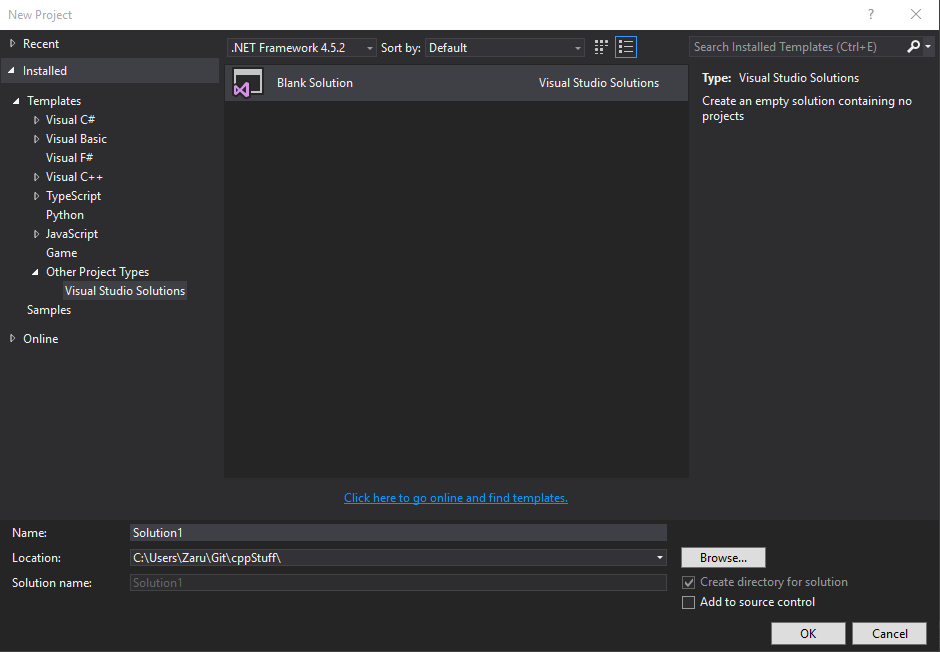
# Guía para organizar un proyecto en visual studio 2015

Creamos una nueva solución de la siguiente manera:

## File -> New Project -> Templates -> Other Project Types -> Visual Studio Solutions

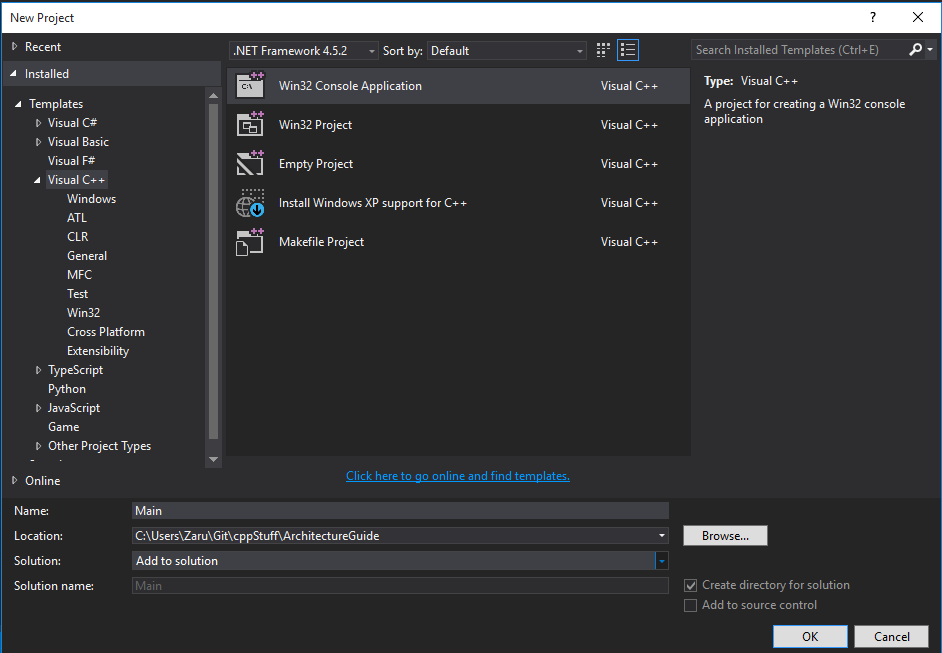
### Y creamos una solución en blanco llamándolo “ArchitectureGuide”



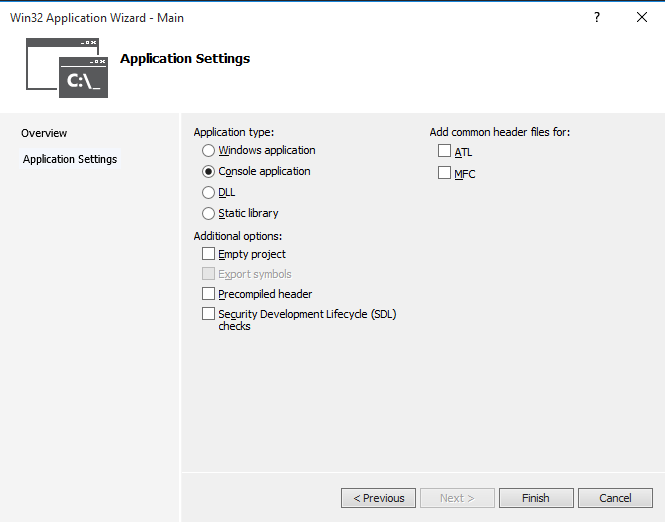
Nada más crearlo, añadiremos a esta solución un proyecto nuevo de la siguiente manera:

## File –> New Project -> Templates -> Visual C++.

Y le damos a “Win32 Console Application” diciéndole que lo añada a la solución que ya tenemos.



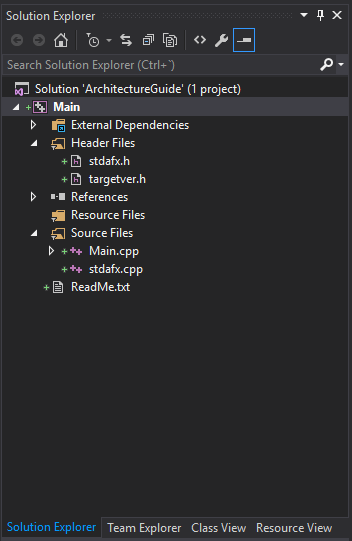
Al darle a Ok nos saldrá una ventana blanca en el cual podremos modificar unos cuantos parámetros dándole a **NEXT.**



Desmarcamos las dos opciones de más abajo por que para este ejemplo no lo necesitaremos (si quieres más información acerca de las opciones busca sobre ello)

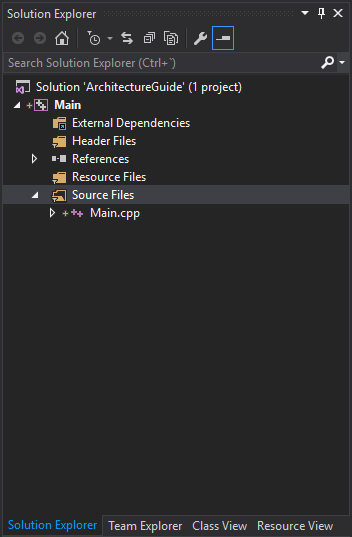
Y Finalizamos el proceso de crear el proyecto.

