MiniWin Documentation

Release 0.2.1

Pau Fernández (@pauek)

09/05/2017 a las 21:13

Contents

Última actualización: 09/05/2017 a las 21:13.

MiniWin es una mini-librería (para ser justos, no llega ni a eso) para poder abrir una ventana, pintar en ella, detectar la presión de las teclas y el movimiento y los botones del ratón. Tiene un objetivo *pedagógico*, sirve esencialmente para poder implementar programas muy simples (como pequeños juegos) que dibujen cosas por la pantalla y con los que se pueda interaccionar mínimamente.

Actualmente funciona en Windows y Linux (si álguien se ofrece a subvencionar un MacBook, quizás entonces me plantee hacer la versión Mac OS X...).

CHAPTER 1

Motivación

MiniWin es una utilidad motivada por la realización de vídeos pedagógicos en la web sobre cursos de programación minidosis.org.

CHAPTER 2

Instalación Rápida

Si usas Code::Blocks, descarga uno de los proyectos siguientes:

- Para Windows, descarga HolaMiniWin-windows.
- Para Linux, descarga HolaMiniWin-linux.

Una vez descargado, descomprime el archivo y dentro de la carpeta `HolaMiniWin` verás un fichero con extensión `.cbp` (un proyecto Code::Blocks). Ábrelo y compila directamente. Te aparecerá una ventana que dice "Hola, MiniWin!".

Si no usas Code::Blocks, consulta el tema *Instalación*.

CHAPTER 3

Autores

Pau Fernández (Google+, Twitter)

CHAPTER 4

Contribuciones

Carlos (@mesjetiu en YouTube).

CHAPTER 5

Índice

Instalación

Antes que nada: descarga MiniWin.

Verás que dentro del Zip hay dos ficheros: `miniwin.h` y `miniwin.cpp`. Puedes ignorar todo lo demás. Estos dos ficheros deben formar parte del proyecto que se está desarrollando. En C++, se pueden hacer programas con más de un fichero fuente (si estás empezando esto no es evidente). Entornos de programación como Dev-C++, Code::Blocks o Geany permiten compilar este tipo de proyectos.

Instalación Rápida para Code::Blocks

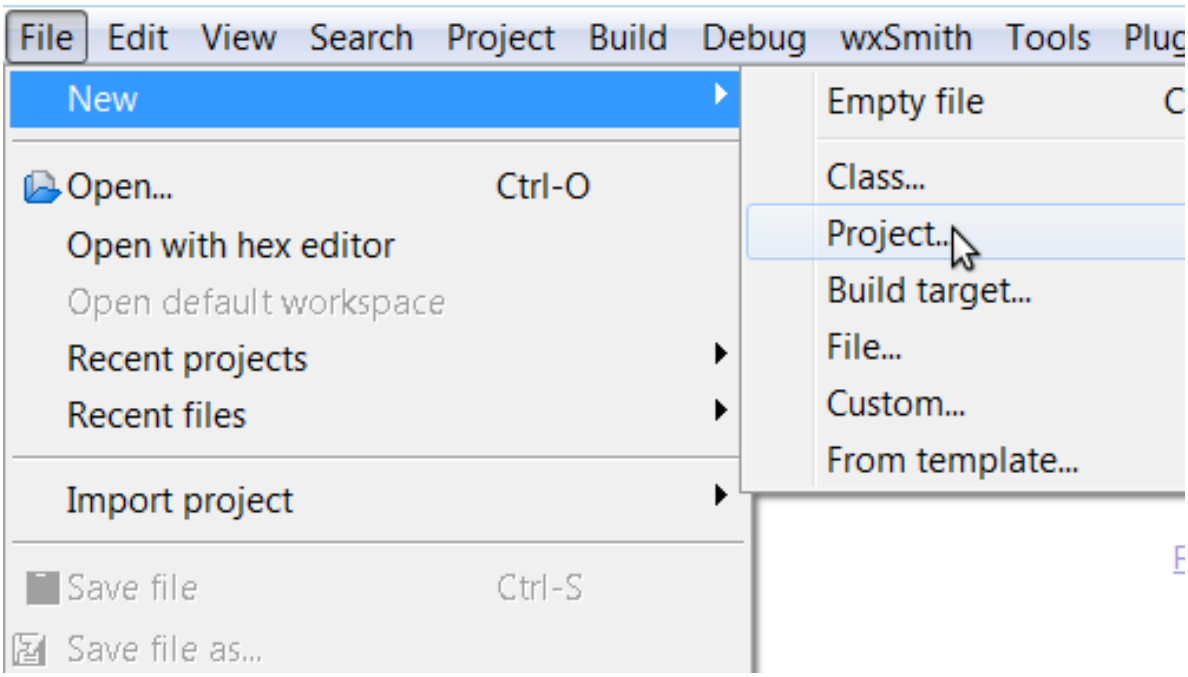
Si usas Code::Blocks en Windows o Linux, descarga el proyecto HolaMiniWin-windows u HolaMiniWin-linux, abre el fichero `HolaMiniWin.cbp` con Code::Blocks y ya puedes empezar trabajar.

Creación de un proyecto en Code::Blocks que use MiniWin

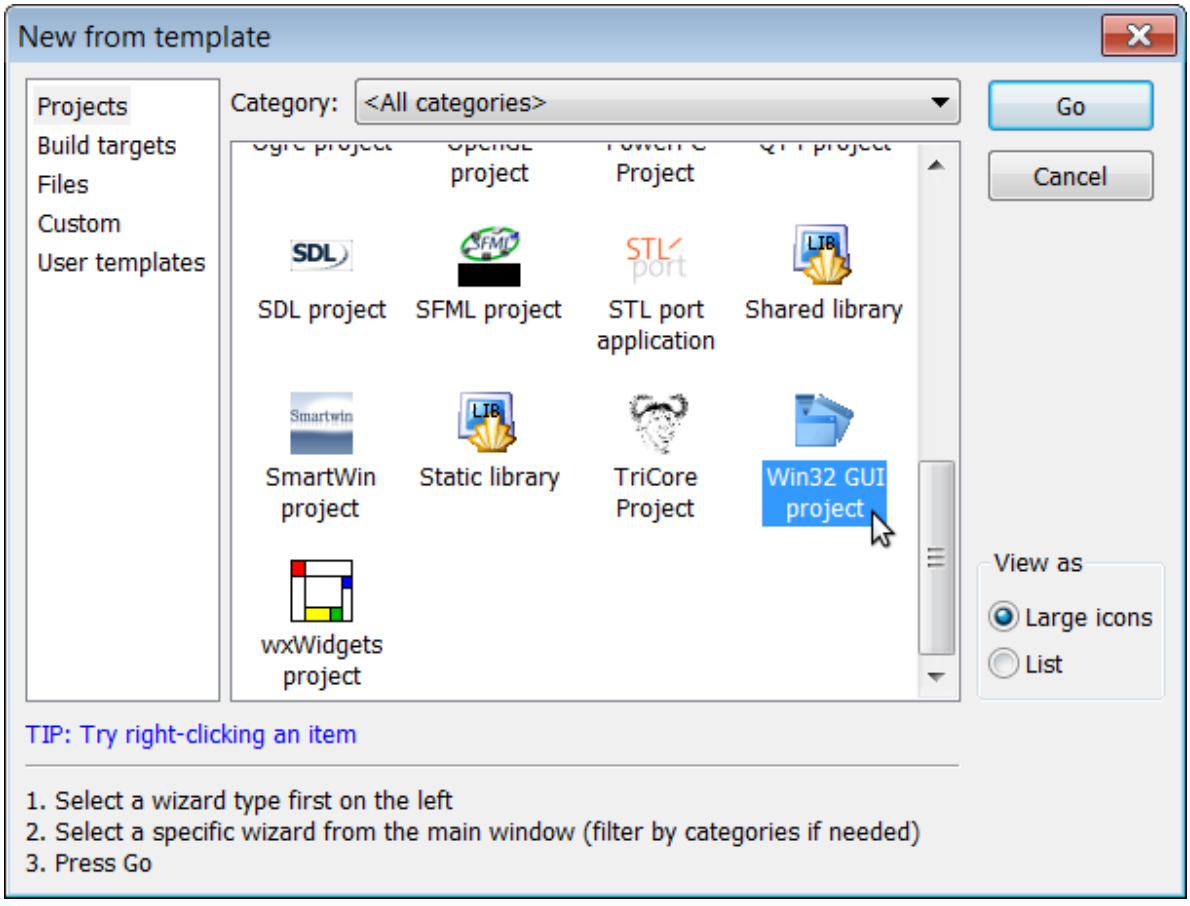
Esta explicación también está en vídeo.

