

Quick Start

MATFOR In C++

ANCAD INCORPORATED

TEL: +886(2) 8923-5411

FAX: +886(2) 2928-9364

support@ancad.com

www.ancad.com

Information in this instruction manual is subject to change without notice.

While AnCAD Incorporated makes every endeavor to ensure the accuracy of this document, it does not accept liability for any errors or omissions or for any consequences arising from the use of the program or documentation.

Quick Start: MATFOR Version 4.1

© Copyright AnCAD Incorporated 2007

All rights reserved.

All trademarks where used are acknowledged.

Contents

CONTENTS	3
CHAPTER 1 INTRODUCTION.....	4
I. PRODUCT DESCRIPTION.....	4
II. COMPILER REQUIREMENTS	5
III. INSTALLATION	5
IV. REGISTRATION.....	7
V. FLOATING LICENSE & CLASSROOM LICENSE	7
CHAPTER 2 PROJECT SETTINGS.....	10
<i>IN WINDOWS</i>	
I. BORLAND C++ BUILDER	10
II. MICROSOFT VISUAL C++	11
III. MICROSOFT VISUAL C++ .NET 2003 & 2005.....	14
IV. INTEL C++	17
<i>IN LINUX</i>	
V. ALL COMPILERS	17
TO RUN A DEMO CASE	18
CHAPTER 3 FUNDAMENTALS.....	19
I. MFARRAY	19
II. NUMERICAL PROCEDURES	20
III. VISUALIZATION PROCEDURES.....	22
CHAPTER 4 A FIRST PROGRAM.....	25
I. HELLO SURFACE	25
II. TO ANIMATE HELLO SURFACE	30
III. TO RECORD HELLO SURFACE	33
CHAPTER 5 THE ADVANCED FEATURES.....	35
I. MFPLAYER	35
II. MATFOR GUI BUILDER.....	36
III. MATFOR WIDGET	38

Chapter

1

INTRODUCTION

This Quick Start gives a brief overview of using the different components in MATFOR version 4.0. Language considerations, the structure of programs, and the use of advanced tools are all covered.

I. PRODUCT DESCRIPTION

MATFOR is a set of numerical and visualization libraries developed to enhance programming in C++ and Fortran environments. Especially designed for scientists and engineers, MATFOR aims to accurately perform computation, dynamically visualize data, and efficiently decrease process time. Its features include:

mfArray, a MATFOR defined data type that simplifies and facilitates object-oriented programming in C++ and Fortran environments.

Numerical Library contains useful linear algebraic functions subject to assist users with computational problem solving.

Visualization Library collects well-designed graphical procedures and controls to support a variety of 2D and 3D visual functions.

Data Viewer, organized in spreadsheet format, is one convenient platform for data management, filter, and analysis.

Graphics Viewer, besides its highly customized user interface, overthrows the convention of post-processing as it instantly visualizes scientific and engineering data.

mfPlayer animates simulation results by reading and displaying numerical data, and further allows additional graphical manipulations.

Graphics Export converts dynamic presentation into standalone movie or image files to enhance accessibility of simulation results.

GUI System facilitates application-building by allowing users to create an interface of their preference.

II. COMPILER REQUIREMENTS

MATFOR supports these compiler choices:

Platform and System Requirements		
Platform	Operating System	C++ Compilers
Intel Based 32-bit systems	Windows 98/NT/2000/Me/XP	<ul style="list-style-type: none"> ◆ Intel C/C++ 8.1/9.0/9.1 ◆ Visual C++ 6.0 ◆ Visual C++ .NET 2003/2005 ◆ Borland C++ Builder 6.0
Intel Based 32-bit systems	Fedora Core 1 / 2 / 3 / 4 Red Hat Enterprise Linux 3.0 / 4.0 White-Box Enterprise 3 SuSE 9.1 Enterprise Mandrake 10	<ul style="list-style-type: none"> ◆ GNU C++ ◆ Intel C++ 8.1/9.0/9.1
EM64T/AMD64 64-bit systems	Fedora Core 2 / 3 Red Hat Enterprise Linux 4.0	<ul style="list-style-type: none"> ◆ GNU C++ ◆ Intel C++ 8.1/9.0/9.1

Due to constant updates of the compilers, these requirements are subjected to change without further notice. Please refer to the latest version of compiler requirements at <http://www.ancad.com/requirements.html>.

III. INSTALLATION

This section provides step-by-step instructions for installing MATFOR. If you encounter any problems during the installation, please contact support@ancad.com.

WINDOWS INSTALLATION

- **Pre-Installation**

1. Exit any of MATFOR programs executing;
2. Remove all previous versions of MATFOR components if this is an upgrade;
3. Ensure the compiler requirements are satisfied;
4. Obtain administrator rights under Windows 98/2000/NT/XP.

- **Begin Installation**

1. Insert the MATFOR CD into the CD-ROM drive.
The standard MATFOR Installation Procedure shall start automatically. If the Procedure fails to start, you may manually start it by double-clicking on the MATFOR.exe file under the <CDROM>\Content\ path.
2. Follow the self-explanatory instructions in the Procedure to set up all MATFOR components.
3. Specify a destination folder to create MATFOR through the Procedure.
By default, the Package creates a program group in your C drive under the

path **C:\Program Files\AnCAD\MATFOR4**.

- **Set Environment Path**

The Procedure automatically sets the \$PATH and \$MATFOR4DIR environment variables.

To manually set the environment variables:

1. Go to **Control Panel\System**.
2. In the **System Dialog Box**, select the **Advance** label and click on the **Environment Variables** tab.
3. In the **System Variables Box**, add **<MATFOR4>\bin** and **<MATFOR4>\tools** to the [path] variable.

NOTE

<MATFOR4> shall specify the directory under which MATFOR is installed.

i.e. C:\Program Files\AnCAD\MATFOR4

LINUX INSTALLATION

- **Pre-Installation**

1. Exit any of MATFOR programs executing.
2. Remove all previous versions of MATFOR components if this is an upgrade.
3. Ensure the OS Requirements are satisfied.
4. Ensure the Compiler Requirements are satisfied.
5. Obtain root rights under Linux OS.
6. Run the '**rpm -ivh compat-libstdc++-33-3.2.3-47.3.i386.rpm**' command to install the compat-libstdc++-33 package. (Optional)

- **Begin Installation**

1. Open a command shell.
2. Change the current directory to the one containing the **matfor_c-4.xx-i686.rpm** Installer.
3. Run the '**rpm -ivh matfor_c-4.xx-i686.rpm**' command.
4. Follow the self-explanatory instructions in the Procedure to set up all MATFOR components. By default, the Installer creates a program group under the path **/usr/lib/matfor4**.

- **Environment Path Settings**

The Procedure automatically sets the \$PATH and \$MATFOR4DIR environment variables.

To manually set the environment variables:

1. Add the following lines to your `~/.bashrc` file.
2.

```
export MATFOR4DIR=/usr/lib/matfor4
export LD_LIBRARY_PATH=$MATFOR4DIR/lib:$LD_LIBRARY_PATH
export PATH=$MATFOR4DIR/tools:$PATH
```
3. Run the `'source ~/.bashrc'` command or re-login your system.

IV. REGISTRATION

At the end of installation, you are required to enter the License Password for registration, which can only be obtained by submitting the Host ID of your local machine to AnCAD, Inc.

1. Go to <http://www.ancad.com/activation.php> to activate MATFOR product with the **Serial Number** and **Host ID** information.
2. MATFOR license key should be sent to you through the email provided within 24 hours.

(Note: trial users may receive trial keys via email upon the receipt of download request. Please go to <http://webphp/download.php> to submit your request.)

UNDER WINDOWS

The registration program can be launched from **Start ► Program Files ► MATFOR4 ► Utilities ► Register MATFOR 4 in Fortran (or C++)**

UNDER LINUX

- Go to `/usr/lib/matfor4/tools`.
- Type `./reg_xxx` to launch the registration program.

V. FLOATING LICENSE & CLASSROOM LICENSE

To enable a group of developers to share a pool of licenses more efficiently, MATFOR offers floating licenses and classroom licenses for cross-system/cross-platform environments. The floating license is designed to be used in any shared network environment; the classroom license is designed for the purpose of teaching in an academic environment. Any project developed under MATFOR classroom license may not be redistributed to a third party with profit interest or being a commercial institution.

These license models consist of one or more license servers; the license server runs the license management process and monitors number of concurrent users using MATFOR licenses in the network. The installation steps are:

WINDOWS INSTALLATION

- **Installation of License Servers**

1. Run **license_server.exe** (download from <http://www.ancad.com/activation.php>) on machines designated as license servers.
2. Retrieve required information from the machine.
Under command prompt,
 - i. Change directory to **C:\Program Files\AnCAD\License Server**.
 - ii. Execute **Imtools** to get Ethernet Address and Host Name.
3. Collect the following product information:
 - i. Compiler version
 - ii. Operating system
 - iii. Number of license of each version
4. Obtain license file(s) for license server(s).
 - i. Go to <http://www.ancad.com/activation.php> to activate MATFOR with the **Serial Number** and **Host ID** information.
 - ii. The license file(s) should be sent to you through the email provided within 24 hours.
5. Place the license file **xxx.lic** into **<AnCAD>License Server** directory.
6. To start the license server.
Under command prompt, type:
Imgrd -c xxx.lic or
Imgrd -c . for multiple license files.

- **Configuration in Client Computers**

1. Install MATFOR on machines designated as clients.
2. Go to **Start ► Program Files ► MATFOR4 ► Utilities ► Register MATFOR**.
3. Under Registration Window, choose Network License.
4. Fill in the Host Name or IP Address of the license server in the blank area. (IP Address can be obtained from the license server by typing **Imhostid -internet -n** under command prompt.)

LINUX INSTALLATION

- **Installation of License Servers**

1. Run license server executable file (download from <http://www.ancad.com/activation.php>) on machines designated as license servers.
tar xzf license_server.tar.gz -c /usr/lib
2. Retrieve required information from the machine.
Under command shell,
 - i. Go to ***/usr/lib/matfor_licsvr***
 - ii. Type ***./lmhostid -n*** to get Ethernet Address.
 - iii. Type ***./lmhostid -hostname -n*** to get Host Name.
3. Collect the following product information:
 - i. Compiler version
 - ii. Operating system
 - iii. Number of license of each version
4. Obtain license file(s) for license server(s).
 - i. Go to <http://www.ancad.com/activation.php> to activate MATFOR with the **Serial Number** and **Host ID** information.
 - ii. The license file(s) should be sent to you through the email provided within 24 hours.
5. Place the license file **xxx.lic** into ***/usr/lib/matfor_licsvr*** directory.
6. Start the license server.
Under command prompt, type:
lmgrd -c xxx.lic or
lmgrd -c . for multiple license files.

- **Configuration in Client Computers**

1. Install MATFOR on machines designated as clients.
2. Go to ***/usr/lib/matfor4*** and type ***./reg_xxx***
3. Under Registration Window, choose Network License.
4. Fill in the Host Name or IP Address of the license server in the blank area. (IP Address can be obtained from the license server by typing ***lmhostid -internet -n*** under command shell.)

For other questions regarding MATFOR floating license or classroom license installation and registration, please contact support@ancad.com.

Chapter

2

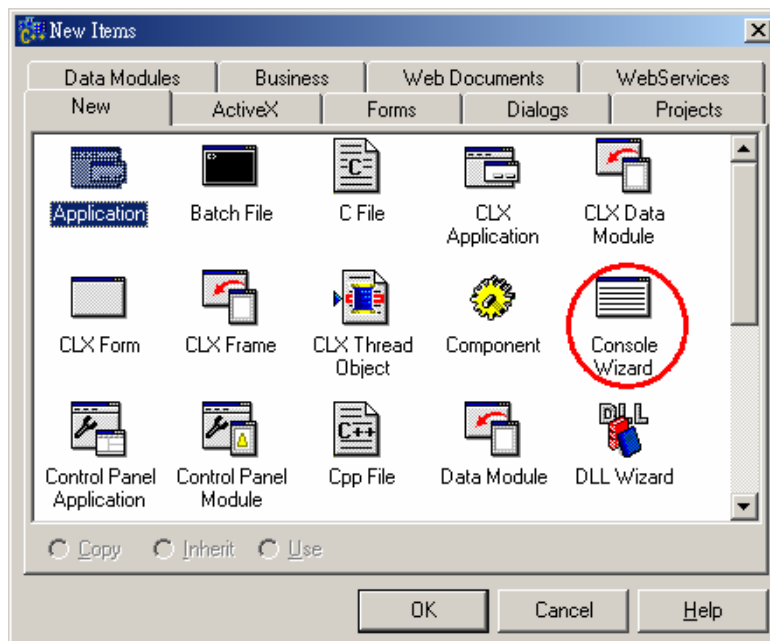
PROJECT SETTINGS

Due to the variety in MATFOR-supporting compilers, this chapter is composed to guide users to configure project settings under different compiler environments.