Al crearlo veremos algo de este estilo:

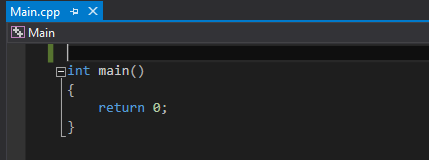


Los archivos **stdafx.h, targetver.h y stdafx.cpp** tienen que ver con el preprocesado de las cabeceras para optimizar el código, pero como lo hemos desactivado (las dos casillas que antes desmarcamos) nos crea los archivos por si en algún momento de nuestro proyecto queremos volver a activarlos. Como no lo queremos para nada los borramos del proyecto y del disco junto con el **ReadMe.txt.**

Después de esto el proyecto quedará de la siguiente manera:



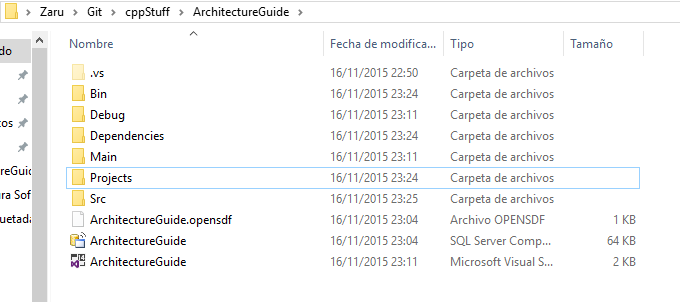
Y modificaremos el **Main.cpp** para que no nos de errores de compilación quitando el **#include “stdafx.h”** quedándonos el fichero de la siguiente manera:



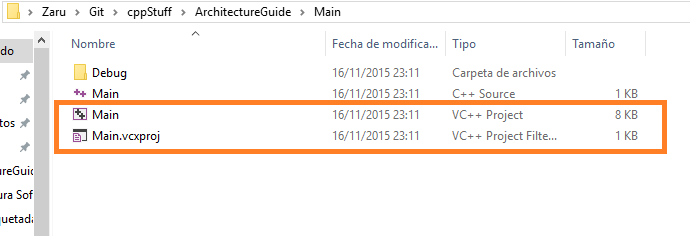
Tras esto, vamos a explicar la organización del proyecto:

* **ArchitectureGuide** (Raiz del Proyecto)
  + **Bin** : Aquí irán todos los documentos que necesitaremos para realizar el juego, música, assets, niveles etc. Y también tendremos aquí el ejecutable.
  + **Dependencies** : Aquí meteremos las librerías que tengamos.
    - **Include:** Tendremos la cabecera de todas las librerías que utilicemos.
    - **Lib:**
  + **Projects** : Aquí tendremos carpetas de nuestros proyectos y dentro de ellas tendremos los archivos **NombreProyecto** y **NombreProyecto.vcxproj** (Después veremos un ejemplo de esto por si queda aquí muy claro).
  + **Src** : Aquí tendremos carpetas de nuestros proyectos, y dentro de ellas el código utilizado.
  + **ArchitectureGuide.sln** : Esto es por vaguería, para tener un acceso rápido al proyecto.

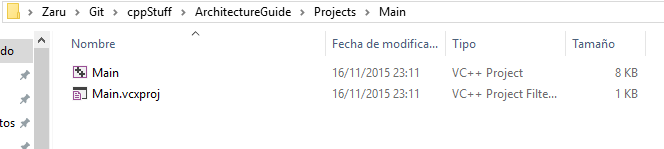
Pues vamos a la obra, lo primero que haremos será crear las carpetas que explicamos más arriba quedando la raíz del proyecto de la siguiente manera.



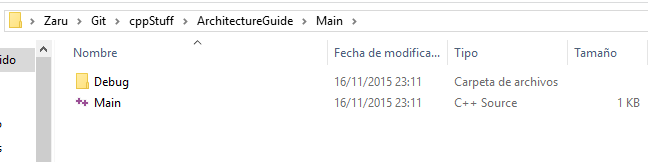
En la carpeta **Projetcs** crearemos una carpeta con el nombre de nuestro proyecto, en este caso nuestro proyecto se llama **Main** y dentro de ella pegaremos los archivos que señalo en la imagen que se encuentran en **ArchitectureGuide/Main:**

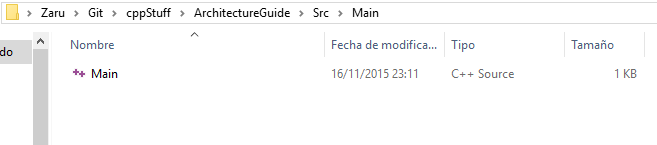


## Cortamos y las pegamos en ArchitectureGuide/Projects/Main:

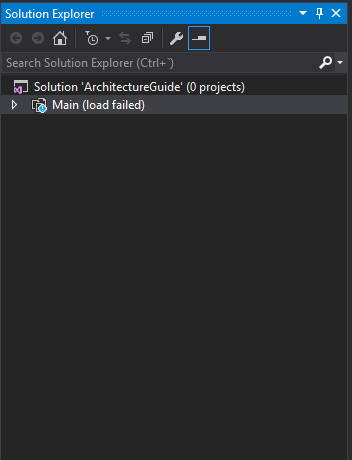


De la misma manera, cogeremos el código de nuestro proyecto (en este caso solo Main.cpp) que está en **ArchitectureGuide/Main** y lo pegaremos en **ArchitectureGuide/Src/Main** (La carpeta **Main** habrá que crearla):