Para usar MiniWin en un proyecto en Code::Blocks sigue los siguientes pasos:

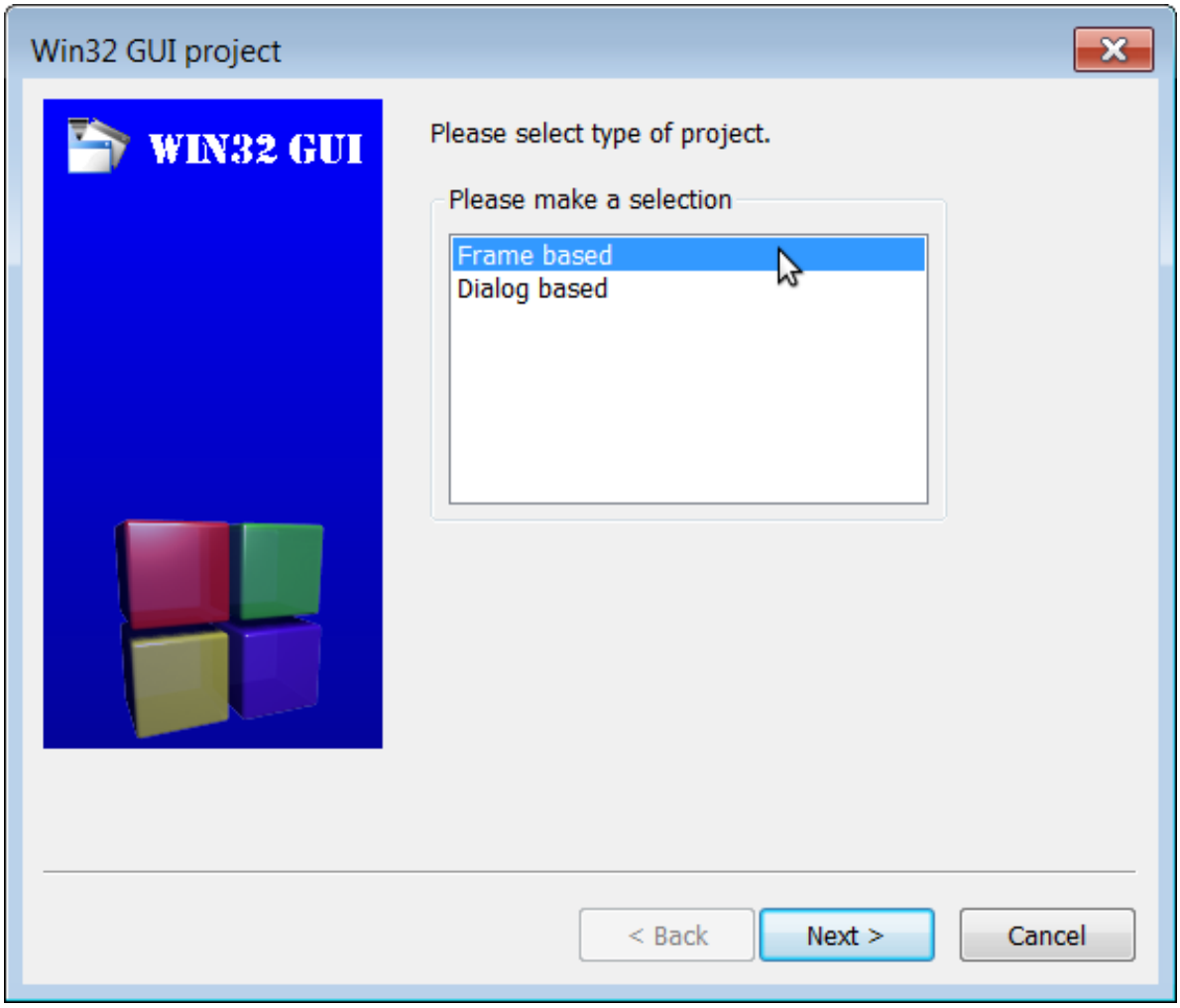
- Crea un nuevo proyecto en Code::Blocks (menú "File - New - Project..."):



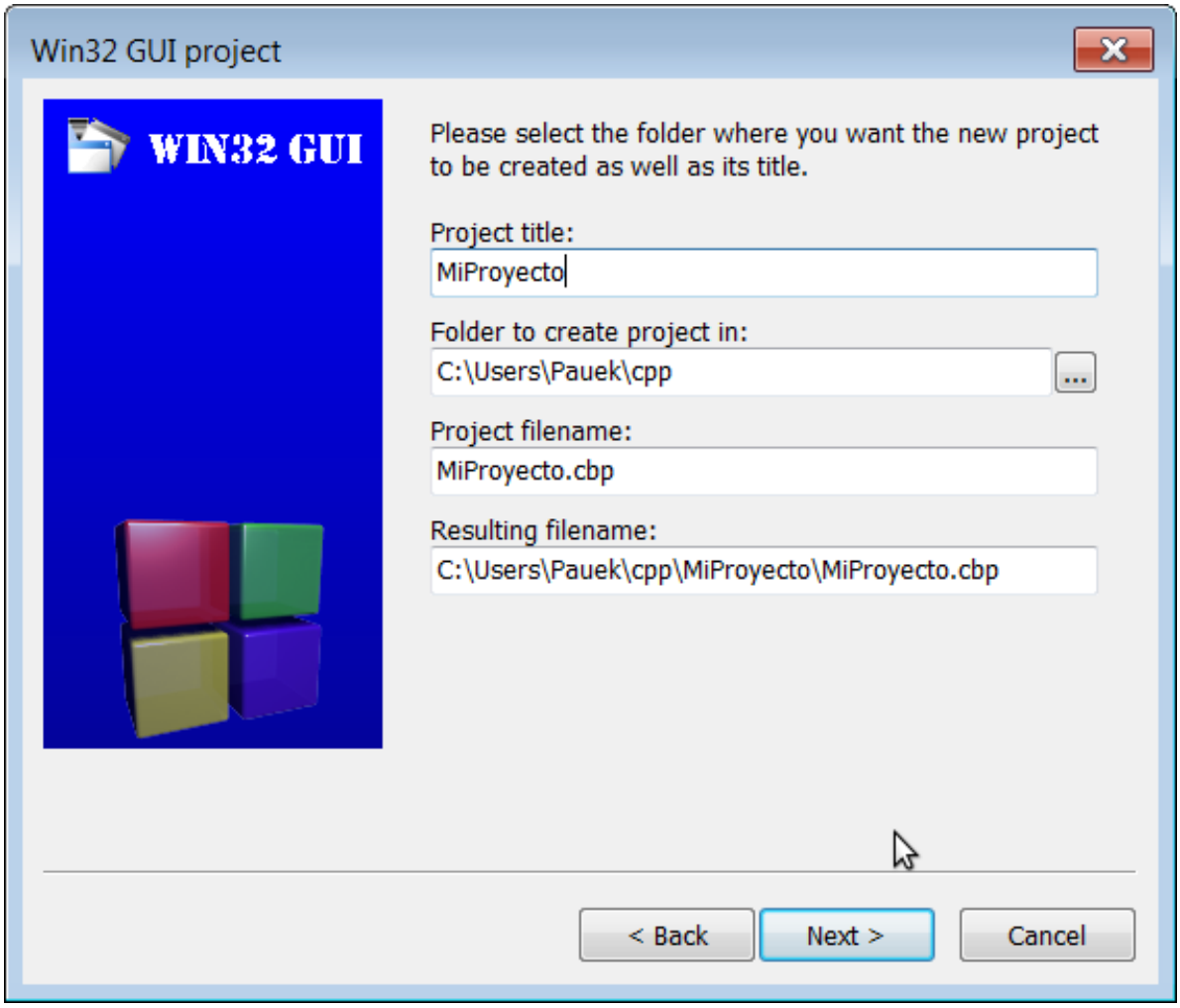
- Escoge el tipo de proyecto como “Win32 GUI”:



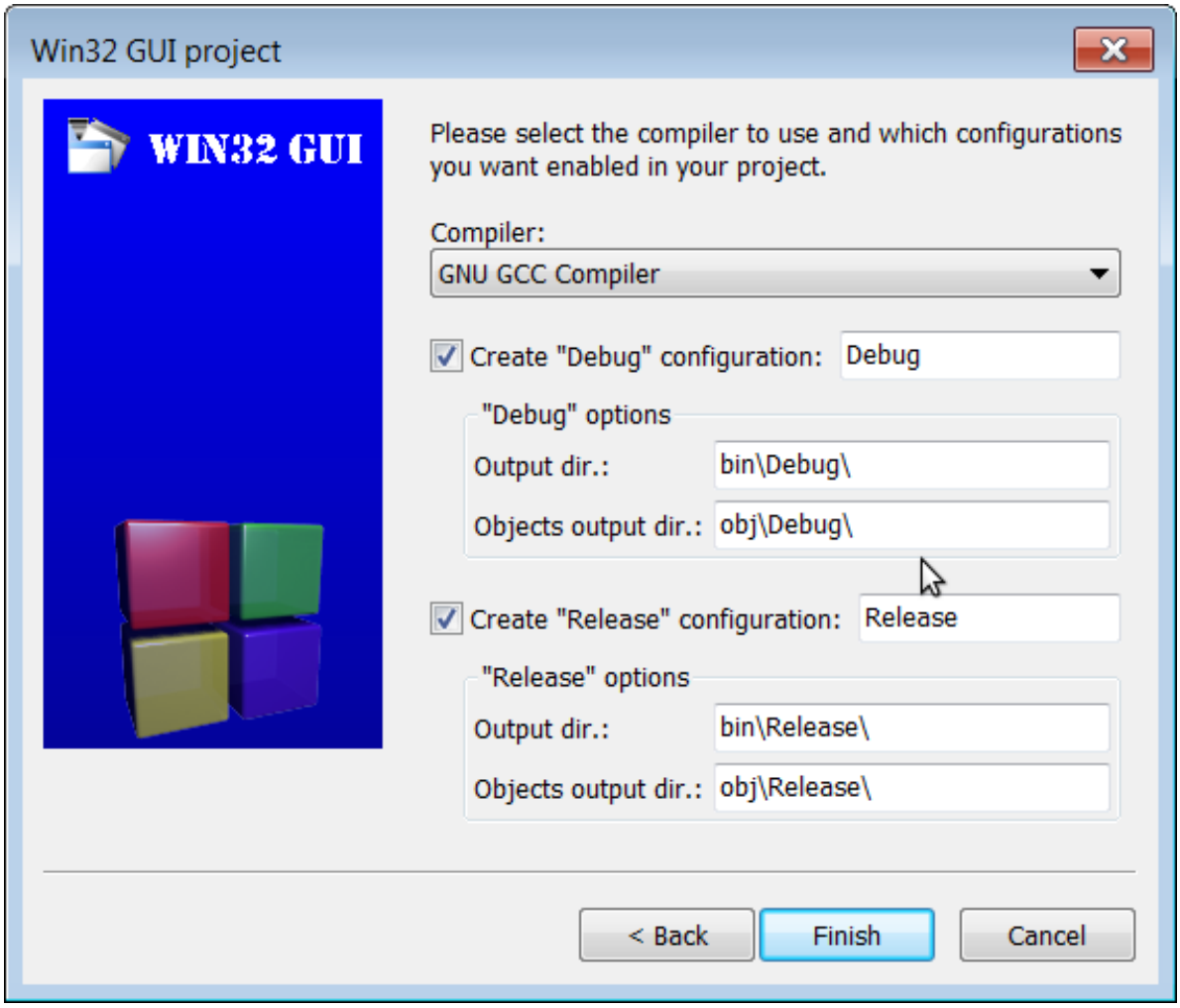
- Escoge “Frame based” en el tipo de proyecto:



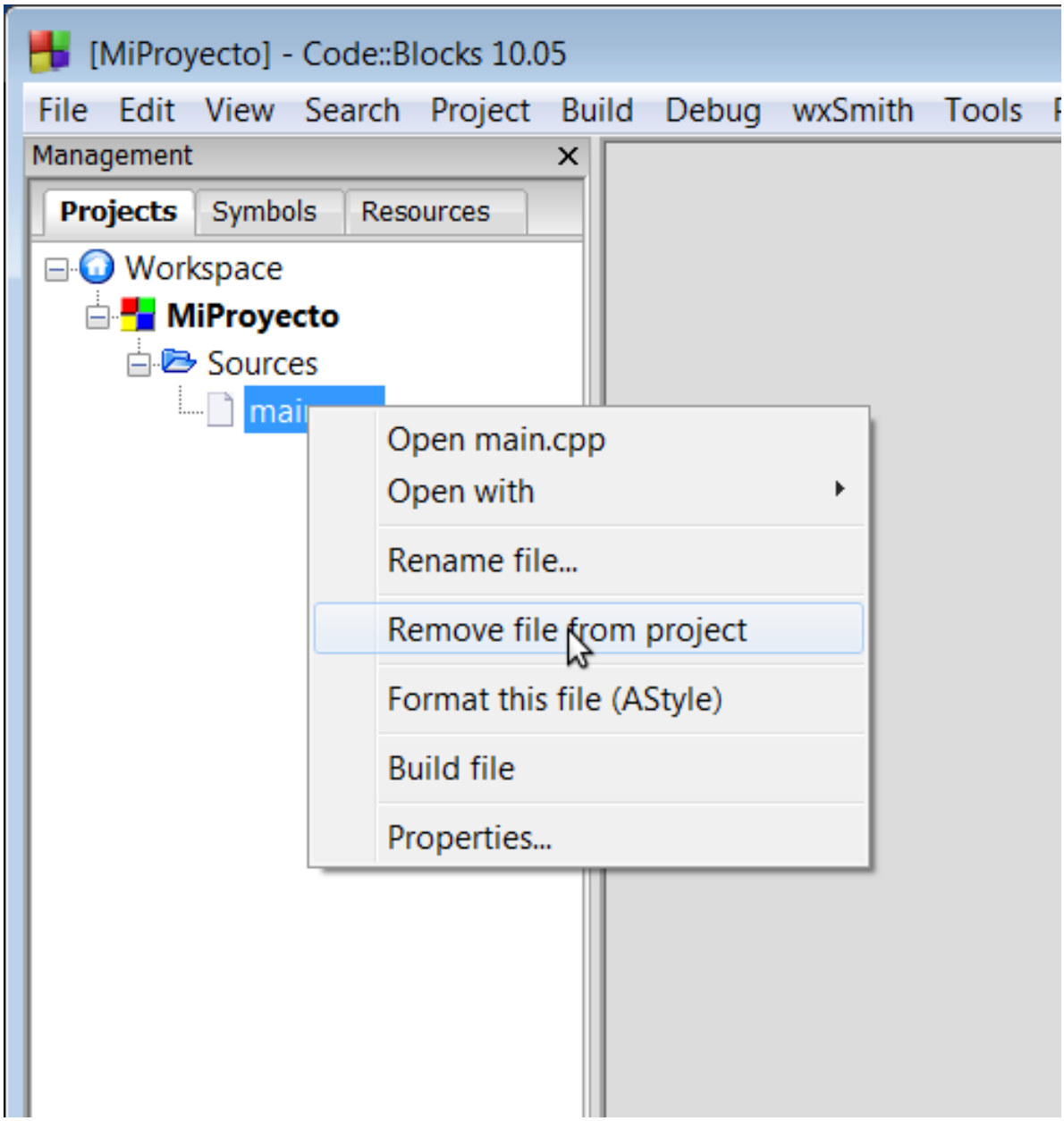
- Pon un nombre al proyecto, como por ejemplo “MiProyecto”, y luego escoge una carpeta donde ponerlo:



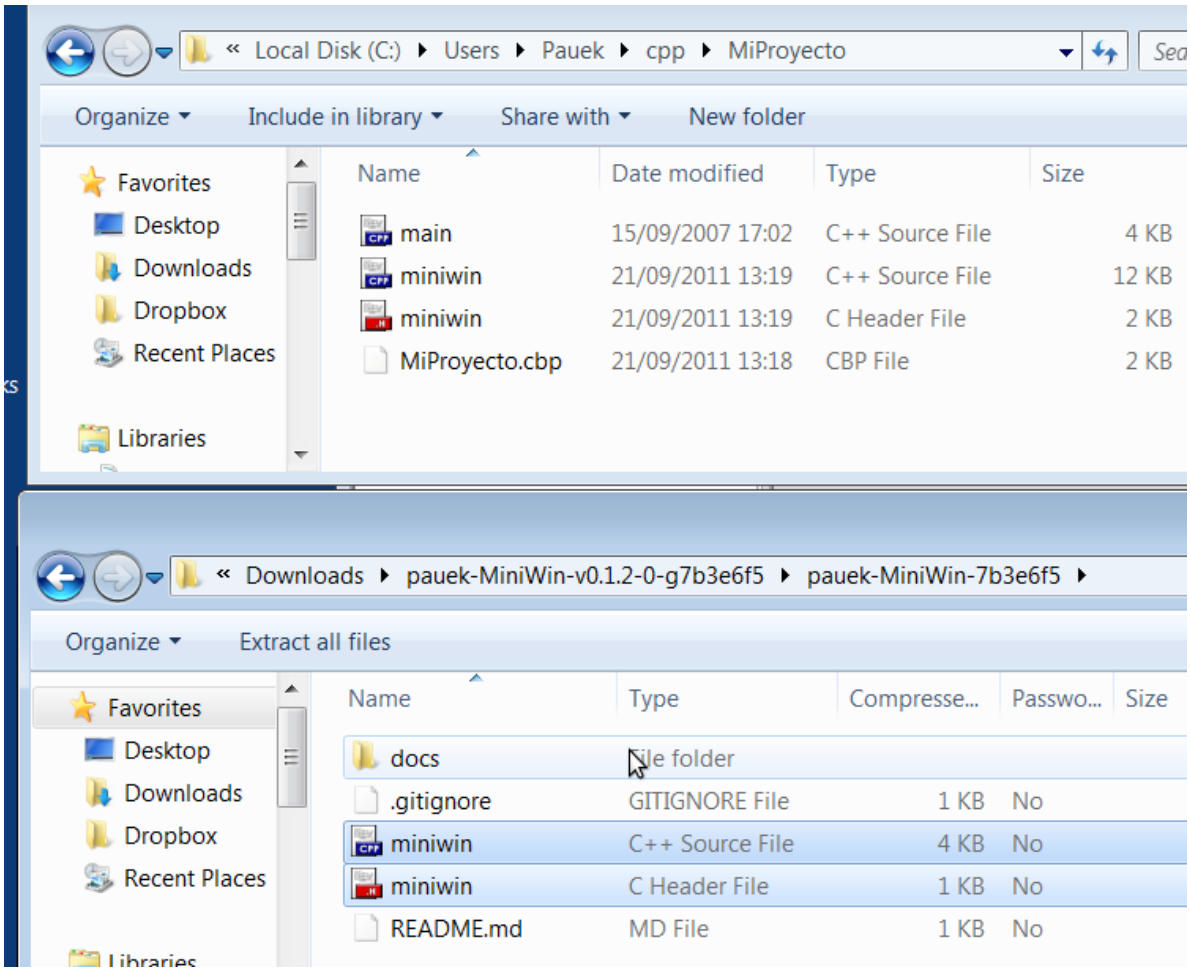
- Marca las dos configuraciones de “Release” y “Debug” (típicamente saldrán marcadas por defecto):