IN WINDOWS

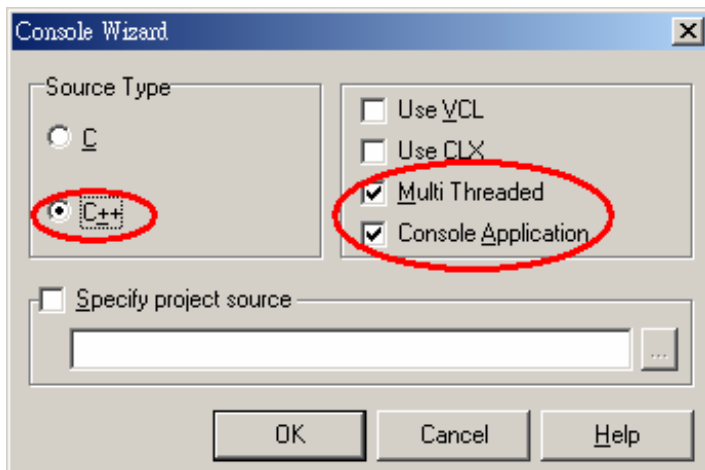
I. BORLAND C++ BUILDER

1. Open Borland C++ Builder 6.0.
2. Select **File ► New ► Other**.
3. Click on the **New** tab, select **Console Wizard**, and click **OK**.

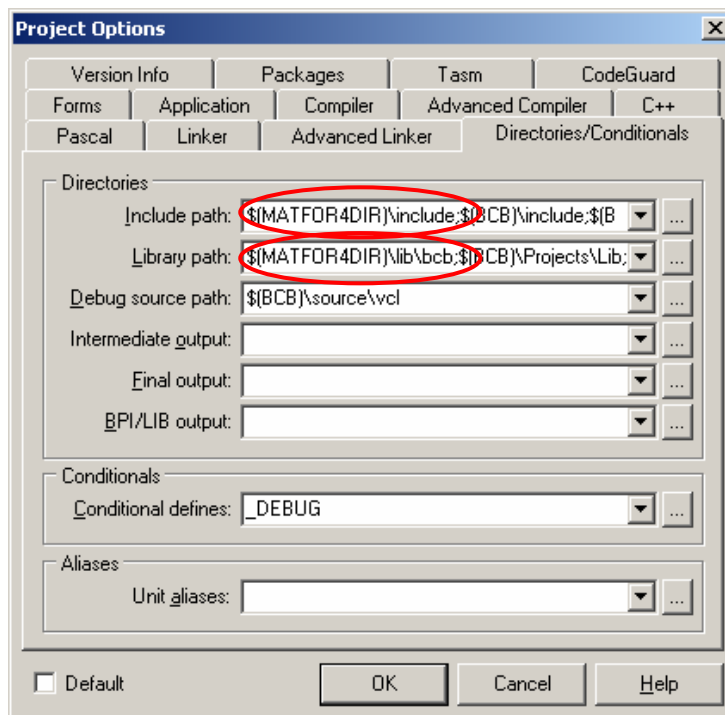


4. Select **C++** and check **Multi Threaded** and **Console Application**.

- Click **OK**.



- Select **Options** from the **Project** menu.
- Select the **Directories/Conditionals** tab.
- Under **Include path**, type in: `$(MATFOR4DIR)\include;`
- Under **Library path**, type in: `$(MATFOR4DIR)\lib\bcb;`



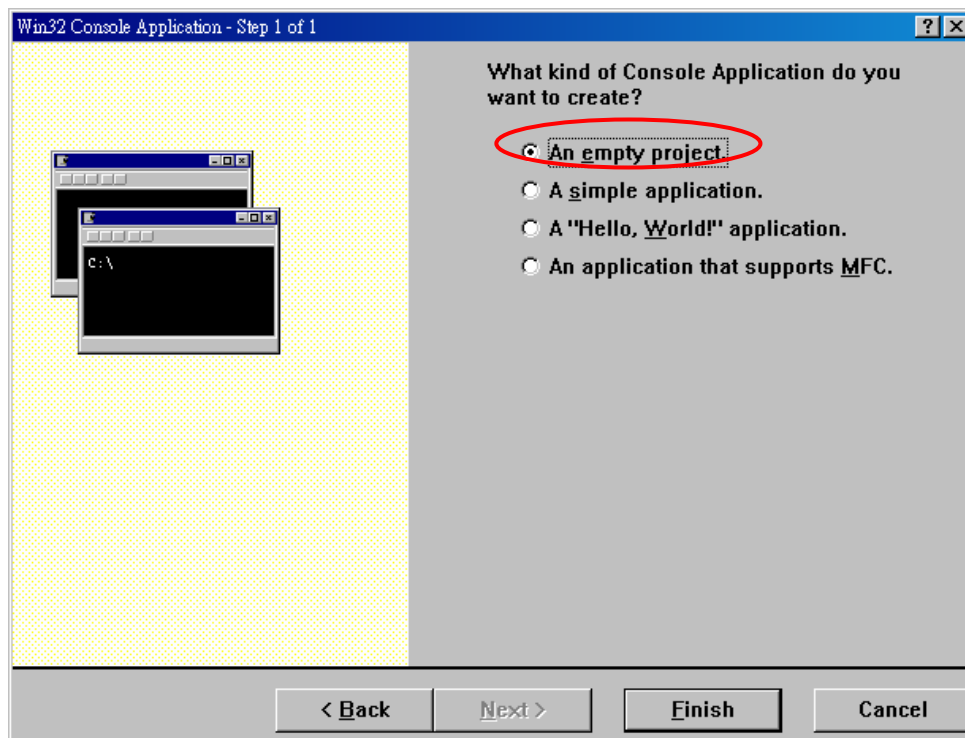
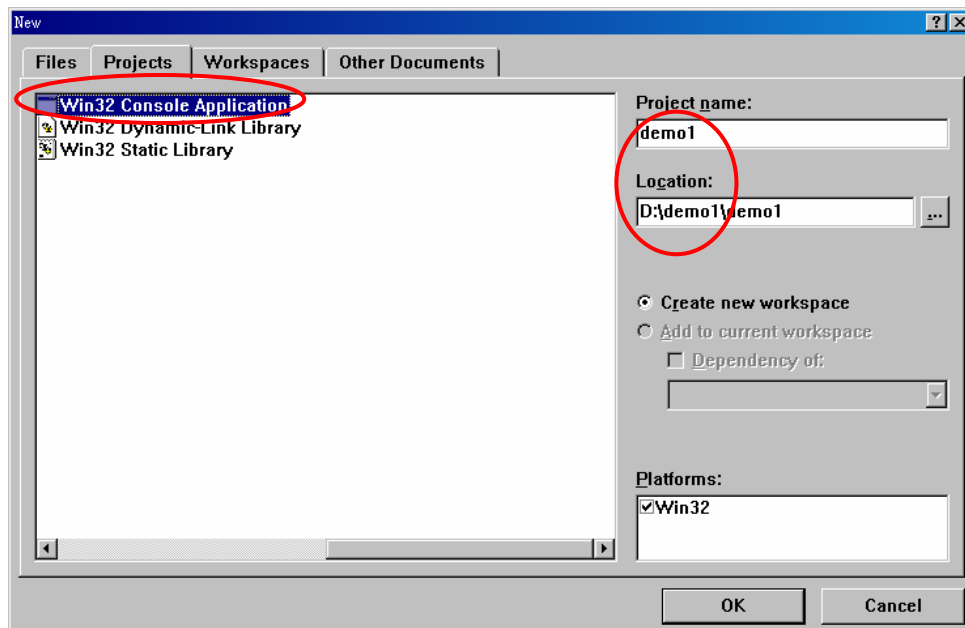
- Add the following code to the header of the program(s):

```
#pragma link "fml.lib"
#pragma link "fgl.lib"
```

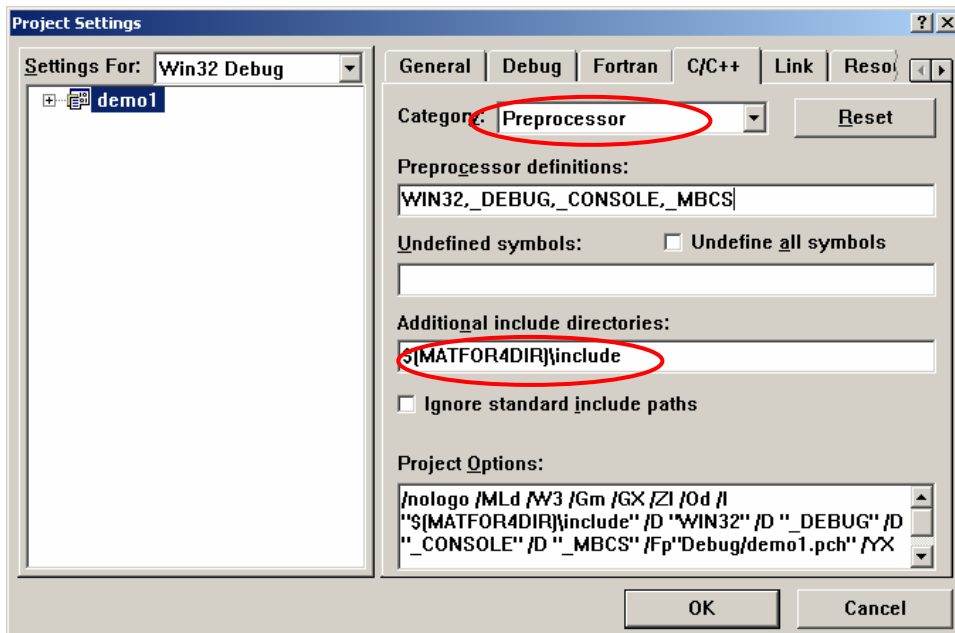
II. MICROSOFT VISUAL C++

- Open Microsoft Visual C++ 6.0.
- Go to **File ► New**.

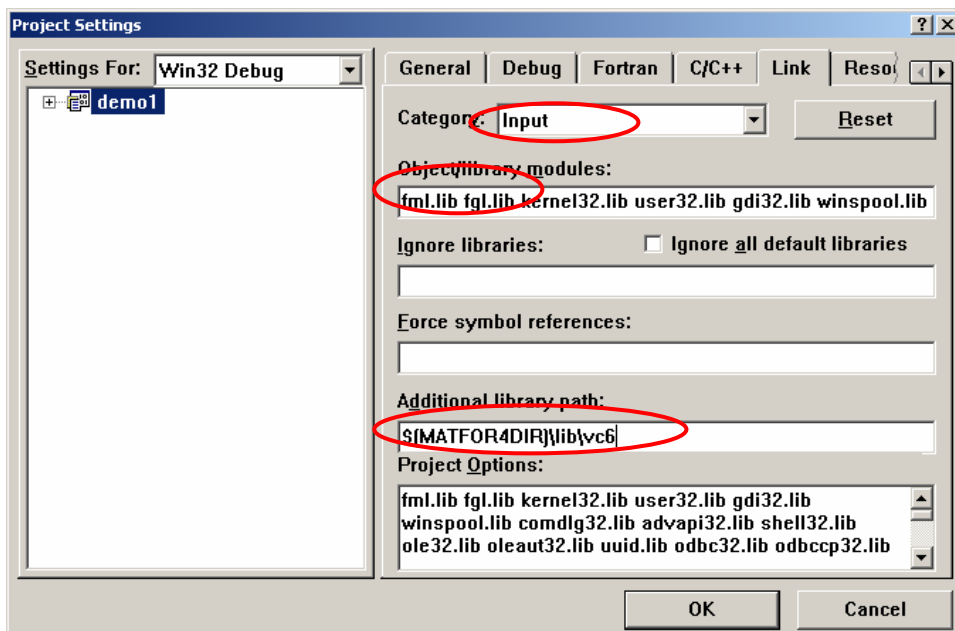
3. Select the project type **Win 32 Console Application**.
4. Enter a name for the new project, specify its location, and click **OK**.
5. Select **An empty project** and click **Finish**.