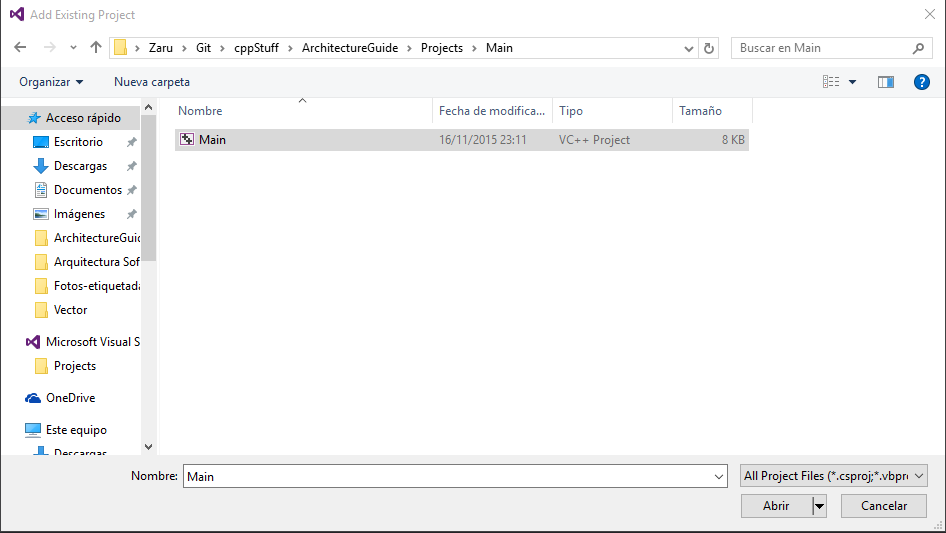


Después de esto abriremos el proyecto desde **ArchitectureGuide.sln** y saldrá algo parecido a esto:

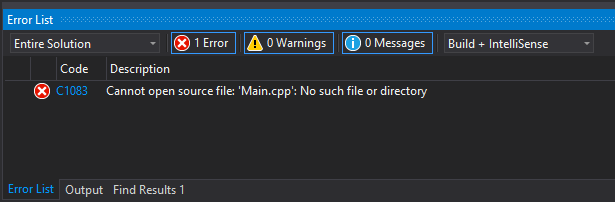


Es obvio que salga esto, hemos quitado los archivos de las ubicaciones anteriores a otras, por ello vamos a configurar el proyecto para que lo cargue bien.

Lo primero que haremos será borrar los proyectos fallidos, en este caso el **Main**. Para añadir otra vez nuestro proyecto hacemos **click derecho en la solución/Add/Existing Project** y buscamos el archivo que necesitamos en **ArchitectureGuide/Projects/Main** y lo abrimos.

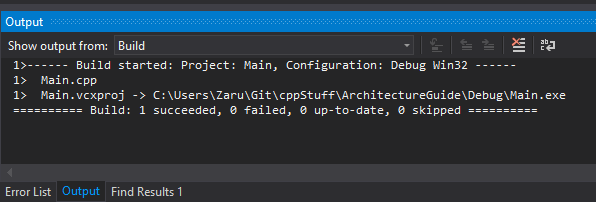


Y tendremos la organización del proyecto que teníamos antes, pero con el problema de que al compilar nos muestra el siguiente error:



Esto sucede porque visual studio tiene la referencia del Main.cpp que teníamos en el antiguo directorio. Para solucionarlo borramos el archivo en el explorador de soluciones y arrastramos **Main.cpp** de la ruta **ArchitectureGuide/Src/Main:**

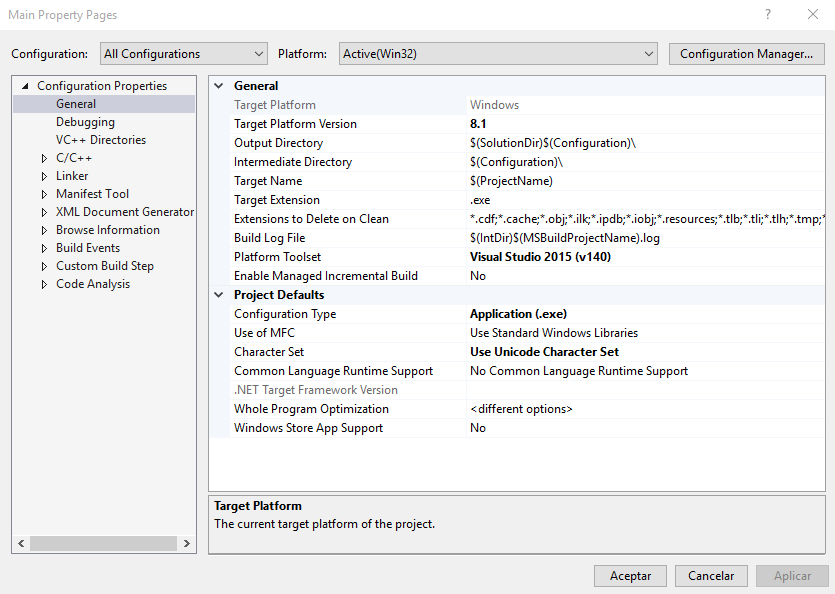
Y ahora compila perfectamente:



Para seguir teniendo el proyecto organizado vamos a cambiar la ruta que tiene por defecto visual studio de crear los ejecutables y meterlos en la carpeta bin.

Para ello nos vamos al explorador de soluciones y **hacemos click con el botón derecho sobre el proyecto Main -> Properties** saliéndonos una ventana blanca.

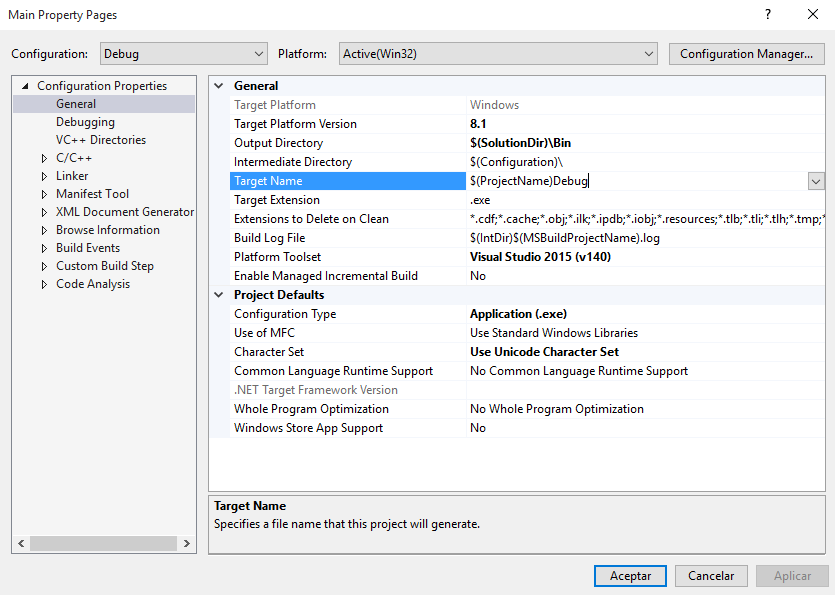
Seleccionamos en Configuration: All Configurations y nos dirigimos a Confirmation Properties -> General:

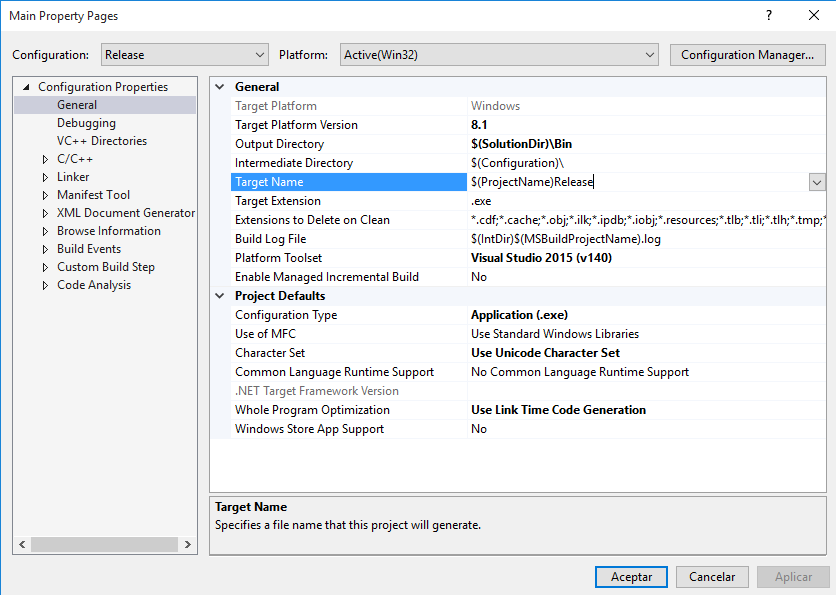


Y vamos a tocar el campo **Output Directory** poniendo lo siguiente: **$(SolutionDir)\Bin,** al final lo que le estamos diciendo es que cree los ejecutables en la siguiente ruta absoluta: **C:\Users\Zaru\Git\cppStuff\ArchitectureGuide\Bin**

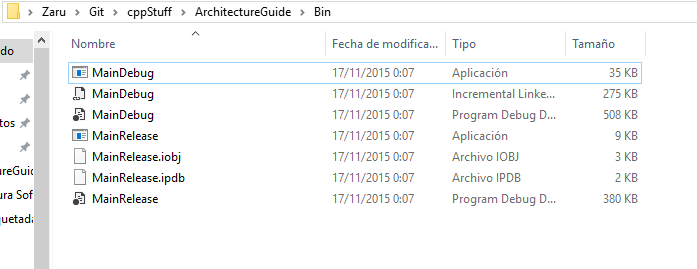
Obviamente en otro PC tendrás otra ruta dependiendo de donde hayas creado tu proyecto, de ahí viene lo de poner **$(SolutionDir)** para generalizar la ruta si vas a utilizar el proyecto en otro PC.

Por ultimo cambiamos en **Configuration a Debug y Release** dando los valores de las imágenes en el campo **Target Name**, que esto lo hacemos para que no se solapen los ejecutables de Debug y Release.

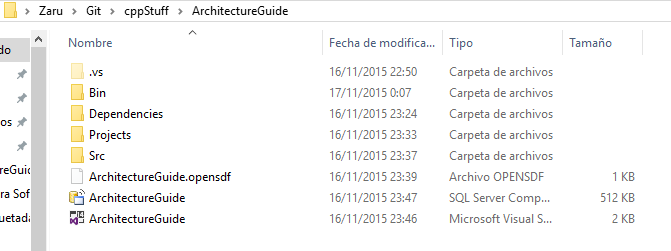


****

Ahora si probamos a compilar en modo Debug y en modo Release podemos comprobar que nos ha generado los ejecutables y algunos archivos más.



Para ir quitándonos cosas que no nos sirven para nada, como acabamos de cambiar el lugar donde se crean los ejecutables, podemos borrar la carpeta **Debug** que hay en la raíz y la carpeta **Main**, quedándonos la raíz del proyecto como sigue:

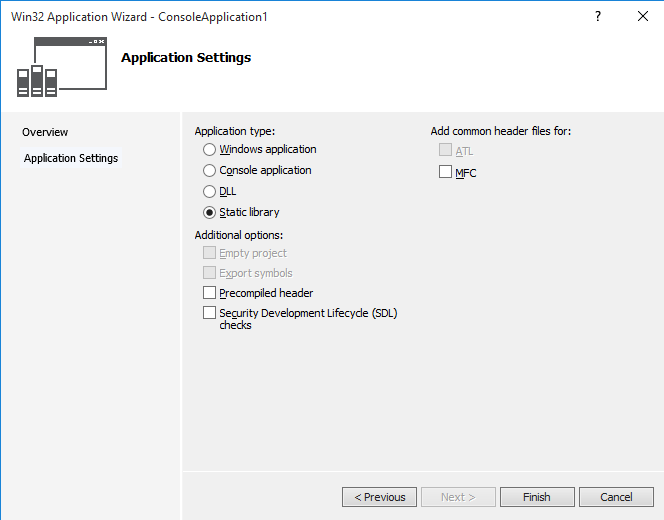


Todo Perfecto por ahora, ¿Pero qué pasa si tenemos una Biblioteca? Vamos a ver como se haría y como se configura.

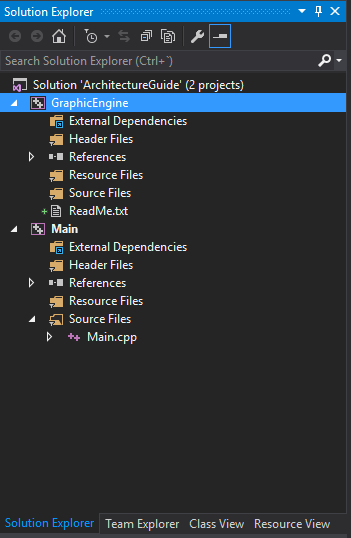
Para crear una Biblioteca con el nombre de **GraphicEngine** hacemos lo siguiente:

## Botón derecho en Solution -> Add -> New Project -> Visual C++ -> Win32 Console Application

Y nos saldrá otra vez la pantalla blanca de hace un rato, pero esta vez clicáremos la opción **Static library** en vez de **Console application**.