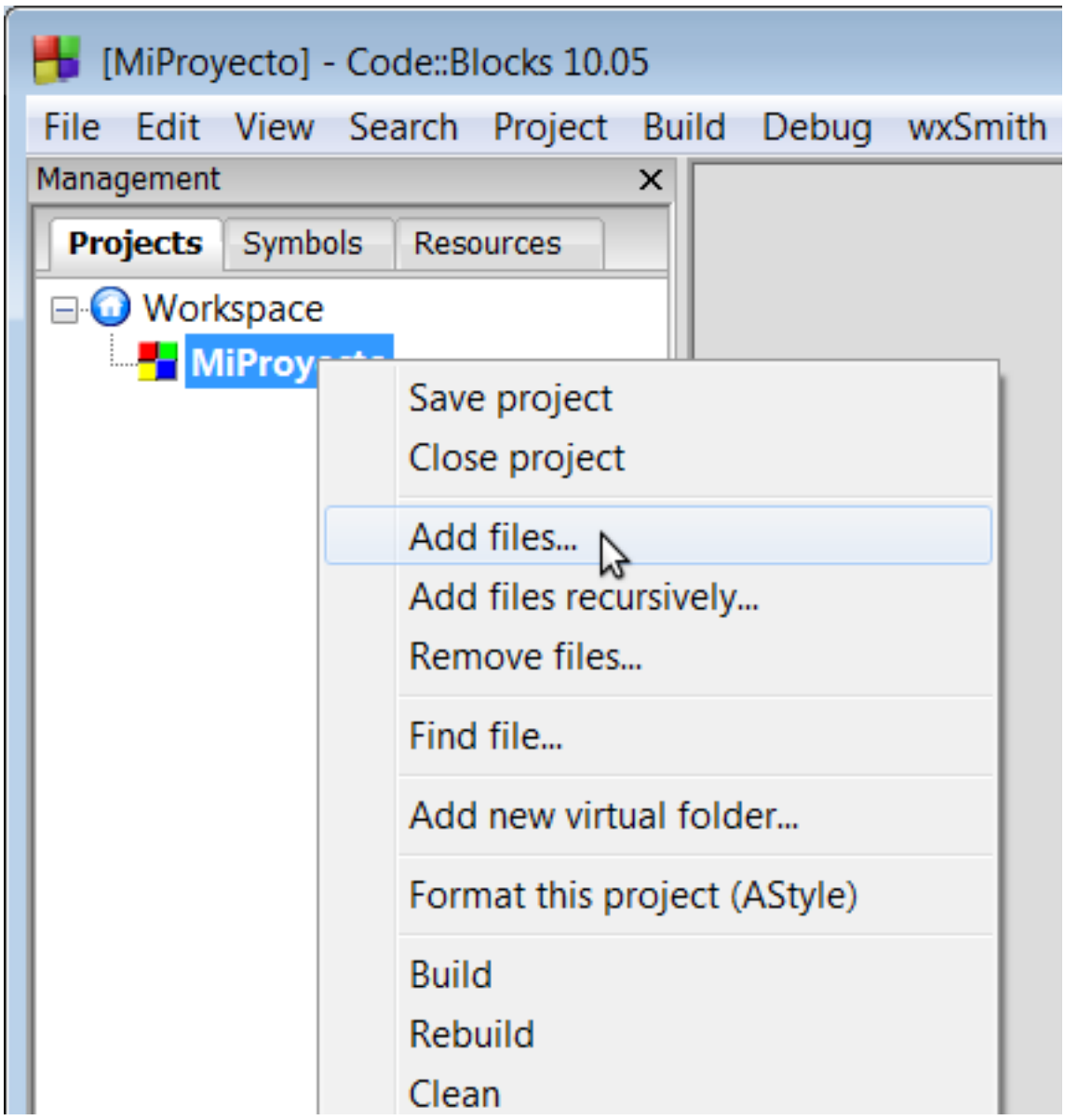
- Quita el fichero `main.cpp` del proyecto:



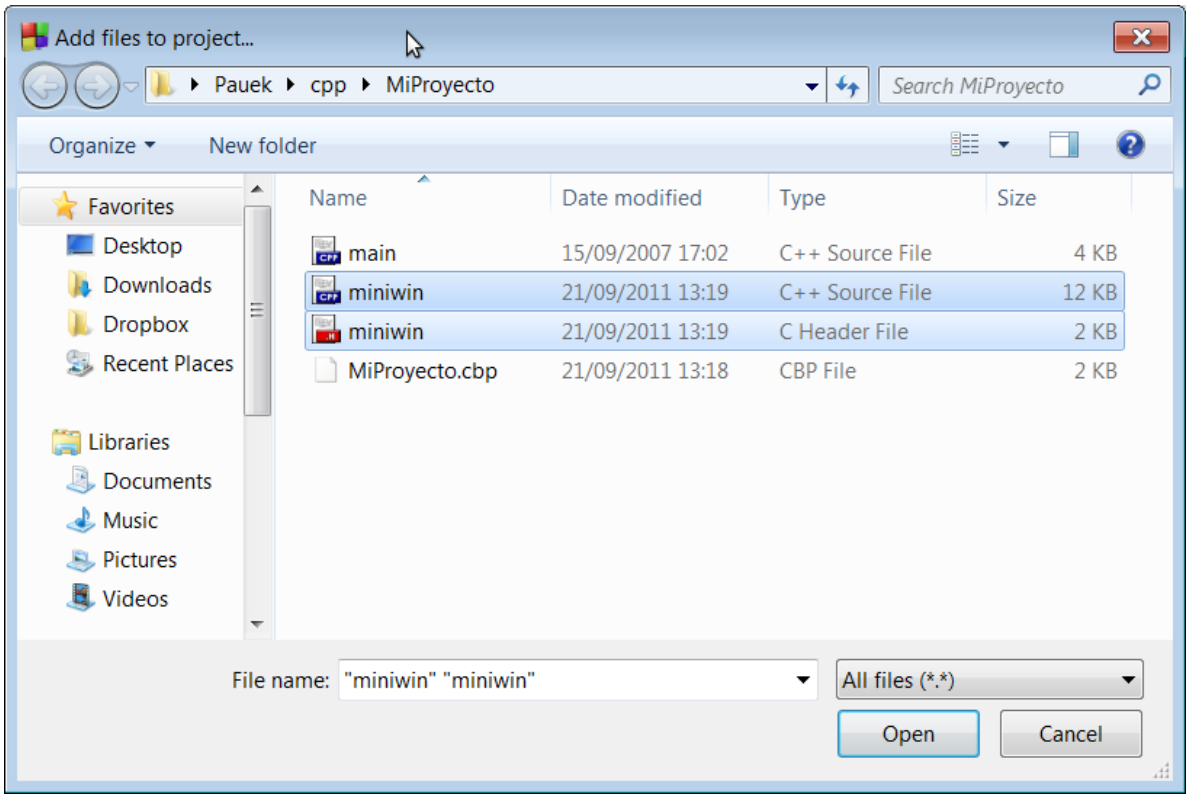
- Descarga MiniWin y copia los ficheros `miniwin.cpp` y `miniwin.h` a la carpeta del proyecto.



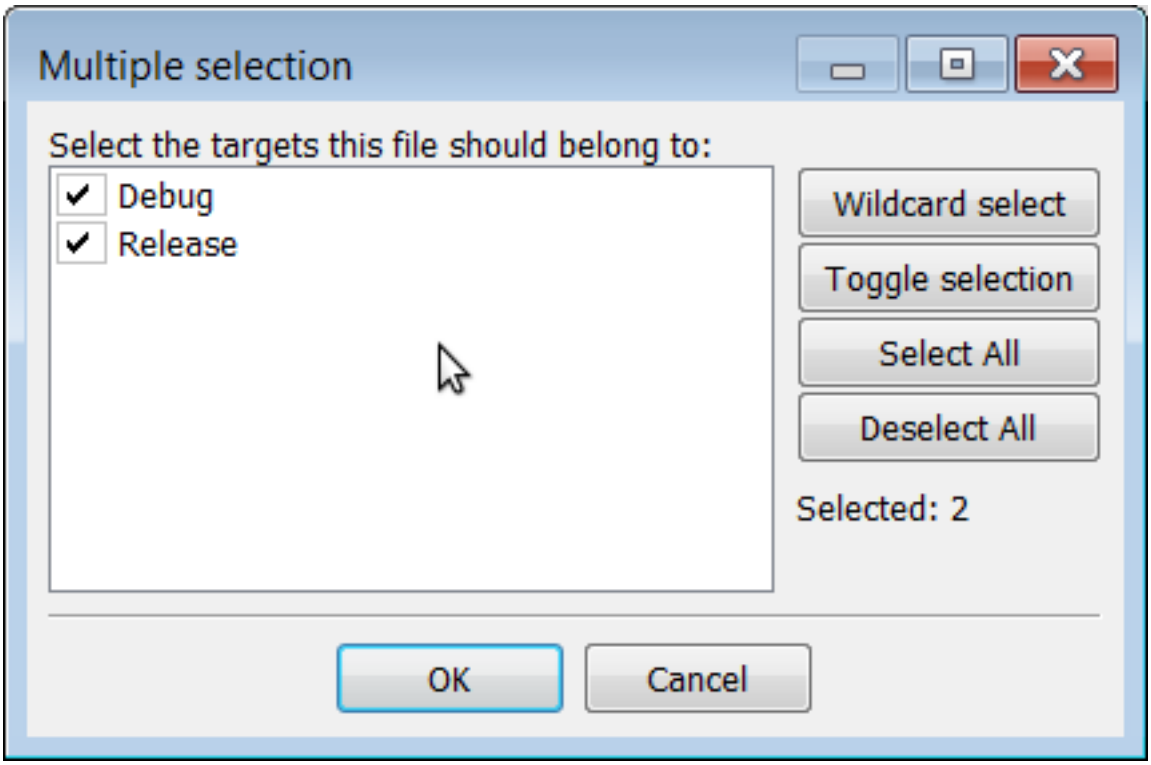
- Añade ficheros al proyecto:



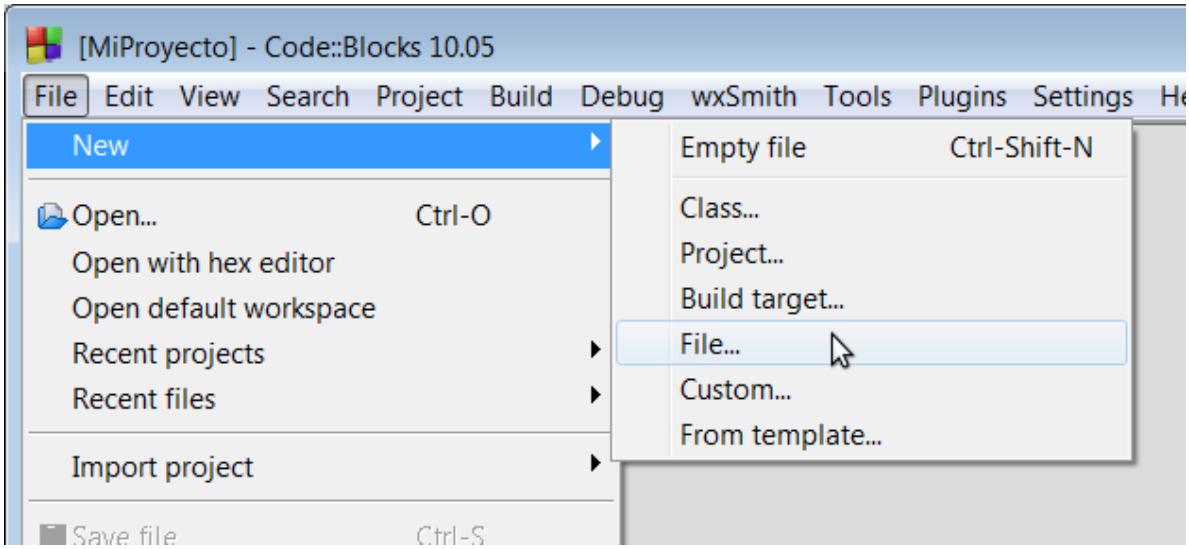
- Escoge `miniwin.cpp` y `miniwin.h` para añadirlos al proyecto:



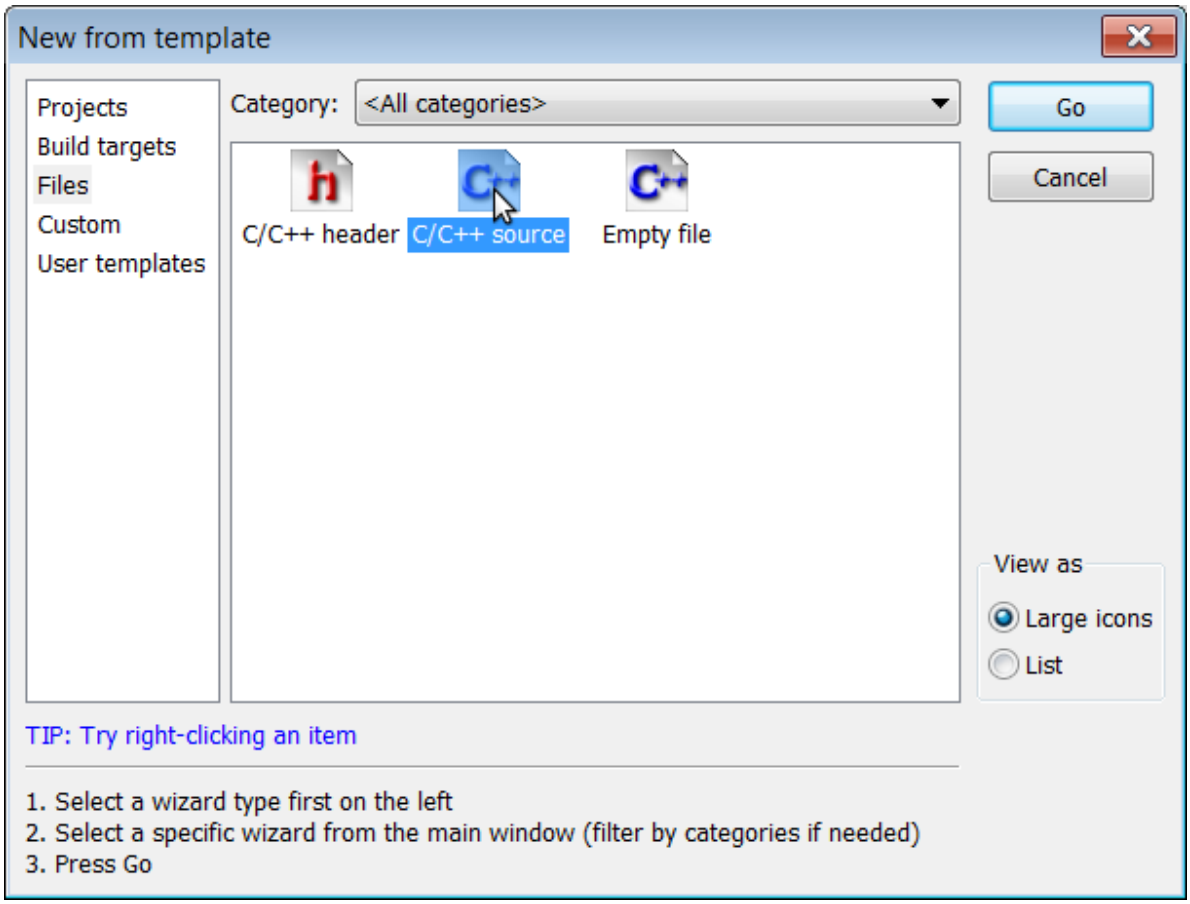
- Marca las configuraciones “Debug” y “Release” (normalmente salen marcadas por defecto):



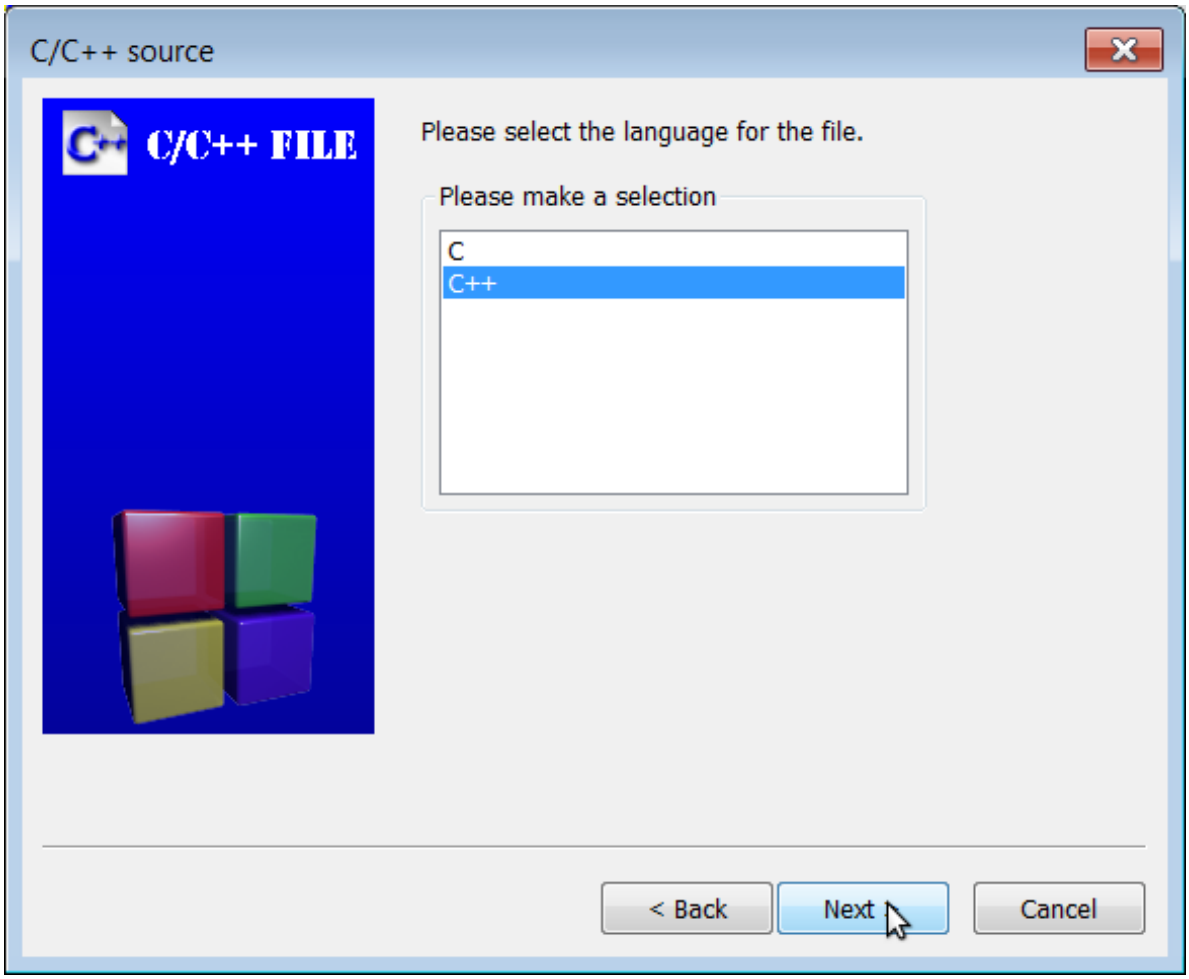
- Crea un nuevo fichero:



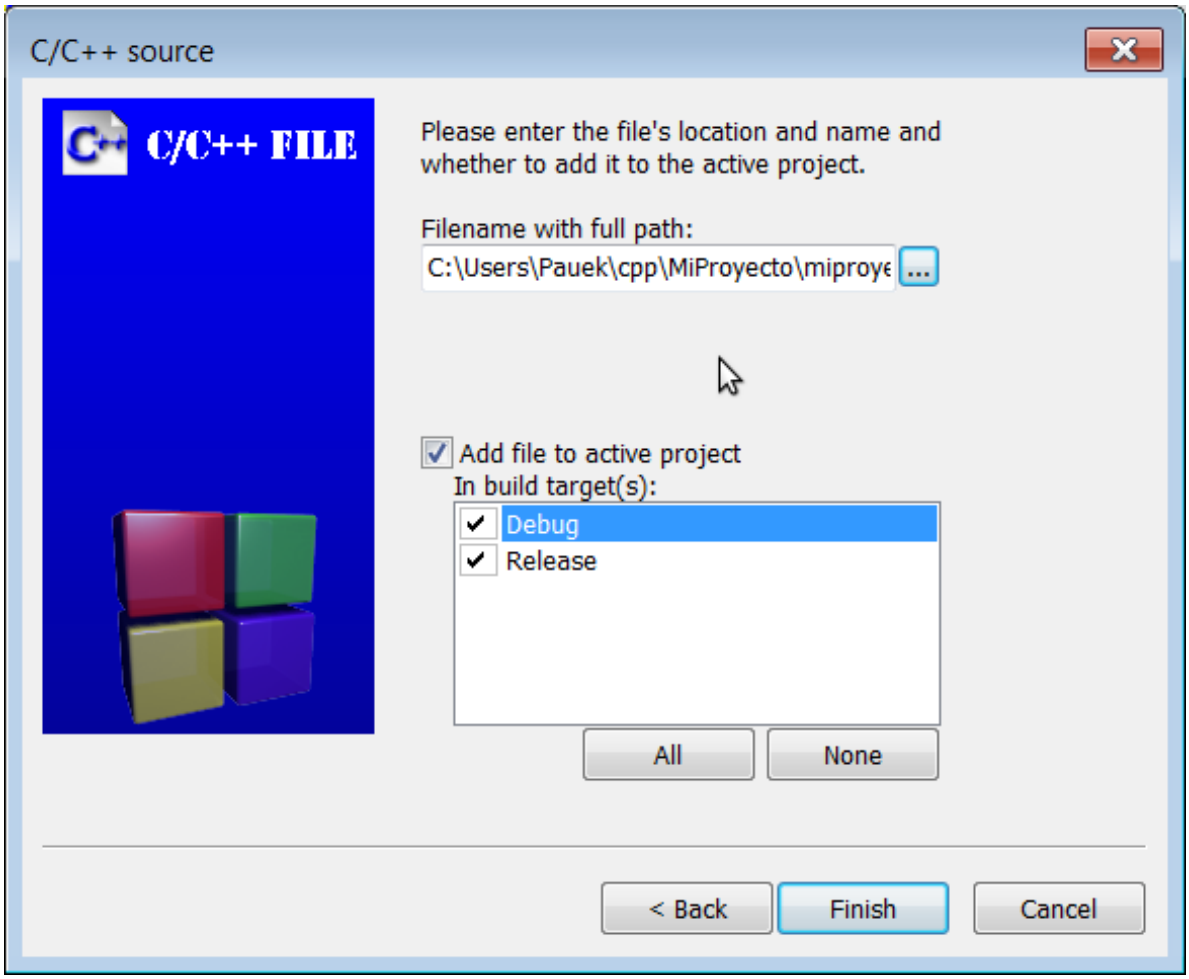
- Escoge el tipo C++/source:



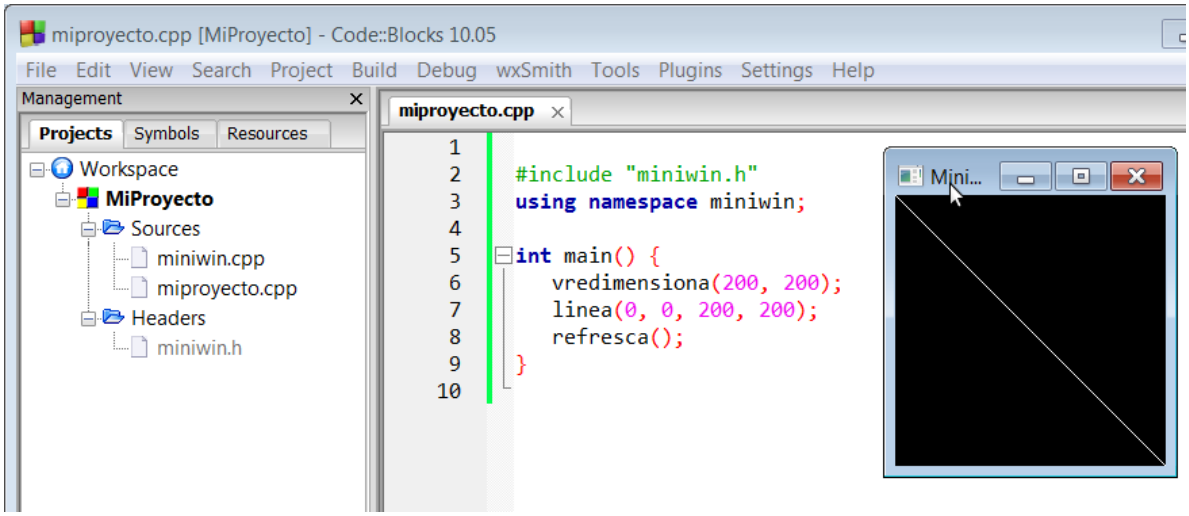
- Escoge el lenguaje C++:



- Escoge un nombre para el nuevo fichero (dentro de la misma carpeta del proyecto) y marca la opción “Add file to active project”, marcando también “Debug” y “Release”:



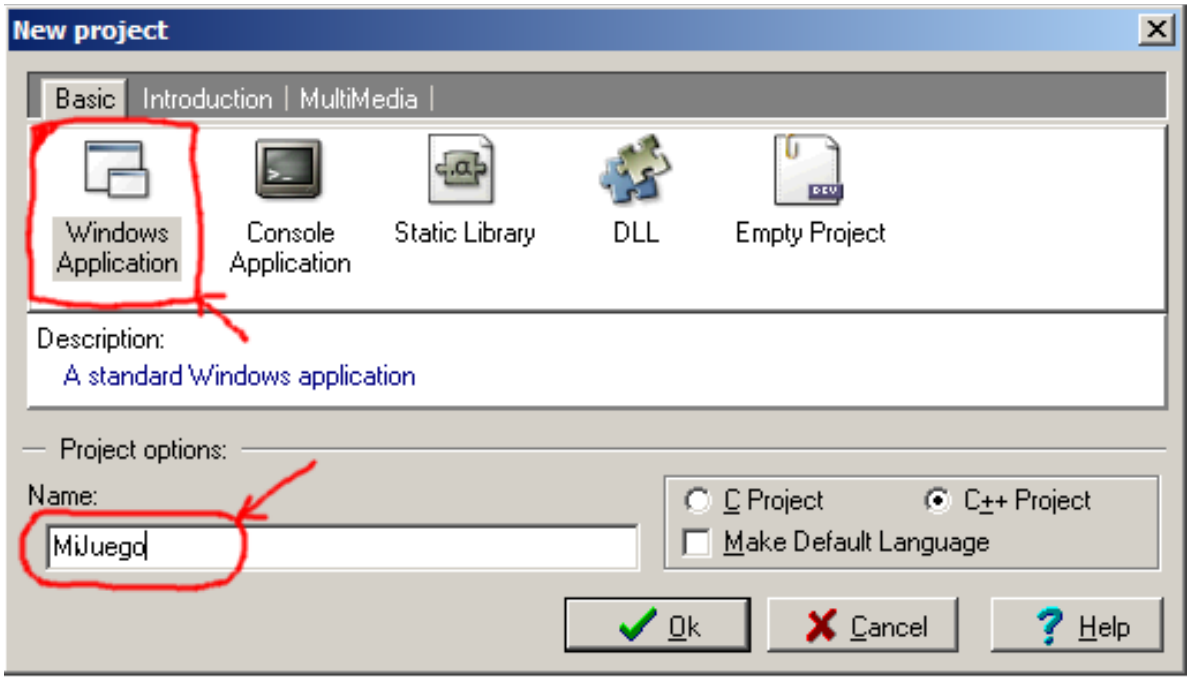
- Escribe el programa principal y compílalo:



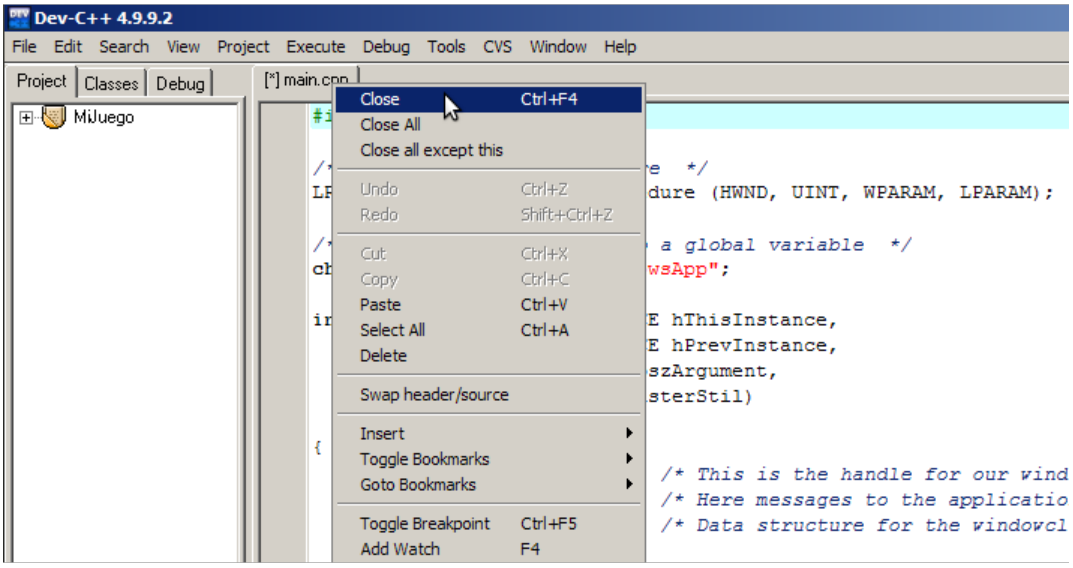
Creación de un proyecto en Dev-C++ que use MiniWin