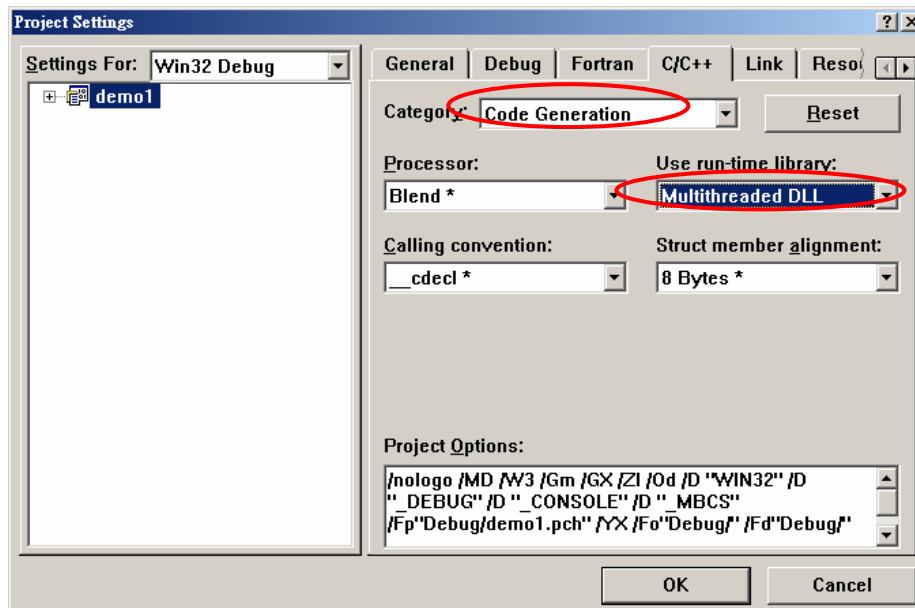
6. Select **Settings** from the Project menu.
7. Click on the **C/C++** label and select **Preprocessor** from the **Category** menu.
8. Under Additional include directories, type in: **\$(MATFOR4DIR)\include**.



9. Click on the Link label and select **Input** from the **Category** menu.
10. Under Object/Library Modules, add: **fml.lib fgl.lib** (and **spml.lib** if using MATFOR Sparse Array).
11. Under Additional library directories, type in: **\$(MATFOR4DIR)\lib\vc6**

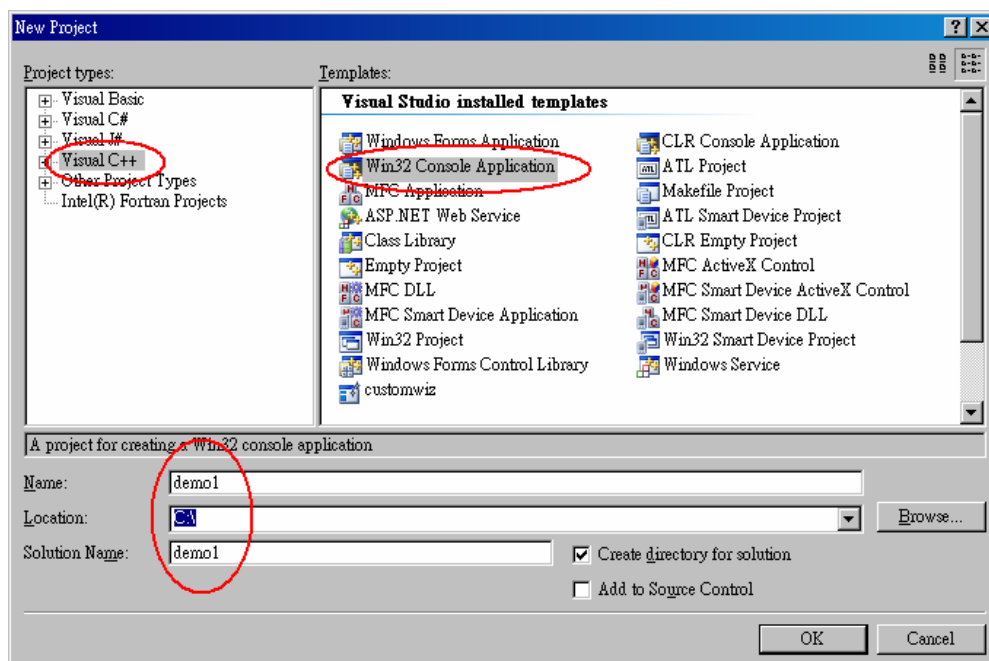


12. Click on the C/C++ and select **Code Generation** from the Category menu.
13. Under Use Run-time Library, choose **Multithreaded DLL**.



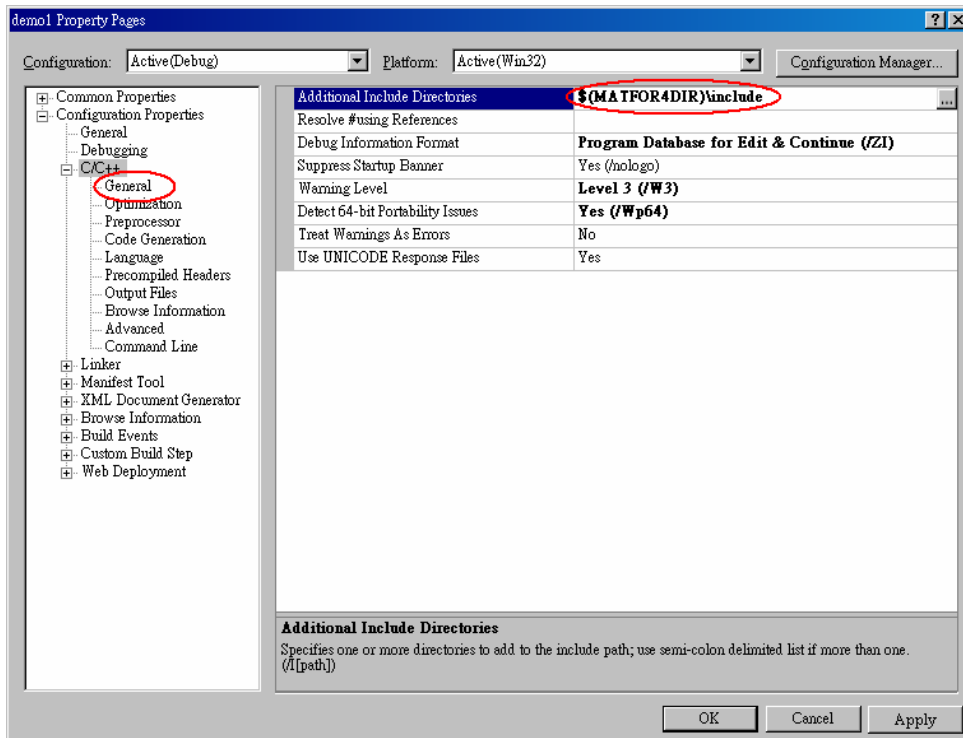
III. MICROSOFT VISUAL C++.NET 2003 & 2005

1. Open Microsoft Visual Studio 2005.
2. Go to **File ► New Project**.
3. Select **Visual C++ Projects** from the **Project Types** menu.
4. Select **Win32 Console Application** from **Templates** and create a solution name. (Under Visual Studio 2003, please select **Win32 Console Project**).
5. Enter a name for the new file and click **OK**.

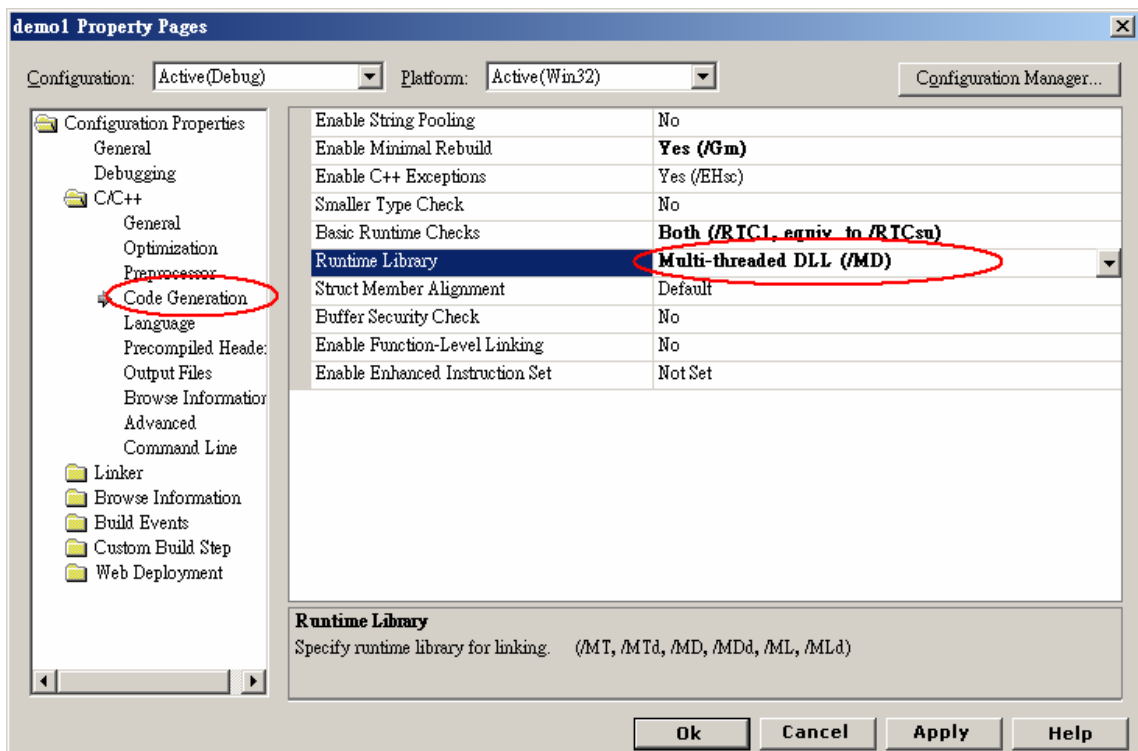


6. Click on **Finish**.
7. Select **Properties** from the **Project** menu.

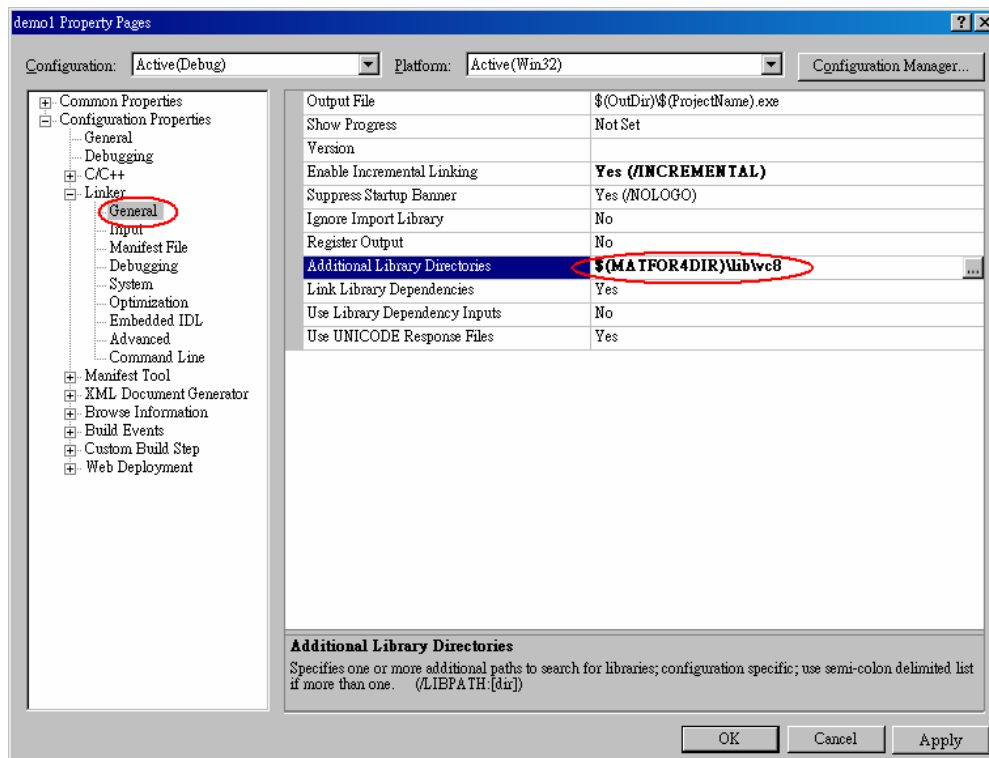
8. Go to **General** under the **C/C++** folder.
9. In **Additional Include Directories**, type in: **\$(MATFOR4DIR)\include**