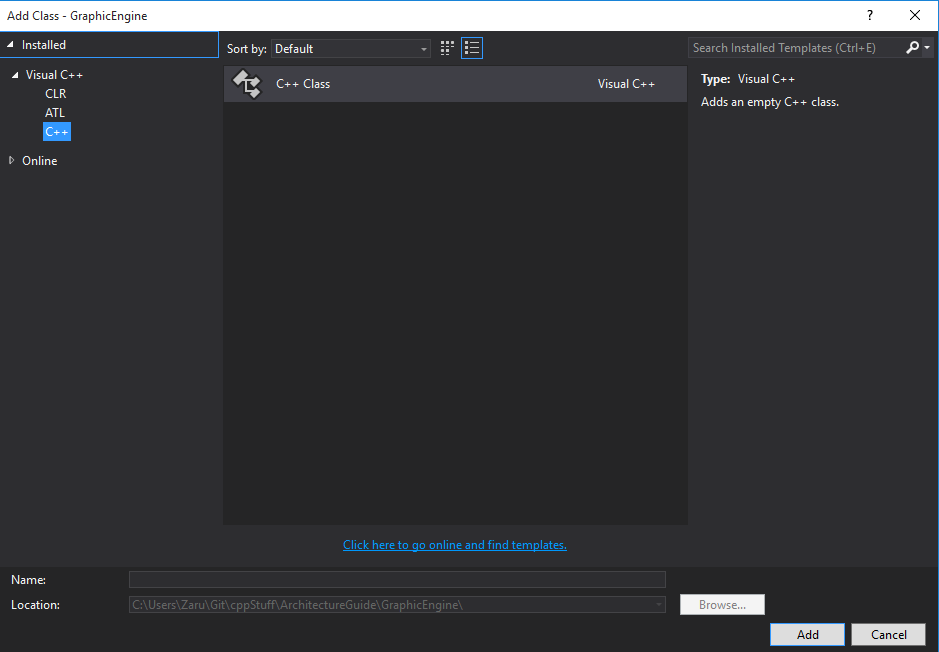


Después de esto así quedará nuestro explorador de soluciones:

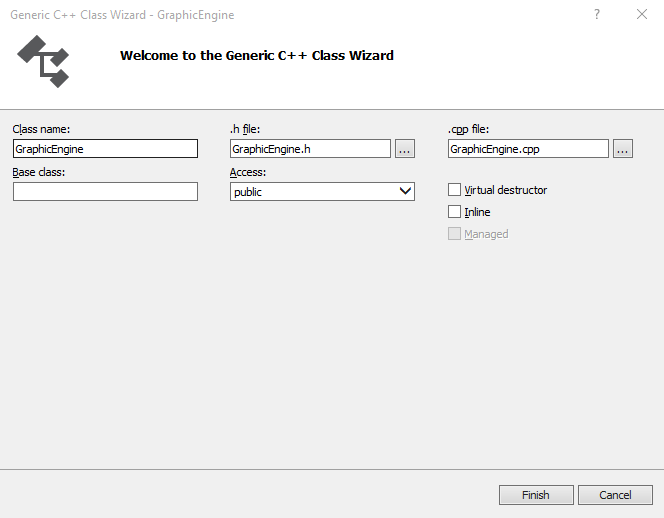


Como no nos interesa el **ReadMe.txt** lo borramos y añadimos una nueva clase al proyecto de la siguiente manera:

### Botón Derecho en GraphicEngine -> Add -> class



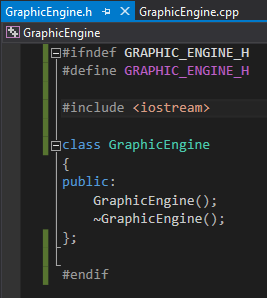
Seleccionamos c++ Class y le damos a **Add.** Al darle nos saldrá otra pantalla para ponerle nombre a nuestro proyecto al cual llamaremos **GraphicEngine**:



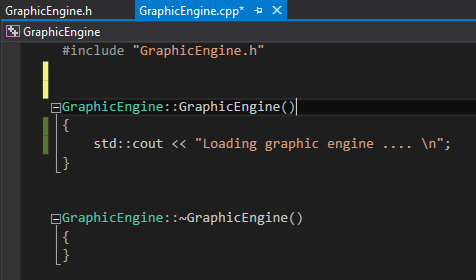
Creándonos en el explorador de soluciones dos nuevos documentos, **GraphicEngine.h** y **GraphicEngine.cpp.**

Rellenamos estos dos archivos hasta tenerlos como la imagen:

## GraphicEngine.h

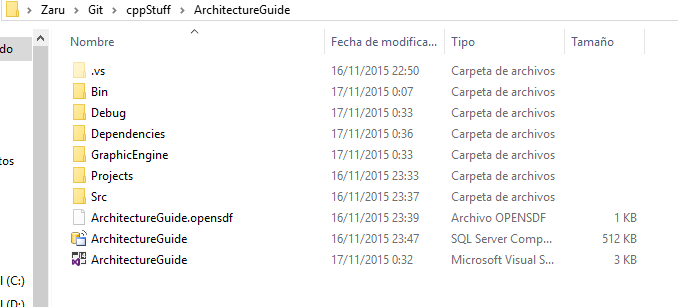
****

## GraphicEngine.cpp

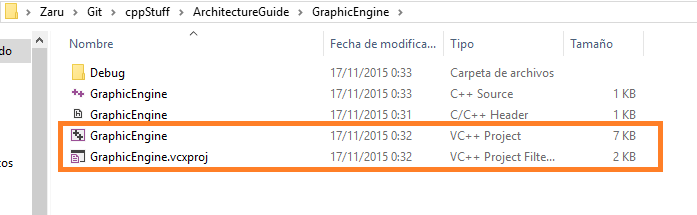
****

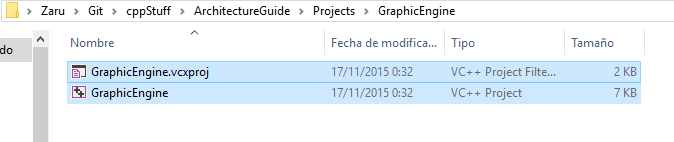
Y veremos que compila perfectamente.

Volvemos a organizar los archivos en nuestra carpeta raíz, que vemos que tenemos una carpeta nueva llamada **GraphicEngine:**

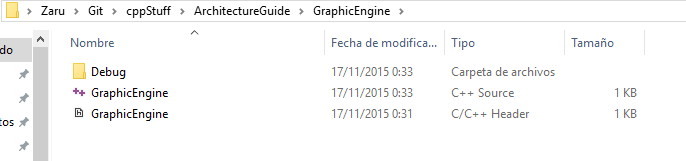


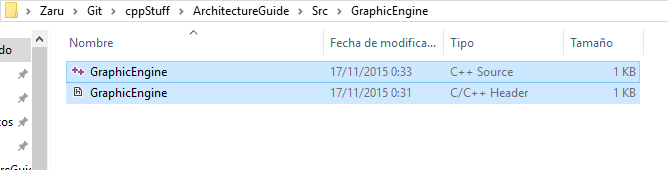
Y como hemos hecho antes desglosaremos los archivos de esta carpeta de manera organizada. Como hicimos la primera vez, los archivo de proyecto que están en **ArchitectureGuide/GraphicEngine** los deplazaremos a **ArchitectureGuide/Projects/GraphicEngine** (Creando la carpeta GraphicEngine)