Para crear proyectos usando MiniWin en Dev-C++, debes hacer lo siguiente:

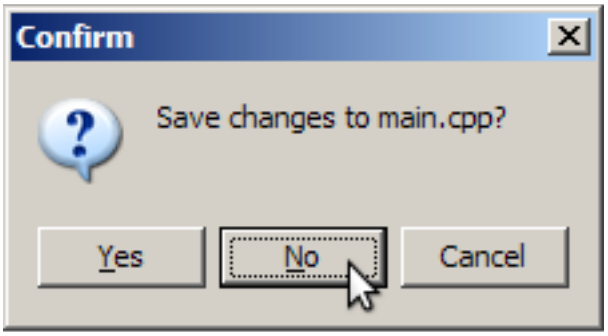
- Creas un nuevo proyecto en Dev-C++, del tipo "Windows Application", y le pones un nombre como por ejemplo "MiJuego":



- DevC++ te pedirá guardar el proyecto (un fichero `.dev`) en un directorio. Crea un directorio especial para el proyecto (supongamos "MiJuego" también).
- Ahora aparecerá un fichero `main.cpp` en la pantalla principal de edición con código C++ escrito. Este fichero lo genera Dev-C++ y es estándar. Debes cerrar la ventana (observa que el fichero no está guardado, porque tiene un asterisco entre corchetes antes del nombre):

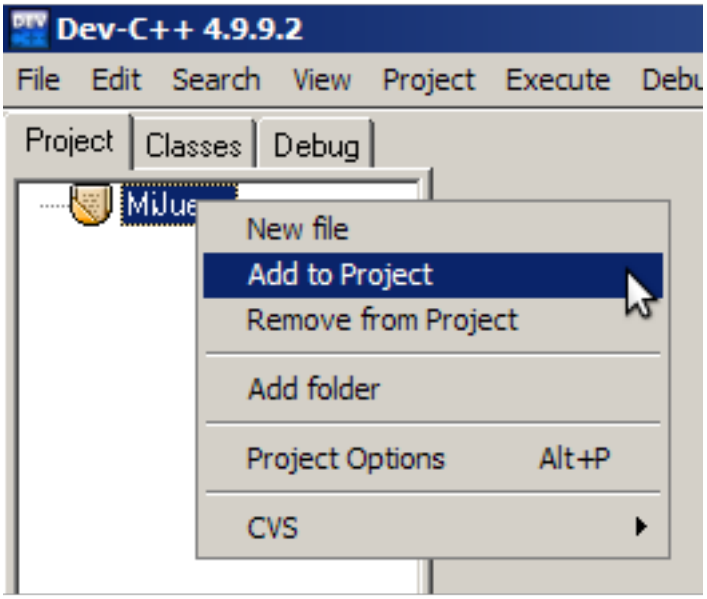


- DevC++ preguntará si quieres guardar los cambios a `main.cpp`. Dí que no:



  El proyecto quedará vacío.

- Copia los ficheros `miniwin.h` y `miniwin.cpp` en el directorio "MiJuego" que has creado anteriormente.
- Añade al proyecto de Dev-C++ el fichero `miniwin.cpp`. Esto se puede hacer clicando con el botón derecho el proyecto y luego seleccionando "Añadir al proyecto":



- Haz el programa principal (la función `main`) en un fichero aparte (por ejemplo: `mijuego.cpp`), y lo añades también al proyecto (tendrás, entonces, 2 ficheros en el proyecto, el tuyo y `miniwin.cpp`).
- En el fichero `mijuego.cpp` añades las líneas:

```
#include "miniwin.h"
using namespace miniwin;
```

  al principio de todo, tal como pones normalmente otros `#includes` o el `using namespace std`.

Ahora puedes compilar el proyecto.

Crear un proyecto en Geany que usa MiniWin

Para crear un proyecto que use MiniWin en Geany hay que seguir los siguientes pasos:

- Crea un directorio (por ejemplo "MiJuego").
- Copia los ficheros `miniwin.h` y `miniwin.cpp` al directorio "Mi Juego".
- Haz el programa principal (la función `main`) en un fichero aparte (por ejemplo: `mijuego.cpp`), y lo añades también al proyecto (tendrás, entonces, 2 ficheros en el proyecto, el tuyo y `miniwin.cpp`).
- Crea un fichero `Makefile` en el directorio "MiJuego" (con Geany mismo) y escribes lo siguiente (puedes sustituir `mijuego` por el nombre que hayas escogido):

```
all: mijuego

mijuego: miniwin.o mijuego.o
   g++ -o mijuego.exe miniwin.o mijuego.o -mwindows
```

Ahora, para compilar el proyecto, debes seleccionar la opción del menú "Construir" que pone "Compilar", pero no la primera, sino la que aparece justo al lado de "Mayúsc + F9" (o si quieres, presiona esa combinación de teclas). Mira la ventana de mensajes para comprobar que todo sale bien.

Funciones

MiniWin es *super*-simple: sólo es un miniconjunto de funciones. Para usar MiniWin solamente hay que seguir 2 pasos importantes:

- Poner arriba del programa principal:

```
#include "miniwin.h"
using namespace miniwin;
```

  Fíjate en que hay comillas dobles y no ángulos alrededor de `miniwin.h` en el `#include`. El `using namespace miniwin` te será familiar por su equivalente con `std`.

- Hacer la función `main` así:

```
int main() {
  return 0;
}
```

  Es decir, sin parámetros y devolviendo `int`. El `return 0` es obligatorio.

Aparte de eso se trata de utilizar las funciones que se comentan a continuación.

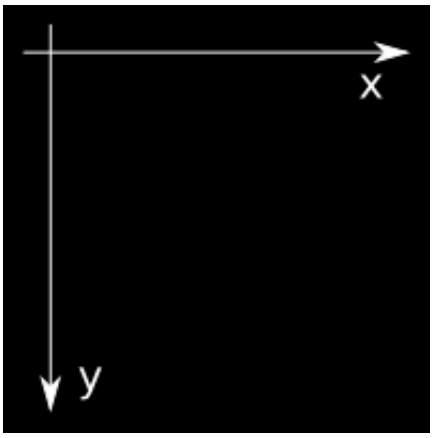
Control de la ventana

En MiniWin, al ejecutar el programa principal siempre se creará una sola ventana, a la que nos referiremos como "la ventana". Esta ventana no se puede redimensionar con el ratón (solamente con la acción *vredimensiona()*) y mantiene el dibujo que pintas aunque la minimices. Algunas funciones utilizan coordenadas en esta ventana. Las coordenadas son un par ordenado de valores, donde el primer valor es *x* y el segundo es *y*:

- La esquina superior-izquierda es el orígen, con coordenadas (0,0).

- A medida que nos desplazamos a la derecha la coordenada *x* crece.
- A medida que nos desplazamos hacia abajo la coordenada *y* crece.

El diagrama sería el siguiente:



Hay 3 funciones relacionadas con las dimensiones de la ventana.

void **vredimensiona** (int *ancho*, int *alto*)
: Acción que cambia las dimensiones (en píxeles) de la ventana. El primer parámetro indica la anchura y el segundo la altura, ambos son enteros. Por ejemplo:

```
vredimensiona(800, 600);
```

cambiará las dimensiones de la ventana a 800 por 600 píxeles. El hecho de redimensionar la ventana implica que ésta se borrará, como si hiciésemos *borra()* justo después del *vredimensiona()*.

int **vancho** ()
: Función que averigua el ancho de la ventana en píxeles, devolviendo un entero. Por ejemplo, el siguiente código utiliza la función *vancho()*:

```
int a = vancho();
if (a > 500) {
  cout << "El ancho de la ventana es mayor que 500" << endl;
}
```

int **valto** ()
: Función que averigua la altura de la ventana en píxeles, devolviendo un entero. Es similar a *vancho()*.

void **vcierra** ()
: Acción que cierra la ventana y termina el programa. Si no se llama esta acción, cuando acaba la función `main` la ventana se queda abierta mostrando el dibujo que hayamos hecho, y hay que cerrarla manualmente. Esto nos puede interesar para observar el dibujo que hayamos hecho.

Pintar en la ventana

Para pintar en la ventana hay que utilizar alguna de las acciones *linea()*, *rectangulo()*, *circulo()*, etc. y luego hay que invocar la acción *refresca()*. Esencialmente, todo lo que se pinta se acumula en un "buffer" y luego la acción *refresca()* hace visible en la ventana lo que se haya pintado previamente. Esto tiene una ventaja y un inconveniente. La ventaja es que permite pintar muchas cosas y luego refrescar solo una vez, que es importante cuando se hacen juegos. El inconveniente es que si se olvida la llamada a *refresca()*, entonces no aparece por pantalla nada de lo que se ha pintado y puede parecer que no funciona nuestro programa. En definitiva, es importante *recordar llamar a :cpp:func:'refresca' al acabar de pintar*.