10. Go to **Code Generation**, in **Runtime Library**, select: **Multi-threaded DLL (/MD)**

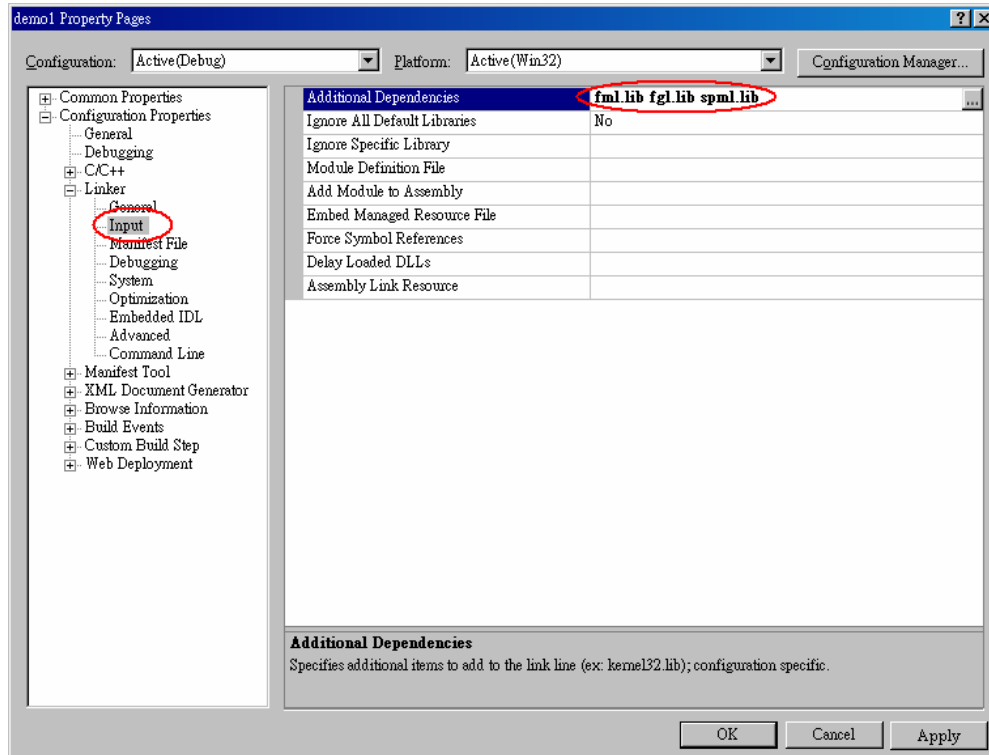


11. Go to **General** under the **Linker** folder.
12. In **Additional Library Directories**, type in: **\$(MATFOR4DIR)\lib\vc8**
(For users of Visual Studio 2003, please type in: **\$(MATFOR4DIR)\lib\vc7**)



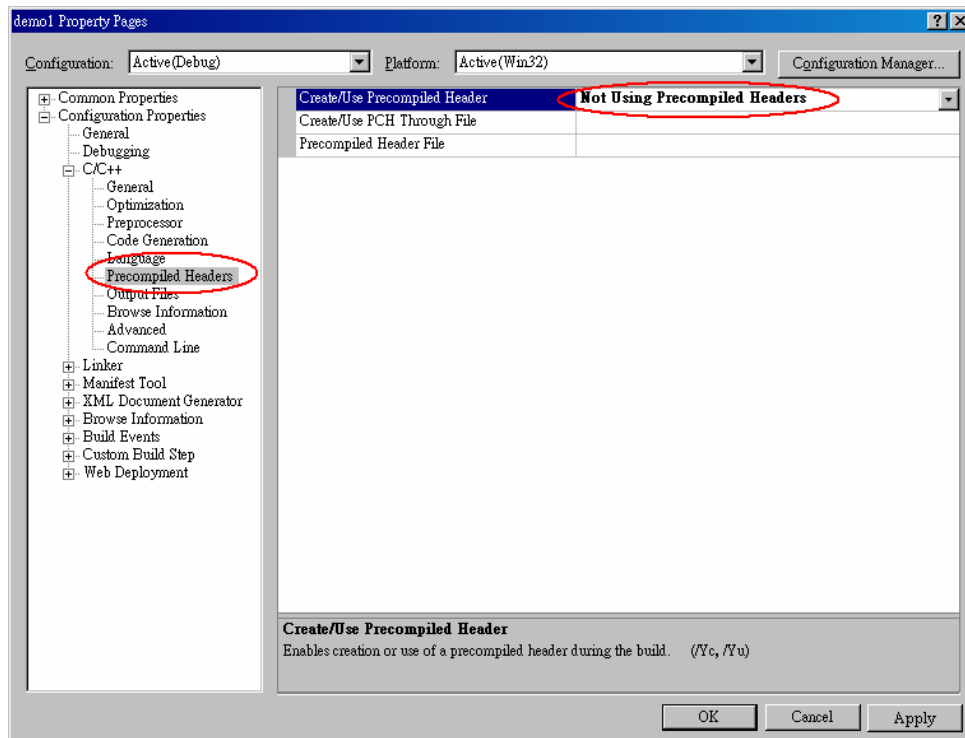
13. Go to **Input** under the **Linker** folder.

14. In **Additional Dependencies**, add: **fml.lib fgl.lib** (and **spml.lib** if using MATFOR Sparse Array).



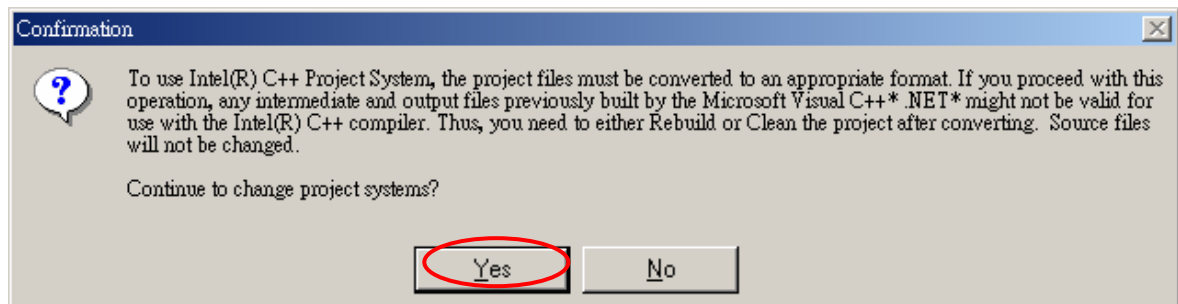
15. Go to **Precompiled Headers** under the **C/C++** folder.

16. In **Creative/Use Precompiled Header**, select **Not Using Precompiled Header**.



IV. INTEL C++

1. Repeat **Step 1** through **6** in previous section (III. MICROSOFT VISUAL C++ .NET 2003 & 2005)
2. Select **Convert to use Intel® C++ Project System** from the **Project** menu.
3. Click **Yes**.



4. Repeat **Step 7** through **16** in previous section

IN LINUX

V. ALL COMPILERS

1. Open Command Prompt.
2. Copy **linux_xxx_template.mak** from **/usr/lib/matfor4/tools/template** to the directory of the project.

3. Rename the make file to **linux_xxx.mak**.
4. Modify **linux_xxx.mak** as required.
5. Type **make -f linux_xxx.mak** to build and compile the project.

TO RUN A DEMO CASE

UNDER WINDOWS

1. Go to the directory where MATFOR 4 is installed.
2. Open **demo/ cpp/<demo case>**.
3. Select the **.dsw** file to run the demo case.

UNDER LINUX

1. To compile demo cases all at once, go to **/usr/lib/matfor4/demo/cpp**
To compile each demo case individually, go to **usr/lib/matfor4/demo/cpp/<demo case>**.
2. Type **make -f linux_gcc.mak** in GNU C++ environment, or
type **make -f linux_icc.mak** in Intel C++ environment
3. Go to the **/usr/lib/matfor4/demo/cpp/<demo case>** to run each demo case.
For example,
cd \$MATFOR4DIR/demo/cpp/3dplot
./3dPlot

Chapter

3

FUNDAMENTALS

This chapter is an introductory summary on how to integrate MATFOR into C++ programming. Beginning with `mfArray`, the core of MATFOR, the chapter subsequently introduces the numerical library and the visualization library.

I. MFARRAY

`mfArray` is a dynamic array endorsing the data types and dimensions listed in Table 3.1.

Data Type	Statement
<code>const char *</code>	<code>a = "string"</code>
<code>bool</code>	<code>a = true</code>
<code>Int</code>	<code>a = 1</code>
<code>float</code>	<code>a = 1.0f</code>
<code>double</code>	<code>a = 3.1418f</code>
<code>dcomplex</code>	<code>a = dcomplex (2.0f, 3.0f)</code>

Table 3.1 | `mfArray` Data Types

`mfArray` Declaration

1. Include MATFOR library to appropriately contain `mfArray`.

2. Declare mfArray using the format *mfArray* <variable>

Example Code (ch3-1)


```
#include "fml.h"           // Include the library file fml.h

void main()
{
    mfArray x, y;           // Declare mfArray variables x and y

    x = 6.0;                // x is a scalar
    y = mfV(1.0, 2.0, 3.0, 4.0, 5.0); // y is a 1-by-five vector containing
    real values

    mfDisplay(x, "x", y, "y"); // Call mfDisplay function to display x
    and y data
}
```

Result



```
x =
6
y =
1 2 3 4 5
```

II. NUMERICAL PROCEDURES

MATFOR embraces a comprehensive set of numerical procedures furnished with easy-to-call syntax to complement mfArray. Various categories of numerical procedures are contained in the set and are hereto listed in Table 3.2.

For a complete list of the numerical routines and descriptions on their usage and functionality, please refer to the Reference Guide.

Procedure Group	Example	Description
Data Manipulations	mfSort	Sort data in ascending order
Arithmetic Operators	mfRDiv	Right-divide the matrix
Trigonometry	mfCos	Request cosine function
Exponential	mfLog	Request natural logarithm
Complex	mfConj	Generate conjugate of complex
Rounding and Remainder	mfCeil	Round towards positive infinity
Matrices	mfMagic	Construct magic square
Matrix Manipulations	mfFind	Find indices of nonzero elements
Matrix Analysis	mfNorm	Generate matrix/vector norm
Linear Equations	mfChol	Request Cholesky factorization
Eigenvalues and Singular Values	mfSchur	Perform Schur decomposition
Factorization Utilities	mfBalance	Perform diagonal scaling

Table 3.2 | Numerical Procedure Brief

Numerical Procedure Call

1. Include “fml.h” to appropriately contain the numerical routine(s).
2. To retrieve the procedure,
 - Use the assignment operator “=”, or
 - Simply call.

Example Code (ch3-2)

```

#include “fml.h”           // Include the library file fml.h
void main()
{
    mfArray m, i, j;       // Declare mfArray variables m, i, and j
    m = mfMagic(3);        // Construct a 3-by-3 magic matrix and assign it
                           // to m
    mfFind(mfOut(i, j), m); // Retrieve the row-column indices of the

```

```

nonzero(s) in m
// i and j describe the row and column,
respectively
mfDisplay(m); // Display matrix m
mfDisplay( i, "i", j, "j"); // Call mfDisplay function to display x and y
data
}

```

Result

```

Ans =
  8  1  6
  3  5  7
  4  9  2

i =
  1  2  3  1  2  3  1  2  3

j =
  1  1  1  2  2  2  3  3  3

```

III. VISUALIZATION PROCEDURES

In addition to the numerical procedures, MATFOR endorses a collection of visualization routines to enhance data comprehension. Table 3.3 summarizes the procedures and introduces an example routine with description for each procedure group.

Procedure Group	Example	Description
Window Management	mfWindowSize	Set the frame size of Graphics Viewer window
Visualization Controls	mfSubplot	Create subplot in active figure
Graph Options	mfSurf	Construct 3D surface plot
Advanced Graph Options	mfGetDelaunay3	Construct 3D Delaunay triangulation
Object Manipulations	mfObjOrigin	Set the origin of the drawn object
Appearance Settings	mfColormap	Specify the colormap type of the drawn object
Annotations	mfTitle	Annotate the graph with title
3D Objects	mfSphere	Draw a sphere

Table 3.3 | Visualization Procedure Brief

For a complete list of the visualization routines and descriptions on their usage and functionality, please refer to the Reference Guide.