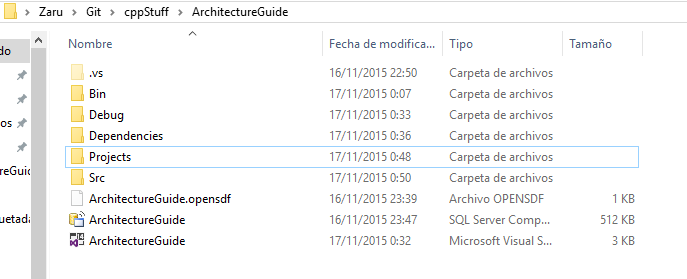


Y haremos de la misma manera con el código que está en **ArchitectureGuide/GraphicEngine** a **ArchitectureGuide/Src/GraphicEngine** (Creando la carpeta GraphicEngine):





Y podemos eliminar la carpeta **GraphicEngine** de la raíz:



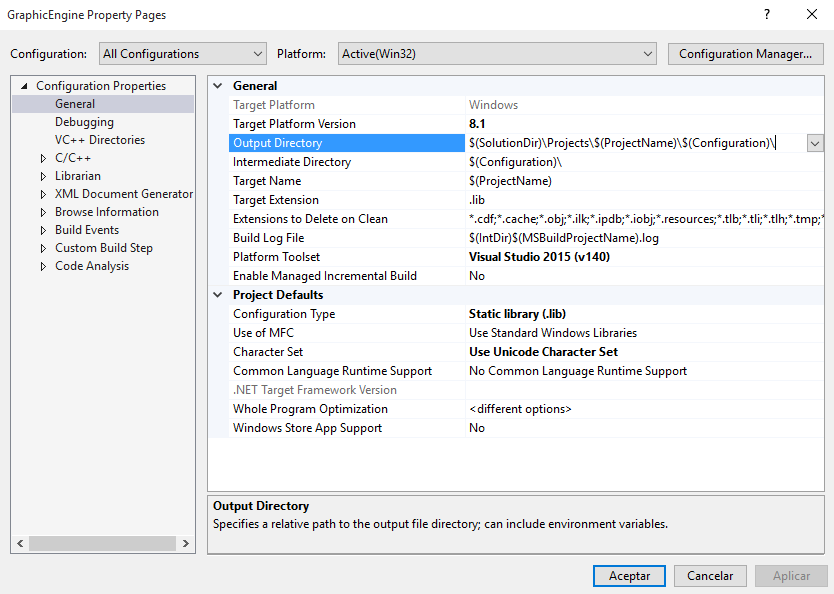
Por último, y no por ello menos importante, al abrir el visual studio, nos aparecerá que no encuentra el proyecto GraphicEngine (como antes con el Main), procedemos a hacer los mismos pasos, borrar el proyecto, incluir el proyecto ya existente, borrar los archivos GraphicEngine.h y cpp y por ultimo arrastraremos desde la carpeta **ArchitectureGuide/Src/GraphicEngine** estos dos archivos.

Volvamos al visual studio para cambiar donde van a crearse los ejecutables:

## Botón derecho sobre GraphicEngine -> properties

Ponemos el menú **Configuration en All Configurations** y en el campo **Output Directory** Ponemos la siguiente dirección:

### $(SolutionDir)\Projects\$(ProjectName)\$(Configuration)\

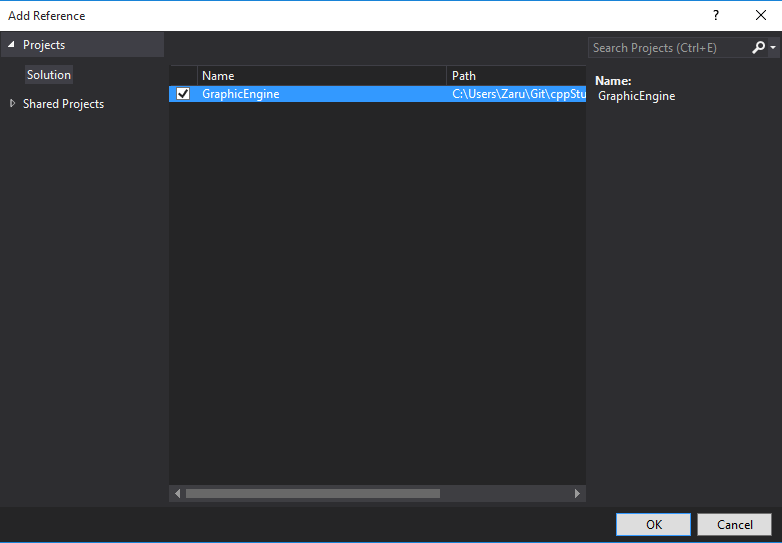
****

### Para que nuestro Main pueda utilizar

Y aplicamos y aceptamos, ahora si compilamos en modo debug o reléase vemos que funciona.

Ahora vamos a configurar el proyecto Main para que encuentre la biblioteca GraphicEngine y podamos utilizarla. Para ello agregamos una referencia de la siguiente manera:

En visual studio 2015 basta con que nos vallamos al exploración de soluciones y dar botón derecho sobre **References de Main -> Add Reference**, y clickear el Graphic Engine:

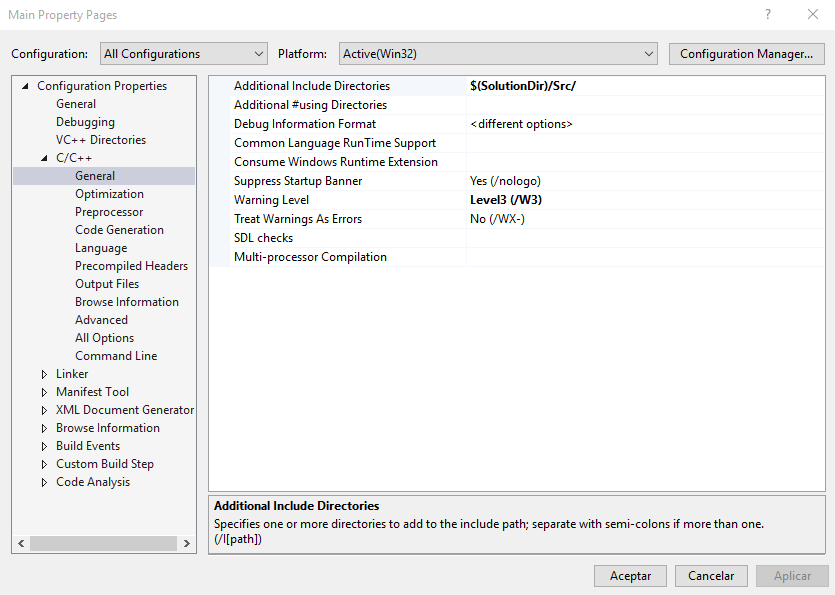


En 2013 lo tendremos en botón derecho sobre **Main -> Properties -> Common properties -> Add new reference** y clickear la casilla que marca.

