GeoLib Getting Started

Document version 1.0

www.geolib.co.uk
support@geolib.co.uk

Contents

Introduction 3

Setting up GeoLib after delivery 3

Building the GeoLib static library 3

Building the GeoLib dynamic library 3

Linking to the GeoLib static library 4

Linking to the GeoLib dynamic library 4

Incorporating the Drawing classes within your project 5

Notes on using GeoLib 5

Introduction

This document is a guide to building GeoLib from the source code and to linking to the static or dynamic libraries. The use of Visual Studio 2005 is assumed throughout although GeoLib can be built with other compilers, the only dependency being the Standard Template Library.

Setting up GeoLib after delivery

1. Create a folder on your C drive called "GeoLib" (an alternative location could be chosen but this would need to be reflected in the following instructions).
2. Open each of your delivered .zip files using WinZip and extract them to C:\GeoLib.

Your directory structure should be as follows (depending on what components you have been delivered).

- C:
    - GeoLib
        - GeoLib Source
        - GeoLib Static
            - Debug
            - Release
            - Headers
        - GeoLib Dynamic
            - Debug
            - Release
            - Headers
        - GeoDraw

Building the GeoLib static library

1. Open the GeoLib Source\GeoLib.vcproj file in Visual Studio 2005. This will be delivered to you configured to be built as a static library, but the following steps need to be taken to reset this configuration.
2. Right click on the project and select "Properties". Under "General", set the configuration to "Static Library(.lib)".
3. Under "C++" and "Preprocessor", ensure that "_STATIC" is defined. This switches off the DLL exporting defined at the start of each class header file.
4. Make sure the above steps are followed for both the "Release" and "Debug" configurations and select the configuration you require.
5. Select "Build" then "Build GeoLib". You can choose to save a solution or not at this stage.

Building the GeoLib dynamic library

1. Open the GeoLib Source\GeoLib.vcproj file in Visual Studio 2005.
2. Right click on the project and select "Properties". Under "General", set the configuration to "Dynamic Library(.dll)".

3. Under “C++” and “Preprocessor”, ensure that “_EXPORTING” is defined. This switches on the DLL exporting defined at the start of each class header file.
4. If building the Debug version, in the project properties, under “Linker” and “Input” and “Additional Dependencies” add “msvcrtd.lib”.
5. Make sure the above steps are followed for both the “Release” and “Debug” configurations and select the configuration you require.
6. Select “Build” then “Build GeoLib”. You can choose to save a solution or not at this stage.

Linking to the GeoLib static library

For the purposes of demonstrating how to link to GeoLib, a new project is created. GeoLib can just as easily be linked to your existing projects as well.

1. Open Visual Studio 2005.
2. Select “File” then “New” then “Project”.
3. Select “MFC Application” and call it “GeoLib Static Test”
4. In the Application Wizard select “Dialog Based” and “Use MFC in a shared DLL”. Deselect “Use Unicode libraries”.
5. Select “Finish”
6. In the project properties for the generated project, select “C++” then “General” and add an additional include directory as “C:\GeoLib\GeoLib Static\Headers” (quotation marks not required).
7. Under “Linker” and “General” add an additional library directory as “C:\GeoLib\GeoLib Static\Release” (quotation marks not required). Change to the “Debug” directory as required.
8. Under “Linker” and “Additional Dependencies” add “GeoLib.lib” (quotation marks not required).
9. Under “C++” and “Preprocessor” define “_STATIC”.
10. Open the file “GeoLib Static TestDlg.h” and add “#include “GeoLib.h””.
11. Add a member variable to the “CGeoLibStaticTestDlg” class. This can be a anything within GeoLib e.g. a “C2DRect”.
12. Build the project.

Linking to the GeoLib dynamic library

1. Open Visual Studio 2005.
2. Select “File” then “New” then “Project”.
3. Select “MFC Application” and call it “GeoLib DLL Test”
4. In the Application Wizard select “Dialog Based” and “Use MFC in a shared DLL”. Deselect “Use Unicode libraries”.
5. Select “Finish”
6. In the project properties for the generated project, select “C++” then “General” and add an additional include directory as “C:\GeoLib\GeoLib DLL\Headers” (quotation marks not required).
7. Under “Linker” and “General” add an additional library directory as “C:\GeoLib\GeoLib DLL\Release” (quotation marks not required). Change to the “Debug” directory as required.
8. Under “Linker” and “Additional Dependencies” add “GeoLib.lib” (quotation marks not required).

9. Open the file “GeoLib DLL TestDlg.h” and add “#include “GeoLib.h””.
10. Add a member variable to the “CGeoLibDLLTestDlg” class. This can be a anything within GeoLib e.g. a “C2DRect”.
11. Build the project.
12. To run the executable, ensure that the DLL is copied into the location of the executable.

Incorporating the Drawing classes within your project

GeoLib provides classes for drawing GeoLib shapes and these can be incorporated into you project as follows.

1. In your project settings, under “C++” and “General” and then “Additional Include Directories” add the directory path to the drawing classes – C:\GeoLib\GeoDraw.
2. In your project, add the header and source files in the GeoDraw directory. (e.g. right click on the Header files and select “Add” then “Existing Item…”.)

Notes on using GeoLib

All geometric entities within GeoLib have an associated set which enables the user to store them as an array. Objects are passed to the array either directly through the “Add” function as pointers or through the “AddCopy” function. The “Add” function means that the pointer passed is stored within the set and will be deleted when the set object goes out of scope. In other words the responsibility for deleting of the object is passed to the set object and this means the user should be careful not to delete an object that has been added to a set already. Similarly, objects passed directly to the set objects should be created on the heap e.g. through the “new” keyword. The “AddCopy” function will make a copy of the object passed and is therefore more robust to user error.

A similar approach is used for polygons with holes; holes and rims can be set directly in which case the pointer is passed to the polygon object and will be automatically deleted when that goes out of scope.