Visualization Procedure Call

1. Include “fgl.h” to appropriately contain the visualization routine(s).
2. To retrieve the procedure,
 - Use the assignment operator “=”, or
 - Simply call.

Example Code (ch3-3)

```
#include "fml.h"           // Include the library file fml.h
#include "fgl.h"           // Include the library file fgl.h

void main()
{
    mfArray nx, ny, nz;
    mfArray x, y, z, c, tet;    // Declare mfArray variables

    nx = mfLinspace(-2, 2.2, 21);    // Construct linearly spaced vector nx
    ny = mfLinspace(-2, 2.25, 17);    // Construct linearly spaced vector ny
    nz = mfLinspace(-1.5, 1.6, 31);    // Construct linearly spaced vector nz

    mfMeshgrid(mfOut(y, x, z), ny, nx, nz); // Construct grid matrices for
    3D plotting

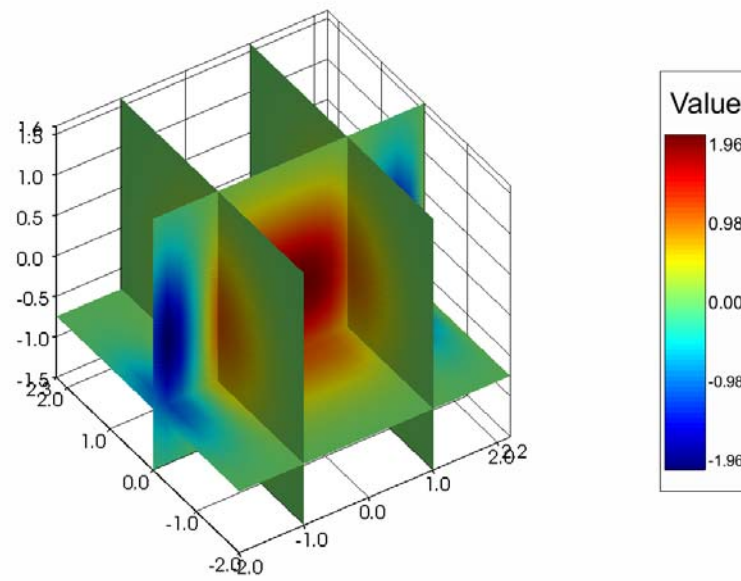
    c = 2 * mfCos(x*x) * mfExp( - (y*y) - (z*z)); // Mathematically define c

    tet = mfGetDelaunay3(x, y, z);    // Construct 3D Delaunay triangulation

    mfTetSliceXYZ( tet, x, y, z, c,
        mfV(-1.0, 1.0), 0, -0.75 );    // Display orthogonal sliced planes

    mfViewPause();                // Pause to display the graph
}
```

Result



Chapter

4

A FIRST PROGRAM

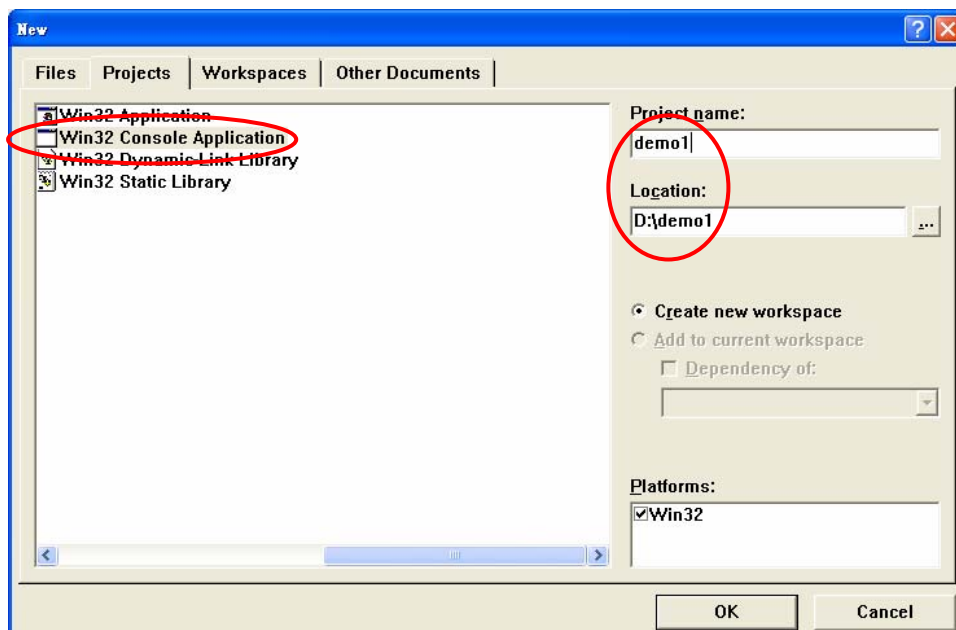
This chapter presents one simple visualization program for users to manipulate and extends as animation and recording procedures are introduced.

I. HELLO SURFACE

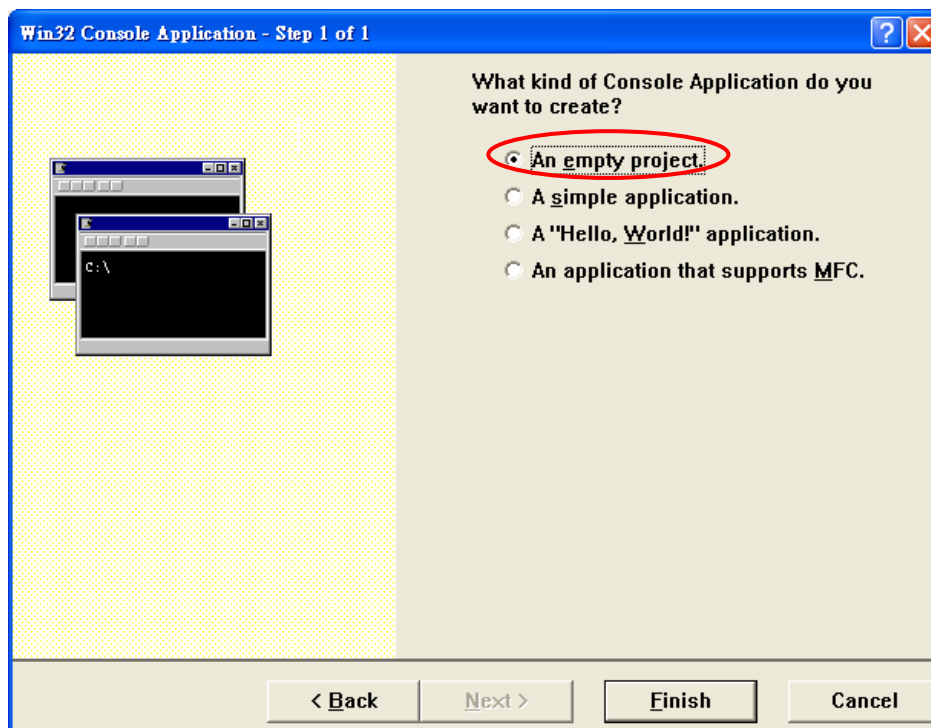
The section shall take you step by step to create a surface plot.

Create a New Project

1. Select the project type **Win 32 Console Application**.
2. Name the project “**demo1**,” specify its location, and click **OK**.

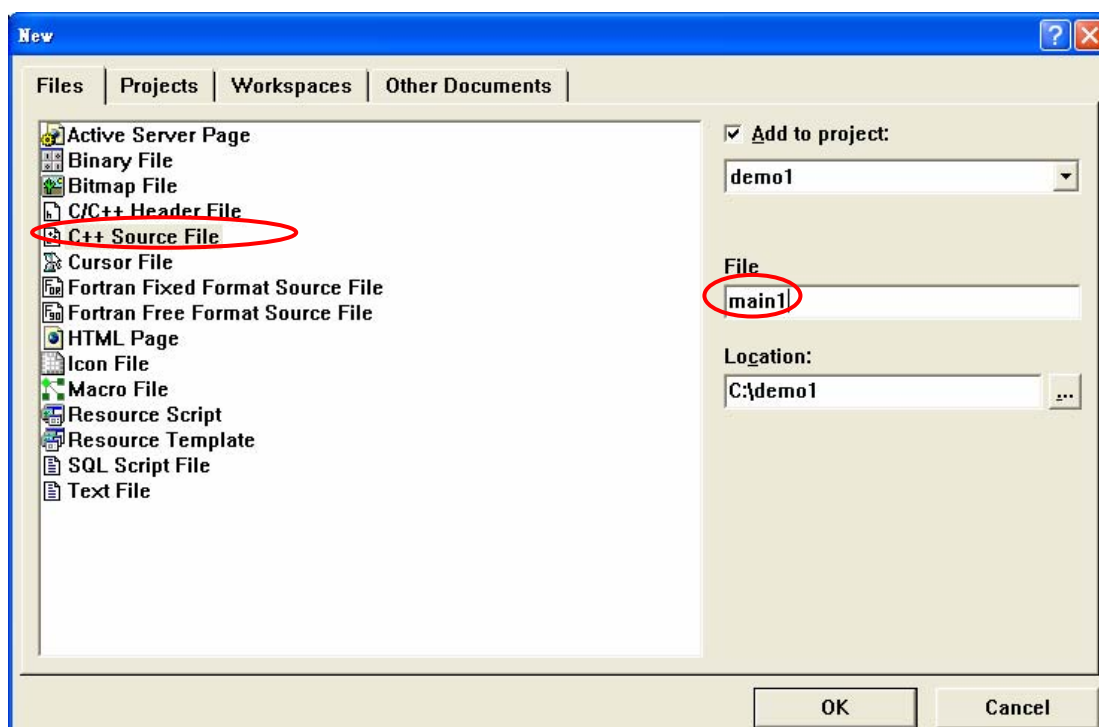


3. Select **An empty project** and click **Finish**



4. Select the file type **C++ Source File**.

5. Name the file "main1," specify its location, and click **OK**.



Adding the Source Code (ch4-1)

In “main1.cpp,” type in:

```
#include "fml.h"           // Include the library file fml.h
#include "fgl.h"           // Include the library file fgl.h

int main(int argc, char* argv[])
{
    mfArray x, y, z;       // Declare mfArray variables

    mfMeshgrid( mfOut(x, y), // Construct grid matrices for 3D
plotting
                mfLinspace(-3, 3, 30),
                mfLinspace(-3, 3, 30) );

    z = mfSin(x) * mfCos(y); // Mathematically define z

    mfSurf(x, y, z);       // Plot a surf using mfArray x, y,
and z
    mfViewPause();        // Pause to display the graph
    return 0;
}
```

Line-by-Line Walkthrough

#include “fml.h”

This includes the library file fml.h which contains the MATFOR numerical procedures.

#include “fgl.h”

This includes the library file fgl.h which contains the MATFOR visualization procedures.

mfArray x, y, z;

This declares mfArray variables x, y, and z.

mfMeshgrid(mfOut(x, y), mfLinspace(-3, 3, 30), mfLinspace(-3, 3, 30));

This creates x and y grid matrices from the domains specified by mfLinspace-generated vectors.