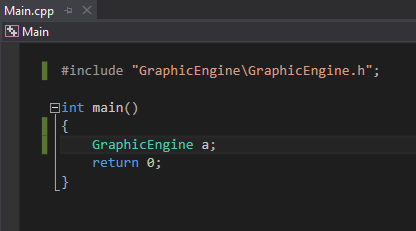
Por ultimo le diremos a c++ donde puede encontrar más bibliotecas de la siguiente manera:

## Botón derecho Main -> properties -> C++ -> General

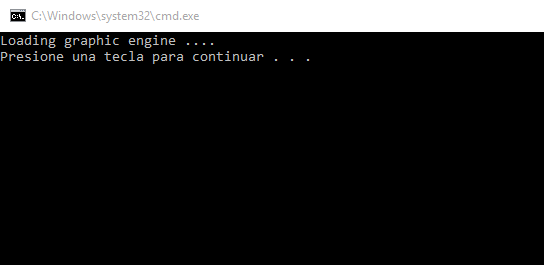
Y en el campo Additional Include Directories añadir nuestra carpeta Src de la siguiente manera: **$(SolutionDir)/Src/**



Con todo esto ya podremos ejecutar la biblioteca Graphic Engine de la siguiente manera:



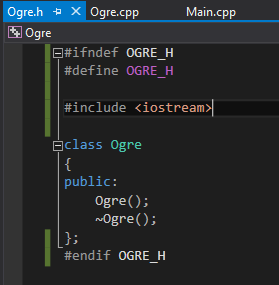
Y como podemos ver funciona perfectamente:



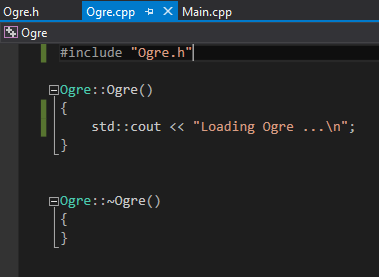
¿Y que pasa si queremos incluir una biblioteca que no estamos haciendo nosotros? Ahora veremos donde lo alojaremos.

Esta vez no lo pongo como lo hago por aquí, pero me creo un nuevo proyecto con el siguiente código, y el proyecto lo llamo Ogre. También lo creamos como biblioteca estática.

## Ogre.h

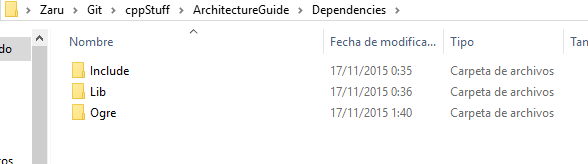


## Ogre.cpp

****

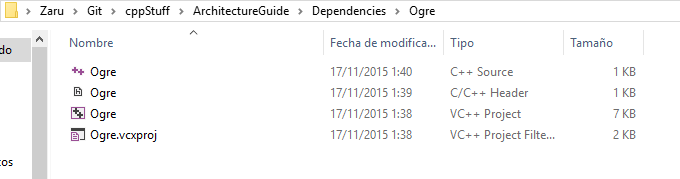
Tras esto vamos a organizar los archivos de la siguiente manera.

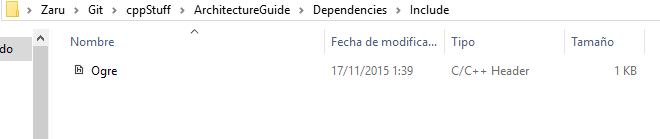
Como podemos ver se ha creado una nueva carpeta llamada **Ogre**, esta carpeta vamos a arrastrarla hasta **Dependencies** quedando esta carpeta de la siguiente manera:



Si no tenemos creadas las carpetas **Include** ni **Lib** las creamos.

Ahora copiaremos todos las cabeceras que haya en la carpeta Ogre (que en este caso solo hay una) y la pegamos en Include. Lo hacemos de esta manera para tener todas las bibliotecas externas organizadas.





Y esta es la organización de las bibliotecas externas.

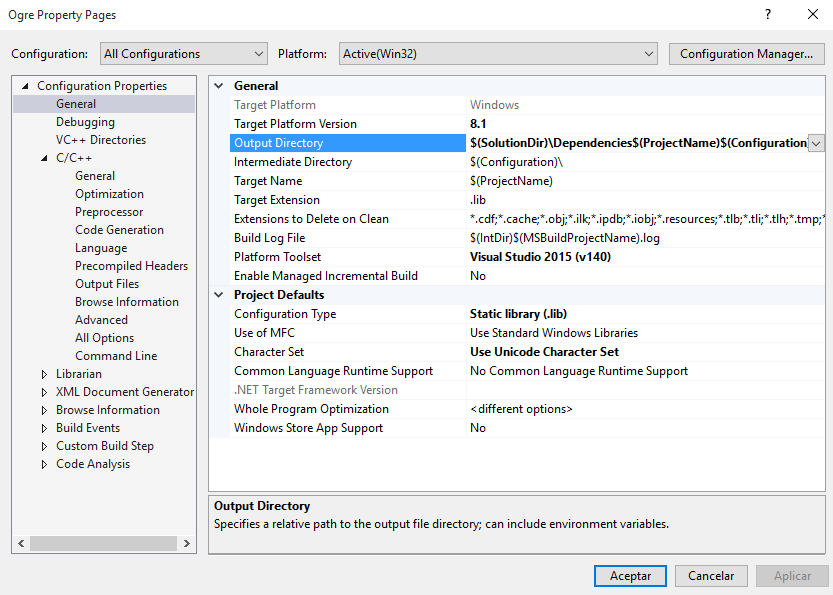
De nuevo, abrimos visual studio que nos dira que no encuentra el proyecto Ogre, hacemos lo de siempre, , borrar el proyecto, incluir el proyecto ya existente, borrar los archivos GraphicEngine.h y cpp y por ultimo arrastraremos desde la carpeta **ArchitectureGuide/Dependencies/include** el .h y desde **ArchitectureGuide/Dependencies/Ogre** el .cpp.

