http://www.extremeprogramming.org/. [Accessed October 2013].

[14] Chuidiang, "Programación extrema," 4 February 2007. [Online]. Available: http://www.chuidiang.com/ood/metodologia/extrema.php. [Accessed October 2013].

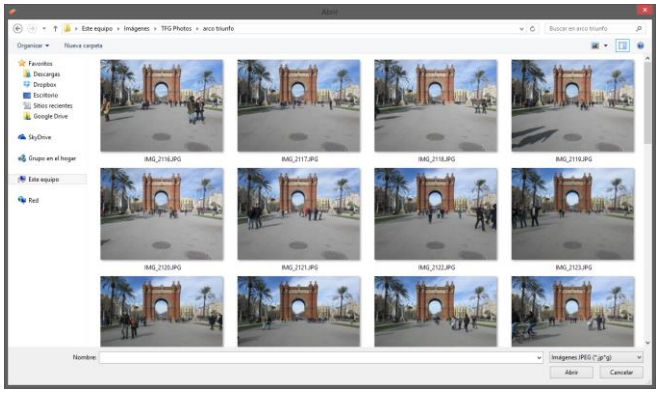
# APÉNDICE

## A1. INTERFAZ DE USUARIO

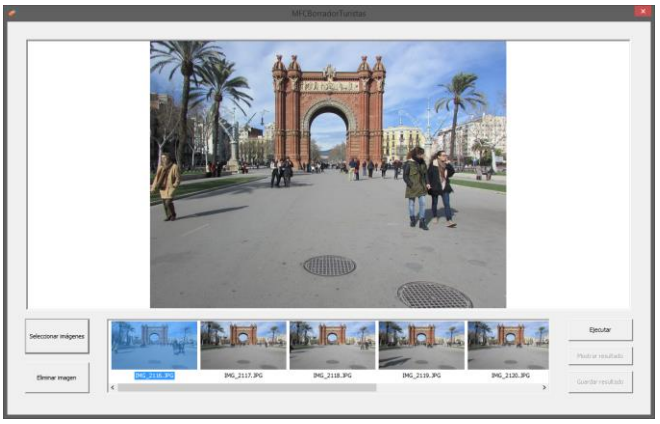
En el presente apartado se explicará el funcionamiento de la interfaz de usuario analizando las casuísticas más comunes.

1. **Interfaz principal**. Consta del módulo de visualización de imagen, la lista de *thumbnails* debajo, y los botones de comandos.

2. **Seleccionar imágenes**. Al pulsar dicho botón, se abre un diálogo de carga de ficheros que permite seleccionar las imágenes deseadas siempre y cuando sean de los formatos bmp, jpeg, png o tiff.

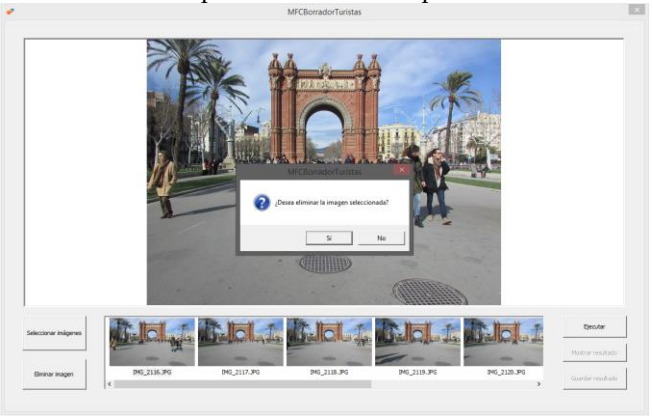


Al seleccionar las imágenes deseadas éstas aparecerán en la lista de *thumbnails* y la primera de ellas se mostrará en el visualizador. Además, se habrán habilitado los botones de "Eliminar imagen" y "Ejecutar".