- mfOut(x, y) specifies x and y as mfArray outputs.
- mfLinspace(-3, 3, 30) constructs a vector with 30 linearly spaced points between -3 and 3.

```
z = mfSin(x) * mfCos(y);
```

This defines z mathematically.

```
mfSurf( x, y, z );
```

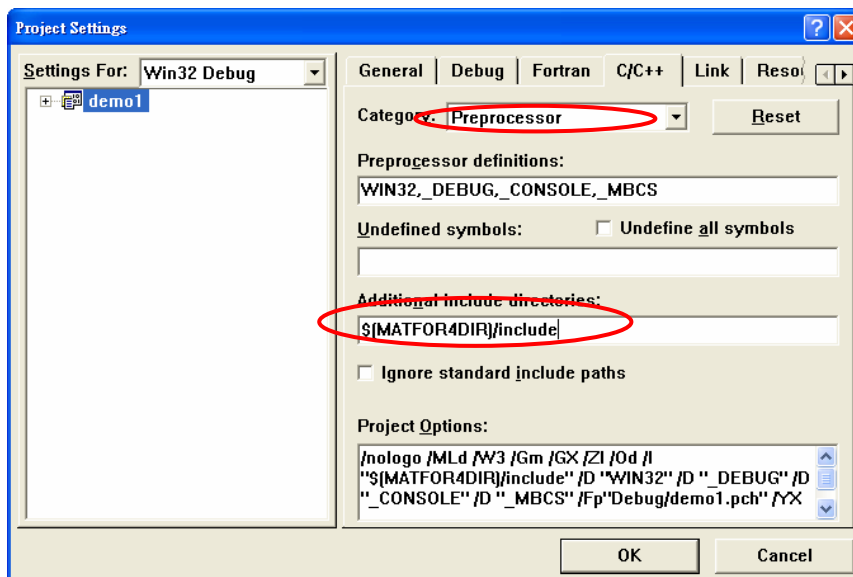
This creates a three-dimensional surface plot of the x-, y-, and z-coordinates.

```
mfViewPause();
```

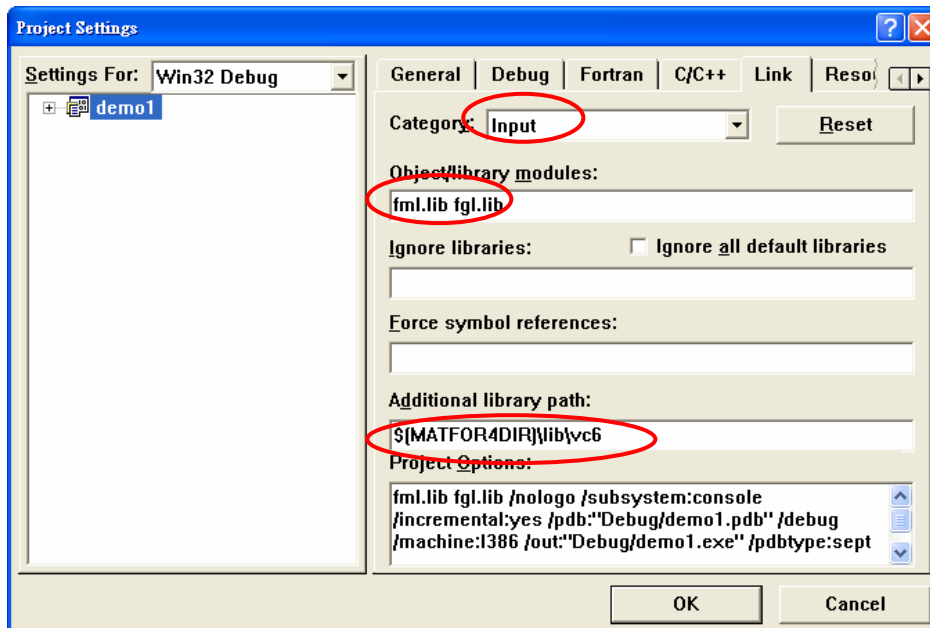
This halts program execution for graphical display. `mfViewPause()` should be added after each set of graphical creation routines.

Adding Library and Include paths:

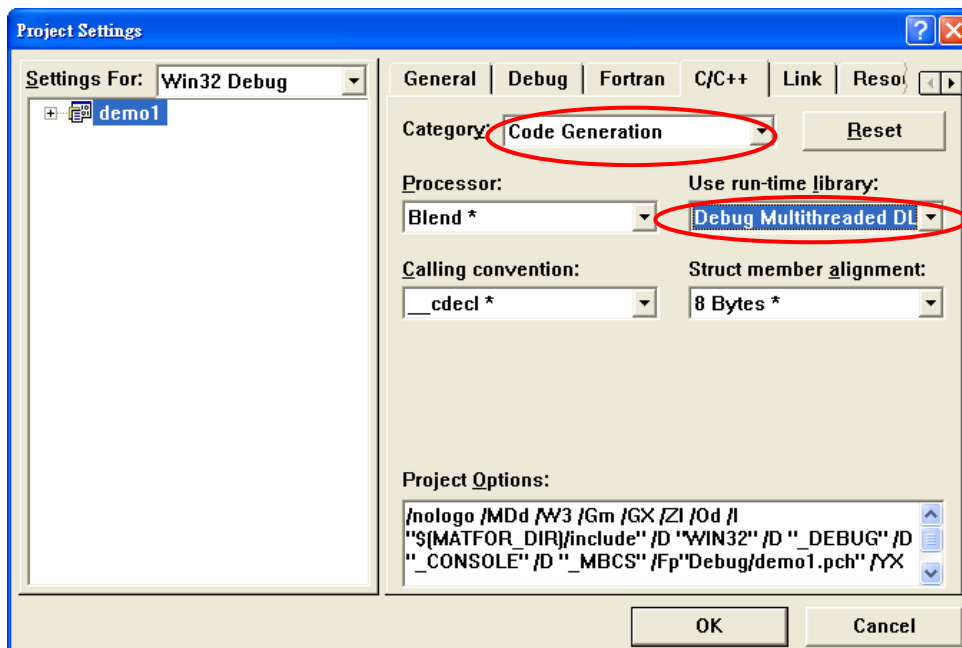
1. Select **Settings** from the Project menu.
2. Click on the C/C++ label and select **Preprocessor** from the Category pull-down menu.
3. Under Additional include directories, type in **\$(MATFOR4DIR)/include**.



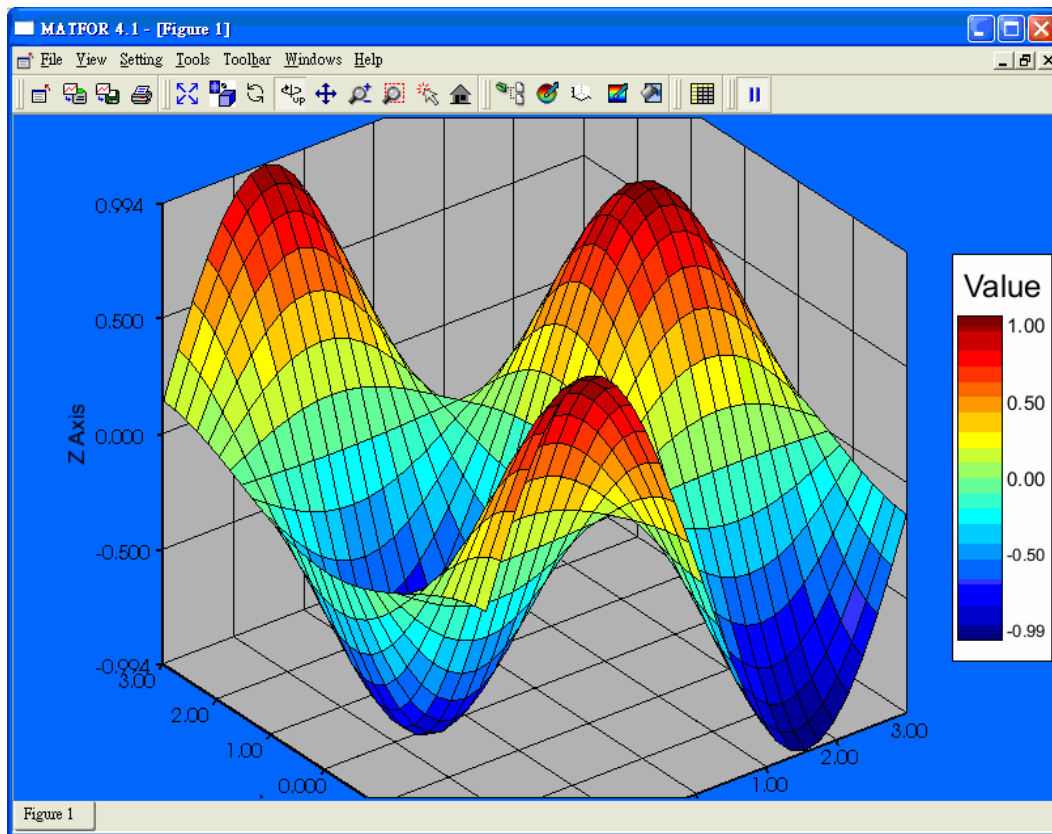
4. Click on the Link label and select **Input** from the Category pull-down menu.
5. Under Object/Library Modules, add **fml.lib** and **fgl.lib**.
6. Under Additional library directories, type in **\$(MATFOR4DIR)/lib/vc6**.



7. Click on the C/C++ and select **Code Generation** from the Category pull-down menu.
8. Under Use Run-time Library, choose **Debug Multithreaded DLL**.



Compile and Run



II. TO ANIMATE HELLO SURFACE

In this section, the Hello Surface demo code is modified to generate animation.

Animation

Animations effects are produced by continuously updating data displayed in the Graphics Viewer.

To Animate Data

1. Include "fgl.h" to appropriately contain the visualization routine(s).
2. Construct and initialize the mfArrays for plotting.
3. Create a static plot of the graph to be animated.
4. Set up an iteration loop for the range of data to be observed through animation.
5. Within the loop, use procedure *call mfGSet(handle, 'axis-data', data)* to update the targeted data of the current draw.
6. Update the drawn figure accordingly by using procedure *mfDrawNow*.
7. Use procedure *call mfViewPause* after the end of animation to observe the static graph.

Example Code (ch4-2)

Modify the previously created “**main1.cpp**” file as follows and rename it “**main2.cpp**”:

```
#include "fml.h"           // Include the library file fml.h
#include "fgl.h"           // Include the library file fgl.h

int main(int argc, char* argv[])
{
    mfArray x, y, z, h;    // Declare mfArray variables
    int i;                // Declare an integer variable

    mfMeshgrid( mfOut(x, y), // Construct grid matrices for 3D
plotting
                mfLinspace(-3, 3, 30),
                mfLinspace(-3, 3, 30) );

    for (i = 1; i <=30; i++) // Create loop
    {
        z = mfSin(x+i/8.0) * mfCos(y); // Mathematically define z
        if (i == 1) {                // Plot a surf using mfArrays x, y, and z
            h = mfSurf(x, y, z) ;
            mfDrawNow();
        }
        else {                        // Update the figure as the
z-coordinate varies
            mfGSet(h, "zdata", z);
            mfDrawNow();
        }
    }                                // End loop
    mfViewPause();                 // Pause to display the graph
    return 0;
}
```

Line-by-Line Walkthrough

#include "fml.h"

This includes the library file fml.h which contains the MATFOR numerical procedures.

#include "fgl.h"

This includes the library file fgl.h which contains the MATFOR visualization procedures.

mfArray x, y, z, h;

This declares mfArray variables x, y, z, and h.

int i;

This declares an integer variable i.

mfMeshgrid(mfOut(x, y), mfLinspace(-3, 3, 30), mfLinspace(-3, 3, 30));

This generates x and y grid matrices from the domains specified by the mfLinspace-generated vectors. This procedure aims to solve functions for two variables by plotting 3D graphs.

for(i=1; i<=30; i++)

This creates a loop running from i=1 to i=30.

z = mfSin(x + i/8.0d0) * mfCos(y);

This defines z mathematically. Note that the equation involves the loop index i.

if (i == 1)

At the beginning of the loop,

h = mfSurf(x, y, z);

This creates a three-dimensional surface plot of the x-, y-, and z-coordinates.

mfDrawNow();

This displays the initial figure.

else

At all other times during loop execution,

mfGSet(h, "zdata", z);

This sets the z-coordinate for update.