Para cambiar el color con el que se pinta, hay que llamar a la función *color()* antes de pintar, es decir, todo lo que se pinta después de la instrucción:

```
color(ROJO);
```

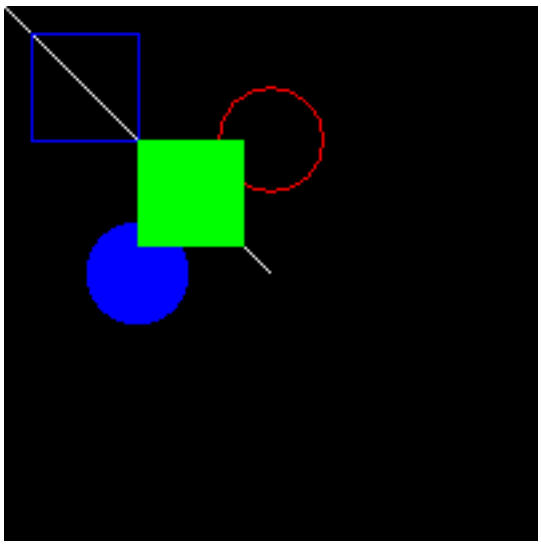
saldrá en color rojo.

Por ejemplo, el siguiente programa

```
#include "miniwin.h"
using namespace miniwin;

int main() {
  vredimensiona(200, 200);
  linea(0, 0, 100, 100);
  color(AZUL);
  rectangulo(10, 10, 50, 50);
  color(ROJO);
  circulo(100, 50, 20);
  color(AZUL);
  circulo_lleno(50, 100, 20);
  color(VERDE);
  rectangulo_lleno(50, 50, 100, 100);
  refresca();
}
```

muestra el siguiente dibujo:



En MiniWin disponemos de las siguiente acciones para pintar objetos:

void **punto** (float *x*, float *y*)
: Pinta un solo punto de la pantalla, en la posición (*x*, *y*). En función de la resolución de la pantalla esto puede costar un poco de ver.

  Por ejemplo:

```
color(AMARILLO);
punto(10, 10);
punto(9, 10);
punto(10, 9);
```

```
punto(10, 11);
punto(11, 10);
refresca();
```

pinta una minicruz centrada en la posición (10, 10).

void **linea** (float *x_ini*, float *y_ini*, float *x_fin*, float *y_fin*)
Dibuja una línea desde el punto (*izq*, *arr*), o sea *izquierda-arriba* hasta el punto (*der*, *aba*) o sea, *derecha-abajo*.

Dos líneas paralelas son, por ejemplo:

```
linea(0,  0, 100,  0);
linea(0, 10, 100, 10);
```

void **rectangulo** (float *izq*, float *arr*, float *der*, float *aba*)
Dibuja el borde de un rectángulo con coordenadas horizontales (las *x*) *izq* y *der* y verticales (las *y*) *arr* y *aba*.

void **rectangulo_lleno** (float *izq*, float *arr*, float *der*, float *aba*)
Dibuja un rectángulo relleno con coordenadas horizontales (las *x*) *izq* y *der* y verticales (las *y*) *arr* y *aba*.

void **circulo** (float *x_cen*, float *y_cen*, float *radio*)
Dibuja una circumferencia con el centro en (*x_cen*, *y_cen*) y con un cierto *radio*. Si llamamos:

```
circulo(50, 100, 20);
```

Aparecerá un círculo de radio 20 con centro en el punto (50, 100)

void **circulo_lleno** (float *x_cen*, float *y_cen*, float *radi*)
Dibuja un círculo (relleno) con el centro en (*x_cen*, *y_cen*) y con un cierto *radio*.

void **texto** (float *x*, float *y*, **const** std::string &*texto*)
Pinta un texto a partir de la posición (*x*, *y*) con el contenido del parámetro *texto*.

Para cambiar el color tenemos las 2 acciones siguientes:

void **color** (int *c*)
Cambia el color a partir de un número entre 0 y 7 ambos incluidos. MiniWin define las siguientes 8 constantes para no tener que recordar a qué color corresponde cada número: `NEGRO`, `ROJO`, `VERDE`, `AZUL`, `AMARILLO`, `MAGENTA`, `CYAN` y `BLANCO`.

void **color_rgb** (int *r*, int *g*, int *b*)
Cambia el color a un valor RGB arbitrario. Los valores de los parámetros *r*, *g* y *b* deben estar entre 0 y 255 ambos incluidos. Por ejemplo:

```
color_rgb(128, 0, 0);
```

selecciona un color rojo apagado.

Después de llamar a las funciones para pintar objetos, es muy importante llamar a la acción *refresca()*:

void **refresca** ()
Pone en la ventana todo los objetos pintados acumulados. Muy importante llamar a esta acción después de pintar. Por ejemplo, para pintar una línea (que se añade a todo lo anterior:

```
linea(0, 0, 100, 100);
refresca();
```

void **borra** ()
Borra el "buffer" interno de pintado. Todo lo que se pinte después se pintará sobre un fondo negro, como

al principio. Para borrar la pantalla y dejarla negra, habría que llamar a *refresca()* después de llamar a *borra()*. O sea, para borrar la pantalla completamente y hacerlo visible:

```
borra();
refresca();
```

Funciones para teclas

En MiniWin se puede determinar si el usuario acaba de presionar una tecla (o hace un rato), con la función *tecla()*. Esta función no espera a que el usuario presione una tecla ya que esto interrumpiría el programa totalmente. Por eso uno de los valores que puede devolver *tecla()* es el valor `NINGUNA` para indicar que el usuario no ha presionado una tecla recientemente. Las teclas que se presionan se guardan en un *buffer* y cada llamada a *tecla()* las retorna una por una.

int **tecla**()

Devuelve un entero para indicar si se acaba de presionar una tecla. Si se han presionado varias, se puede llamar a *tecla()* varias veces y se irán obteniendo las teclas presionadas por orden.

Si no se ha presionado ninguna tecla, se devuelve la constante `NINGUNA`

Para evitar tener que recordar qué enteros representan a cada tecla, MiniWin define las siguientes constantes:

Tecla	Constante
Ninguna tecla	`NINGUNA`
Esc	`ESCAPE`
Barra espaciadora	`ESPACIO`
Return	`RETURN`
Flecha Arriba	`ARRIBA`
Flecha Abajo	`ABAJO`
Flecha Derecha	`DERECHA`
Flecha Izquierda	`IZQUIERDA`
Teclas de Función	`F1`, `F2` hasta `F10`

Para las letras y los números, el código devuelto es el mismo código ASCII. Para los números entre 48 y 57 y para las letras entre 65 y 90 (las mayúsculas).

El siguiente ejemplo detecta la presión de una tecla y si es la letra `A` se muestra un mensaje:

```
int t = tecla();
if (t == int('A')) {
  mensaje("Has presionado la tecla 'A'");
}
```

Funciones para el ratón

En MiniWin se puede saber donde está situado el puntero del ratón y también si los botones derecho y/o izquierdo estan presionados. Las funciones disponibles son las siguientes:

bool **raton**(float &*x*, float &*y*)

Obtiene las coordenadas de la posición del ratón (`x` e `y`), y además devuelve un Booleano que indica si el ratón se encuentra dentro de la ventana. Si el ratón está fuera de la ventana los valores `x` e `y` no serán reales y hay que ignorar su valor.

bool **raton_dentro**()

Retorna un valor Booleano indicando si el ratón se encuentra dentro de la ventana. Es importante llamar a esta

función antes de obtener las coordenadas de la posición del ratón ya que estas son correctas solamente cuando el ratón está dentro de la ventana.

float **raton_x**()
Devuelve la coordenada `x` de la posición del ratón. Esta posición es válida si el ratón se encuentra dentro de la ventana, por eso es necesario antes llamar a *raton_dentro()* o llamar directamente a *raton()*.

float **raton_y**()
Devuelve la coordenada `y` de la posición del ratón. Esta posición es válida si el ratón se encuentra dentro de la ventana, por eso es necesario antes llamar a *raton_dentro()* o llamar directamente a *raton()*.

void **raton_botones**(bool *&izq*, bool *&der*)
Obtiene el estado de los botones derecho e izquierdo del ratón. Si el valor Booleano `izq` es cierto el botón izquierdo está presionado, y lo mismo ocurre para `der` con el botón derecho.

bool **raton_boton_izq**()
Retorna un booleano indicando si el botón izquierdo está presionado.

bool **raton_boton_der**()
Retorna un booleano indicando si el botón derecho está presionado.

Funciones para mensajes

En MiniWin se puede mostrar un mensaje al usuario a través de una pequeña ventana. También se puede hacer una pregunta (tipo Sí/No) usando una ventanita especial.

void **mensaje**(string *msj*)
Muestra una pequeña ventana auxiliar con el mensaje *msj* y espera que se presione el botón "Ok". *[Esta función aún no está implementada en Linux]*

Por ejemplo, si llamas a la acción así:

```
mensaje("MiniWin es fácil");
```

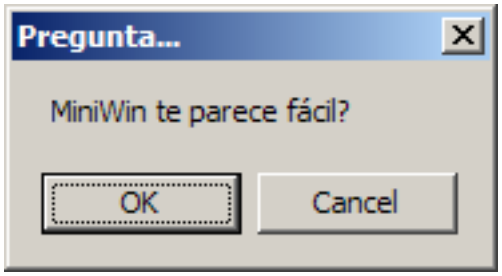
aparecerá una ventana como la siguiente:

bool **pregunta**(string *msj*)
Muestra una ventana auxiliar con la pregunta que se pasa en *msj* y devuelve `true` o `false` en función de la contestación del usuario. *[Esta función aún no está implementada en Linux]*

Por ejemplo, si llamas a la acción así:

```
pregunta("MiniWin te parece fácil?");
```

aparecerá una ventana como la siguiente:



Otras funciones

void **espera** (int *miliseg*)

Detiene el programa un número de milisegundos especificado por *miliseg*. Por ejemplo, al hacer:

```
espera(1000);
```

el programa se detiene durante un segundo.

Index