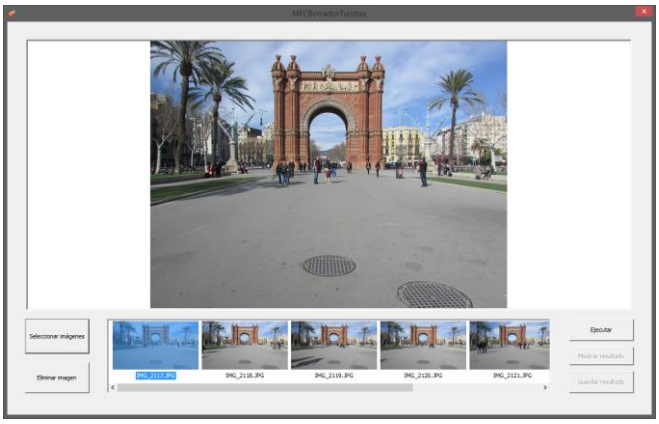


3. **Módulo de visualización de imagen**. El visualizador nos permite mover la imagen, expandirla o reducirla, y volver a dejarla en el tamaño original. Todo esto utilizando el botón central o rueda del ratón.

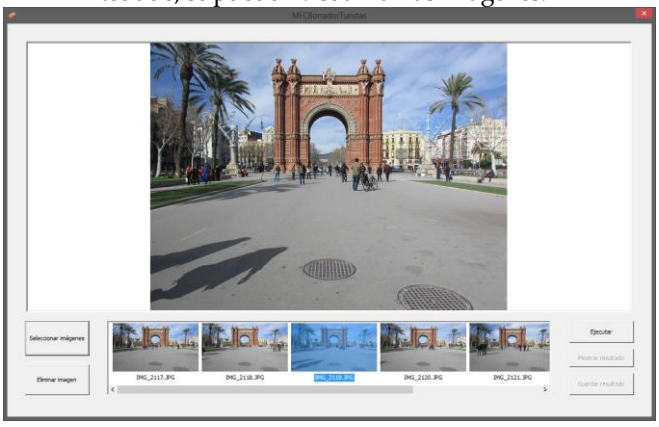
4. **Eliminar imagen**. Pulsando sobre el botón de eliminar imagen se procederá a borrar la imagen seleccionada de la lista. Esta acción también puede realizarse pulsando la tecla "Supr".



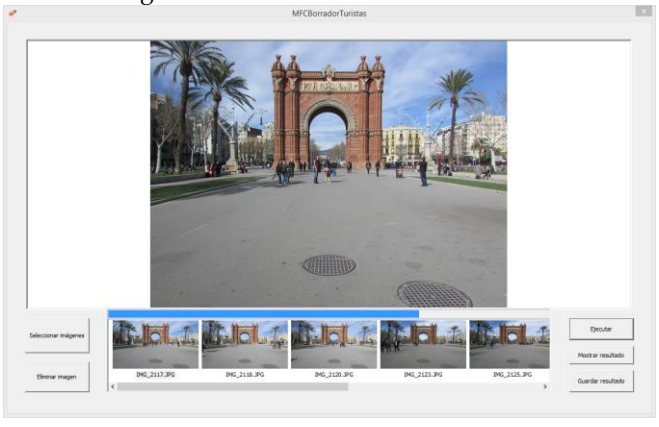
Tras confirmar la eliminación, puede observarse como el primer elemento (el seleccionado en este caso) ha sido borrado.



5. **Visualizar imagen.** Clicando sobre un elemento de la lista, o utilizando las teclas de desplazamiento del teclado, se pueden visualizar las imágenes.

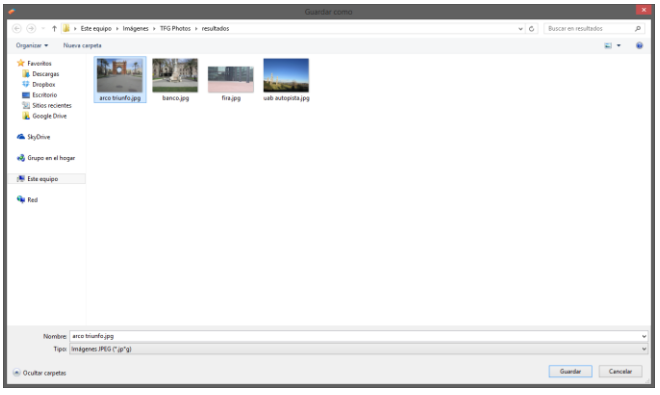


6. **Ejecutar.** Al pulsar sobre este botón se ejecuta el algoritmo de mediana. Durante el proceso aparece una barra de progreso y finalmente se muestra la imagen resultado.



7. **Mostrar resultado.** Este botón sirve para mostrar en el visualizador el último resultado obtenido sin tener que volver a ejecutar el programa de nuevo, evitando la espera del proceso. Es útil cuando se ha clicado en alguna otra imagen y se ha perdido la vista del resultado.