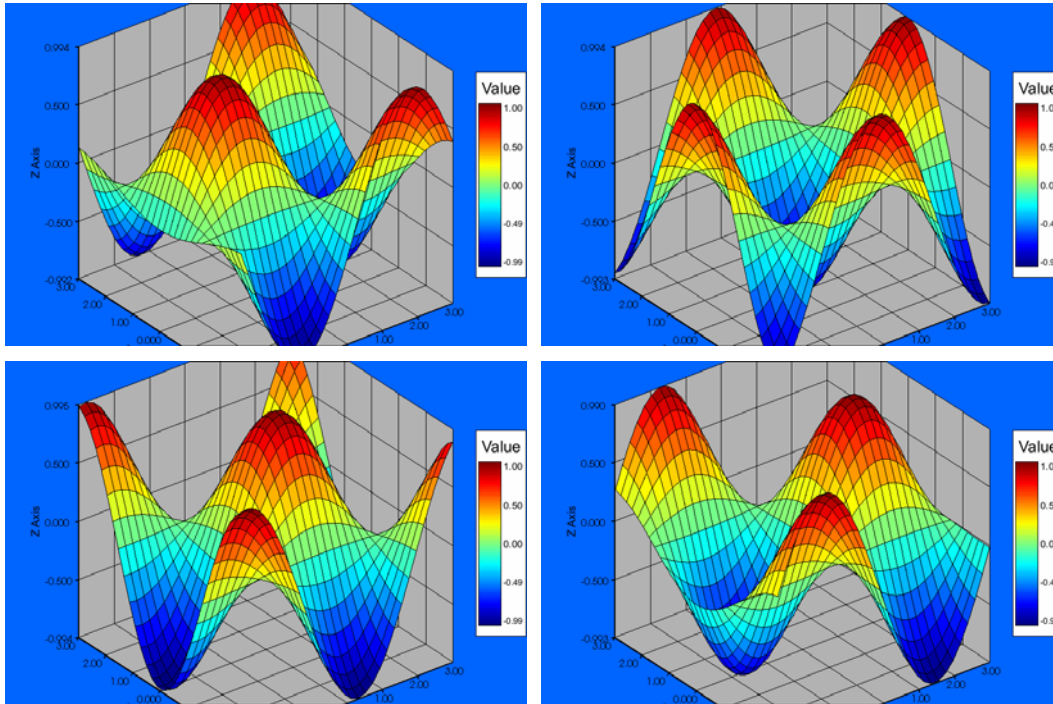
mfDrawNow();

This displays and updates the figure during loop execution.

mfViewPause();

This halts program execution for graphical display.

Compile and Run



III. TO RECORD HELLO SURFACE

This section revises the previous sample code to perform animation recording.

Recording

MATFOR allows recording of animated simulation to facilitate data presentation with comprehensiveness and readability.

To Record Animation

1. Include "fgl.h" to appropriately contain the visualization routine(s).
2. Use procedures **mfRecordStart("animation.avi")** and **mfRecordEnd()** before and after the animation codes to record the animation.

Example Code (ch4-3)

Add the recording routines to the previously created "**main2.cpp**" file and rename it "**main3.cpp**" (Note: only the modified and added lines are shown in colors):

```
#include "fml.h"           // Include the library file fml.h
#include "fgl.h"           // Include the library file fgl.h

int main(int argc, char* argv[])
{
    mfArray x, y, z, h;    // Declare mfArray variables
    int i;                 // Declare an integer variable
    mfMeshgrid( mfOut(x, y), // Construct grid matrices for 3D
plotting
                mfLinspace(-3, 3, 30),
                mfLinspace(-3, 3, 30) );

    mfRecordStart( "demo3.avi" ); // Begin recording

    for (i = 1; i <=10; i++)      // Create loop
    {
        z = mfSin(x+i/8.0) * mfCos(y); // Mathematically define z
        if (i == 1)                // Plot a surf using mfArrays x, y, and z
        {   h = mfSurf(x, y, z) ;
            mfDrawNow();
        }
        else                        // Update the figure as the z-coordinate
varies
        {   mfGSet(h, "zdata", z);
            mfDrawNow();
        }
    }                               // End loop

    mfRecordEnd();               // End recording

    mfViewPause();               // Pause to display the graph
    return 0;
}
```

Chapter

5

THE ADVANCED FEATURES

This chapter presents the innovative MATFOR 4 features that are functionally powerful in advanced programming. The first part concerns mfPlayer, which aims to enhance data presentation and accessibility. The second part introduces MATFOR GUI Builder with its friendly interface. The last part focuses on MATFOR widget components, and how they are used to facilitate application-building and code integration.

I. MFPLAYER

mfPlayer is an exclusive visual tool by which the previously saved numerical data is read and displayed. As MATFOR saves the simulated data into a MATFOR-defined MFA file, mfPlayer is one approach to present the file as a recorded animation.

To record simulated result(s) into an MFA file, simply use procedures **msRecordStart()** and **msRecordEnd()** before and after the animation codes to record the animation:

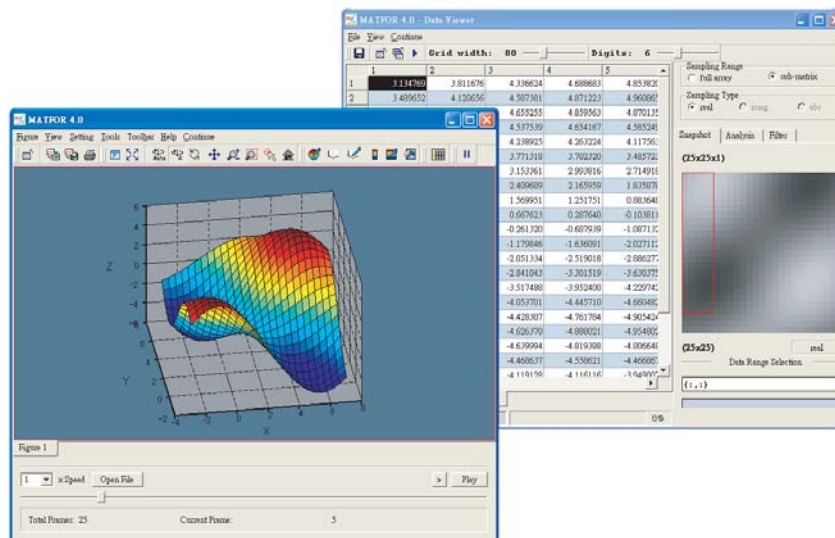
```
call msRecordStart( 'filename.mfa' )
```

```
// animation codes
```

```
call msRecordEnd
```

To play an MFA file with mfPlayer.

1. Go to Start ► Program Files ► MATFOR4 ► Utilities ► mfPlayer
2. Click **Open File** to find MFA file and click **Play**.

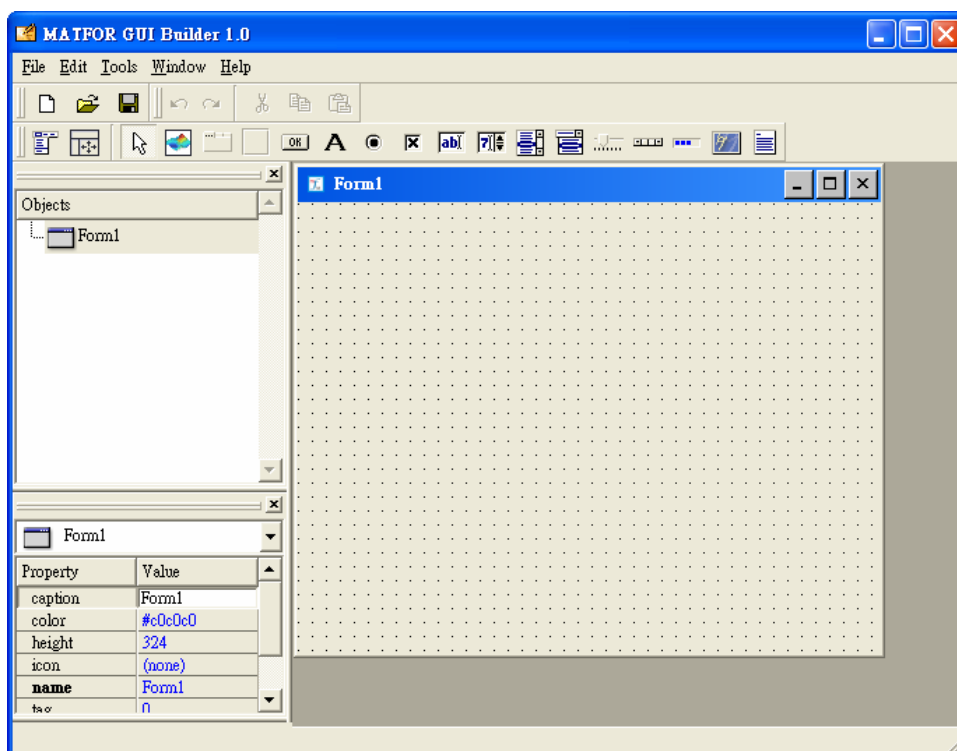


II. MATFOR GUI BUILDER

MATFOR GUI Builder allows users to create an interface of their preference, facilitating application-building by packaging the complex codes.

To Build an Application

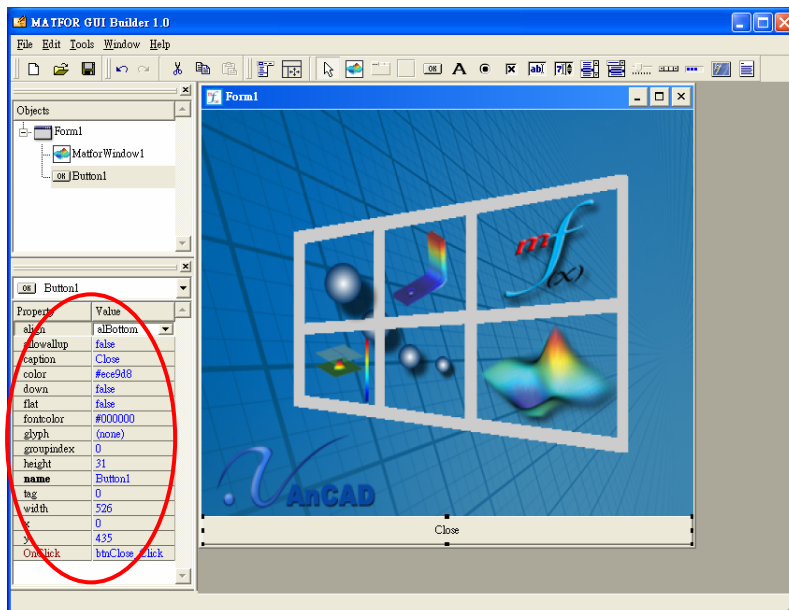
1. Go to Start ► Program Files ► MATFOR4 ► Utilities and run MATFOR GUI Builder.



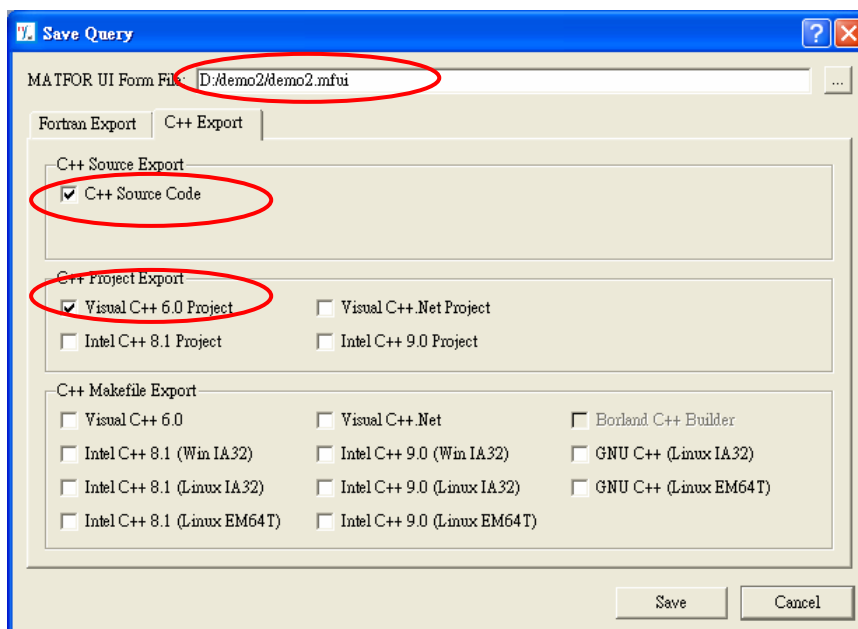
2. Add **MATFOR Window** and **Button**; click on **Button**.
3. Set **Align** to **alButton**; enter **Close** for **Caption** to; and type in **btnClose** for

Name.

4. Select **OnClick** from **Event Handler** and set it to **btnClose_Click**.
5. Set **MATFOR Window Align** to **alClient**.



6. Click **Save** and select **C++ Export Tab**.
7. Name the file “**demo2.mfui**.”
8. Select the checkboxes **C++ Main** and **VC++ 6.0 Project**.