Configuramos donde se generan los ejecutables/Bibliotecas:

**Boton derecho Ogre -> properties -> Configuration Porperties -> General**

Y teniendo en **Configuration : All Configurations** rellenamos el campo OutPut Directory de la siguiente manera:

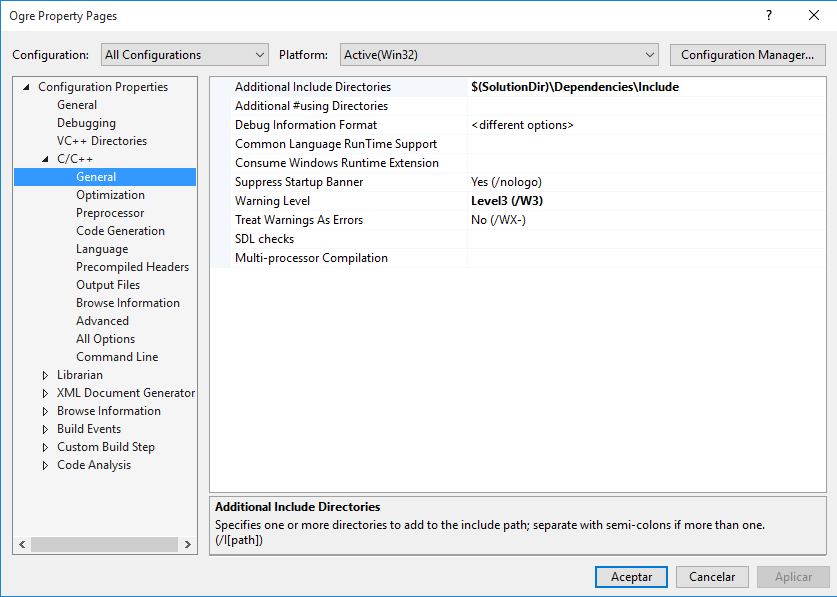
**$(SolutionDir)\Dependencies\Lib\$(Configuration)\**

****

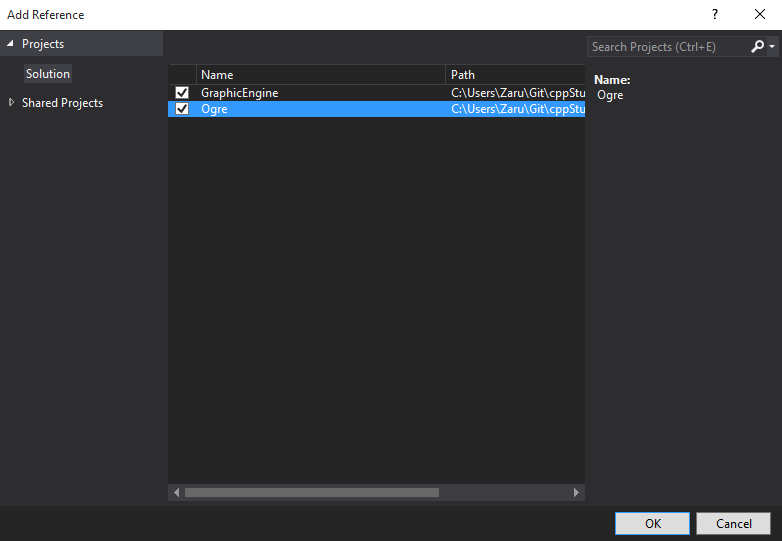
Para que visual studio pueda encontrar el .h hay que configurar los directorios de inclusión en el proyecto de Ogre:

**Boton derecho Ogre -> Properties -> c++ -> General**

Y rellenar el campo Additional Include Directories como sigue: **$(SolutionDir)\Dependencies\Include**

****

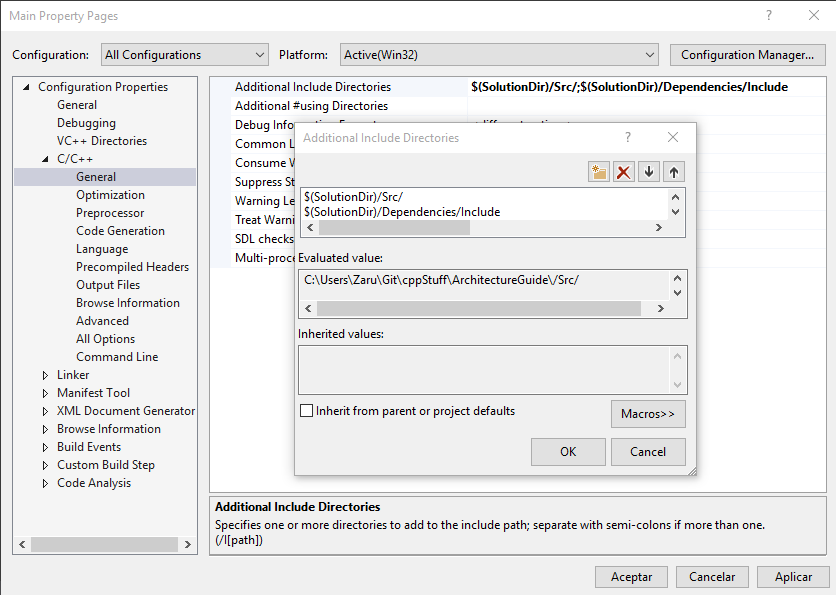
Ahora añadiremos una referencia a la biblioteca para poder utilizarla desde el Proyecto Main de la manera que ya se ha descrito más arriba.

****

También hay que decirle a c++ que directorio es en el que debe buscar la librería, que lo haremos de la siguiente manera:

**Botón derecho Main -> Properties -> c++ -> General**

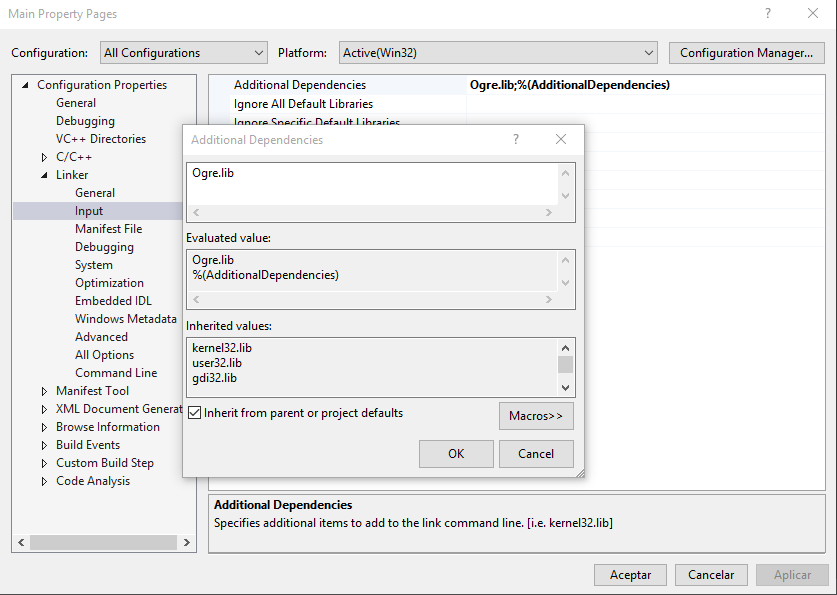
Y añadimos otra ruta más a **Additional Include Directories**



Por ultimo vamos a contarle al vinculador sobre las librerías a cargar.

**Botón derecho Main -> properties -> Linker -> Input**

En el campo **Additional Dependencies** añadir **Ogre.lib**



Y en **Botón derecho Main -> properties -> Linker -> General**

En el campo **Additional Libray Dependencies** rellenamos con **$(SolutionDir)\Dependencies\Lib\$(Configuration)** que es donde están las librerías.