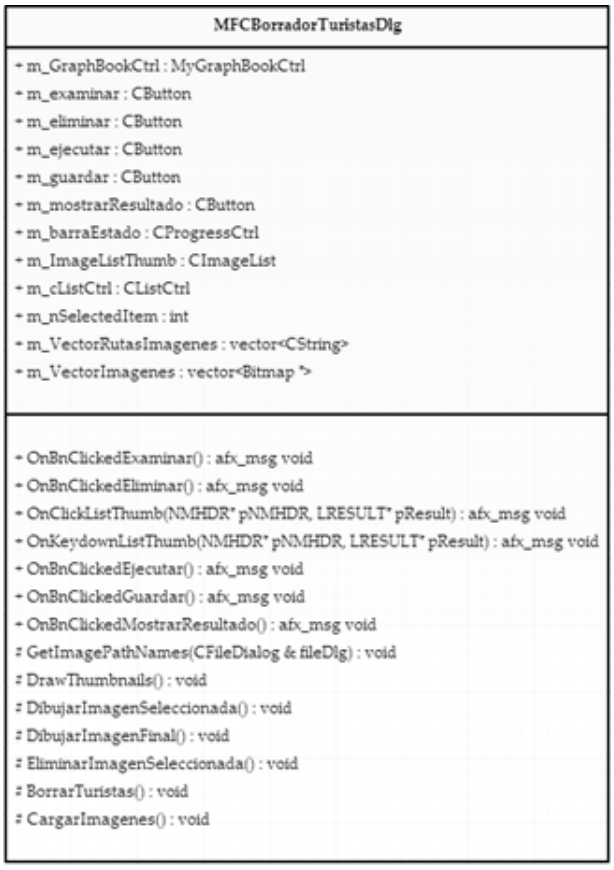
8. **Guardar resultado.** Al pulsar dicho botón, aparece un diálogo de guardado de ficheros que permite almacenar en disco el resultado del programa en los formatos bmp, jpeg, png o tiff.



## A2. DETALLES TÉCNICOS

En este apartado se explicará el funcionamiento del programaroyecto. Para una mayor simplicidad y escalabilidad el código se ha modularizado y se ha dividido en distintos métodos. Existen dos clases en el programa, MFCBorradorTuristas y MFCBorradorTuristasDlg, ambas autogeneradas por Visual Studio al crear el proyecto MFC basado en Diálogos. La que se ha editado para añadir la lógica del programa es MFCBorradorTuristasDlg.

La clase dispone de los siguientes miembros:




- **Variables miembro**. Muchas de las variables miembro de la clase son variables de control de elementos de la interfaz. Éstas sirven para alterar el estado del elemento al que representan, E.g. activar o desactivar un botón, modificar una barra de progreso o añadir elementos a una lista. No funcionan como *listeners*, de eso se encargan algunas funciones miembro que se explicarán más adelante en este apartado.
    - **m_GraphBookCtrl**: variable control del GraphBook de la interfaz. Es el módulo donde se visualizan las imágenes.
    - **m_examinar**: variable control del botón "Seleccionar imágenes" de la interfaz.
    - **m_eliminar**: variable control del botón "Eliminar imagen" de la interfaz.
    - **m_ejecutar:** variable control del botón "Ejecutar" de la interfaz.
    - **m_guardar:** variable control del botón "Guardar Resultado" de la interfaz.
    - **m_mostrarResultado:** variable control del botón "Mostrar resultado" de la interfaz.
    - **m_barraEstado:** variable control de la barra de progreso de la interfaz. Permite controlar dicho progreso a medida que avanza la ejecución del programa para proporcionar información del progreso de la ejecución al usuario.
    - **m_ImageListThumb:** variable que contiene la lista de imágenes que se visualiza en la lista de *thumbnails* de la interfaz. No es el control, solamente contiene las imágenes que se muestran en el control.
    - **m_cListCtrl:** variable control de la lista de *thumbnails* de la interfaz. Permite setear una lista de imágenes a visualizar y controlar los elementos en ella.
    - **m_nSelectedItem:** variable numérica que permite conocer el elemento seleccionado de la lista de *thumbnails* en cada momento. Por defecto es cero, primer elemento.
    - **m_VectorRutasImagenes:** variable de tipo vector (array dinámico) en el que se almacenan las rutas a las imágenes seleccionadas.
    - **m_VectorImagenes:** variable de tipo vector (array dinámico) que contiene las imágenes seleccionadas en formato mapa de bits.

- **Funciones miembro.** Las funciones miembro cuyo nombre es del tipo "On…" y que tienen el *type modifier* "afx_msg" son los *listeners* de eventos de la interfaz. Éstas son las encargadas de llamar al resto de funciones, en las que se han encapsulado unas tareas concretas.
    - **OnBnClickedExaminar:** crea un diálogo de apertura de ficheros, el cuál está limitado a abrir solamente imágenes de tipo bmp, jpeg, png o tiff. Una vez seleccionadas las imágenes, esta función llama a GetImagePathNames, DrawThumbnails y finalmente a DibujarImagenSeleccionada. También habilita los botones de eliminar y ejecutar.
    - **OnBnClickedEliminar:** llama a la función EliminarImagenSeleccionada.
    - **OnClickListThumb:** detecta qué imagen de la lista ha sido clicada y la muestra mediante DibujarImagenSeleccionada.
    - **OnKeyDownListThumb:** detecta qué tecla se ha pulsado mientras teníamos foco en la lista de *thumbnails*. Si ha sido una de las teclas de desplazamiento izquierda o derecha, altera el valor de m_SelectedItem para seleccionar la imagen correspondiente. Si la tecla pulsada ha sido "Supr", se procede a eliminar esta imagen mediante "EliminarImagenSeleccionada".
    - **OnBnClickedEjecutar:** llama a BorrarTuristas.

- **OnBnClickedGuardar:** crea un diálogo de guardado de ficheros y permite guardar la imagen resultado en bmp, jpeg, png o tiff.
- **OnBnClickedMostrarResultado:** llama a DibujarImagenFinal.
- **GetImagePathNames:** utilizando el cuadro de diálogo de apertura de ficheros que crea OnBnClickedExaminar y en el que se han seleccionado las imágenes a tratar, obtiene la ruta de cada una de estas imágenes y las almacena en m_VectorRutasImagenes.
- **DrawThumbnails:** a partir de la información contenida en m_VectorRutasImagenes, esta función abre cada una de estas imágenes las adapta al tamaño del *thumbnail* y las añade a la lista. También añade el nombre de cada una debajo de ellas.
- **DibujarImagenSeleccionada:** accediendo a la posición "m_nSelectedItem" de m_VectorRutasImagenes, obtiene la ruta de la imagen seleccionada, la carga utilizando GIPL y la añade a m_GraphBookCtrl para poder visualizarla.
- **DibujarImagenFinal:** abre la copia de la última imagen resultado obtenida de la ejecución del programa y la carga en m_GraphBookCtrl para poder visualizarla.
- **EliminarImagenSeleccionada:** después de pedir confirmación al usuario, elimina la imagen seleccionada de la lista de *thumbnails*. Para esto eliminará el elemento en la posición m_nSelectedItem de m_VectorRutasImagenes y llamará a DrawThumbnails para recargar la lista de imágenes en la interfaz. Además mostrará la primera imagen de la lista llamando a DibujarImagenSeleccionada después de haber seteado m_nSelectedItem a cero.
- **BorrarTuristas:** núcleo del programa. Ejecuta el algoritmo de mediana sobre las imágenes contenidas en m_VectorImagenes. Se recorren todas las imágenes y en cada una de ellas se llama a LockBits para obtener un puntero a memoria en la posición en la que se encuentra la información de la imagen. Se itera con este puntero para obtener los valores RGB de la imagen y se añade cada uno en un vector (un vector por color). Se calcula la mediana para cada valor de colores de cada píxel de las imágenes, es decir; cada píxel equivalente en cada una de las imágenes. Esto permite obtener un valor mediano de color en cada píxel, descartando así valores anómalos y puntuales que equivalen a objetos en movimiento. Con cada valor mediano de cada píxel se construye la imagen resultado, la cual se almacena temporalmente en memoria para poder acceder a ella o para guardarla posteriormente. Finalmente se muestra el resultado llamando a DibujarImagenFinal.
- **CargarImagenes:** utilizando m_VectorRutasImagenes, se accede a cada una de estas rutas y carga la imagen correspondiente en un mapa de bits que se almacena en m_VectorImagenes.