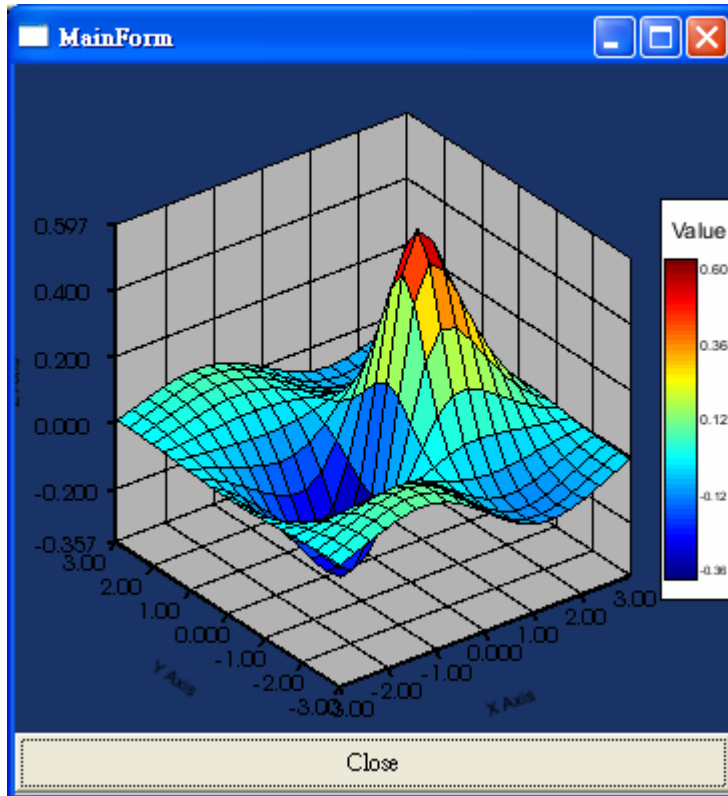
9. Open the project “**demo2_vc6.dsp**,” and modify the code in “**demo2.cpp**” as follows:

```
MF_CALLBACK btnClose_Click(const char* sender)
{
```

```
exit(0);
```

```
}
```

Compile and Run.

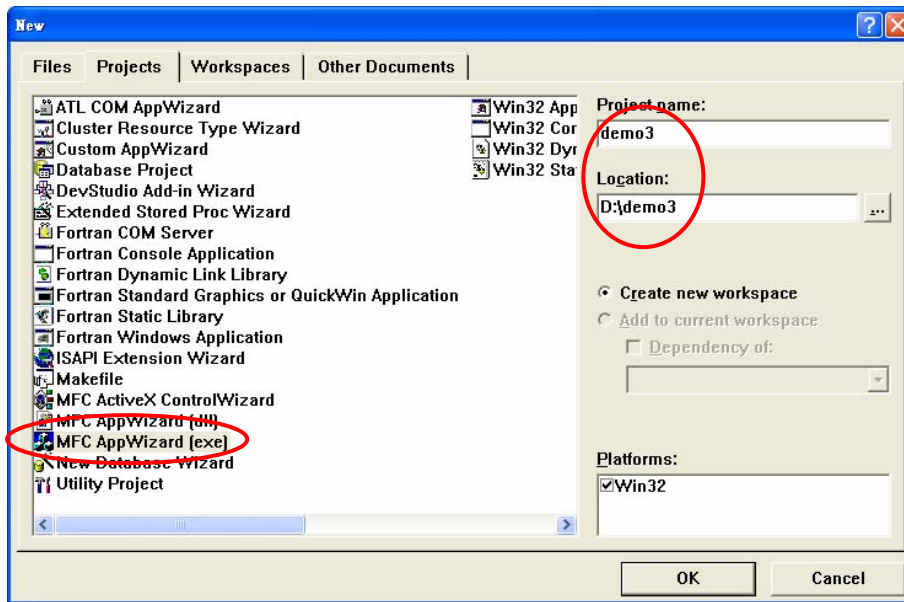


III. MATFOR WIDGET

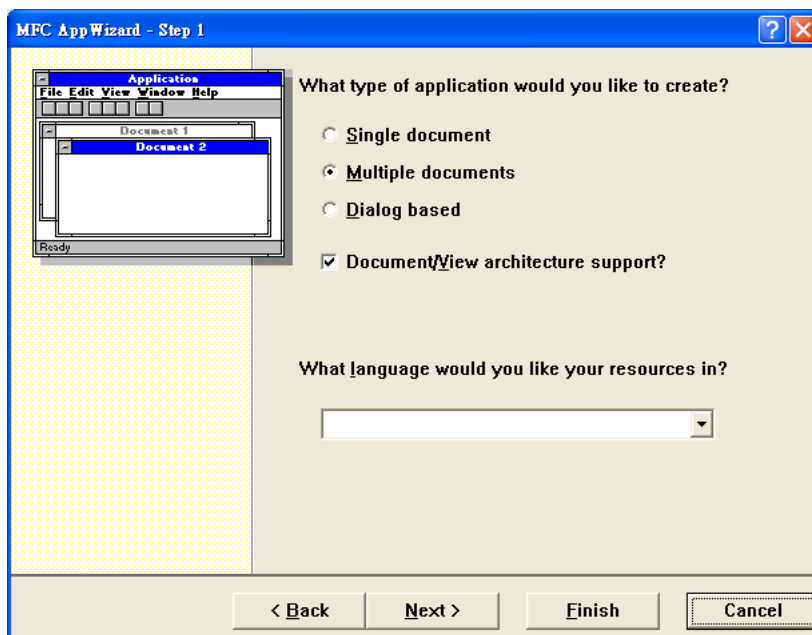
As MATFOR GUI Builder develops an interface MFUI file, the MATFOR widget component along with MATFOR libraries can be embedded into common UI design environments to enable application execution on multiple C++ compilers.

Integration with Visual C++ (ch5-2)

1. Select the project type **MFC AppWizard (exe)**.
2. Name the project "**demo3**," specify its location, and press **OK**.



3. Select **Multiple documents** and click **Finish**.



4. Open “**demo3View.cpp**” and add:

```
#include "MxWidget.h"
#include "fml.h"
#include "fgl.h"
```

Then add,

```
void CDemo3View::OnDraw(CDC* pDC)
{
```

```

CDemo3Doc* pDoc = GetDocument();
ASSERT_VALID(pDoc);
// TODO:    Add draw code for native data here
//          Add this line to draw content when OnDraw occurs
m_Widget->Render();
}

```

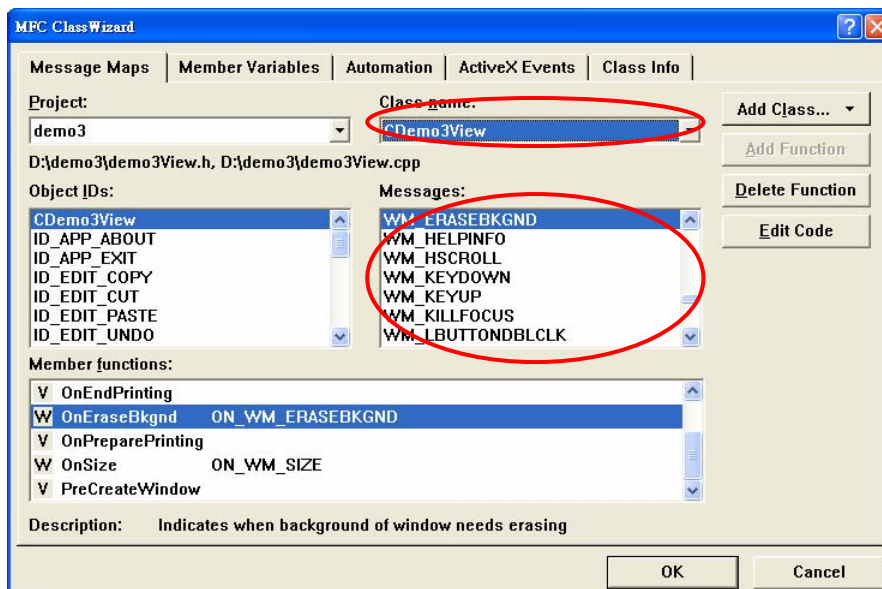
5. Add the following codes to “demo3View.h”:

```

class MxWidget;
class CDemo3View : public CView
{
protected:
    MxWidget* m_Widget;
};

```

- Open MFC ClassWizard (Ctrl+W) and set Class Name to **CDemo3View**.
- Under Messages, select **WM_CREATE**, **WM_ERASEBKGND**, and **WM_SIZE** and click **Add Function** respectively.



8. Add/Modify the following codes in “demo3View.cpp”:

```

int CDemo3View::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CView::OnCreate(lpCreateStruct) == -1)
        return -1;
}

```



```

static int flag = 0;

// Create MxWidget by given win32 window handle
m_Widget = new MxWidget(this->m_hWnd);
m_Widget->SetBackground(0.2, 0.5, 0.7);
m_Widget->SetActive();

// Create surf or other draw other contents
mfArray x,y,z;
mfCreateSurfData( mfOut(x, y, z), 1, 30, 35 );
mfSurf( x, y, z );
mfResetCamera();

// Optional: change view mode
if (flag==0)
{
    mfAxis2DMode();
    flag = 1;
}
else
{
    mfAxis3DMode();
    flag = 0;
}

return 0;
}

void CDemo3View::OnSize(UINT nType, int cx, int cy)
{
    CView::OnSize(nType, cx, cy);

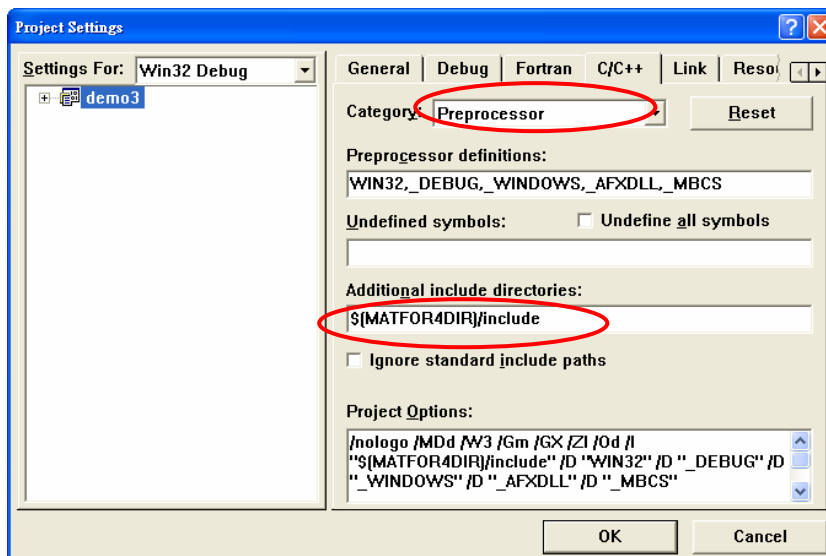
    // Resize when necessary
    m_Widget->SetSize(cx, cy);
}

BOOL CDemo3View::OnEraseBkgnd(CDC* pDC)

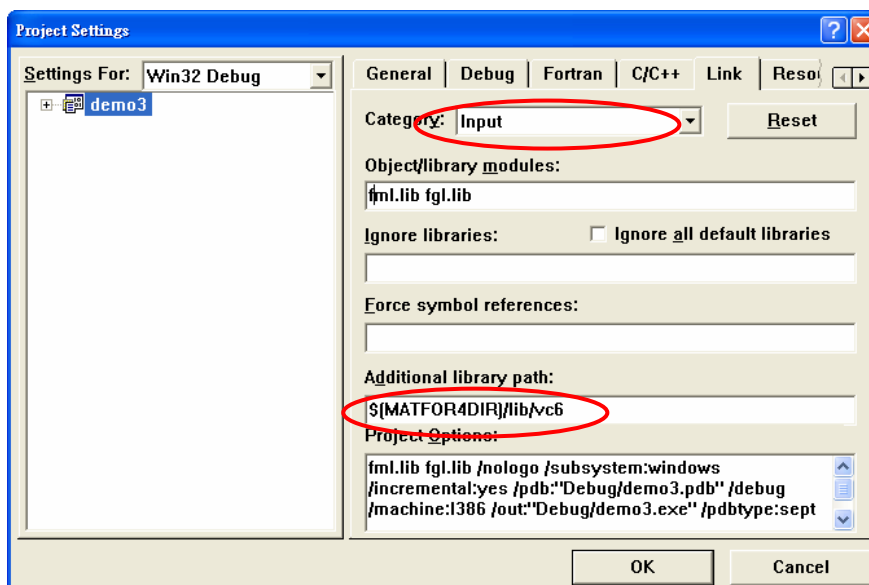
```

```
{
    // Overwrite OnEraseBkwnd function to avoid screen flick
    return true;
}
```

9. Select **Settings** from the Project menu.
10. Click on the C/C++ label and select **Preprocessor** from the Category pull-down menu
11. Under Additional include directories, type in **\$(MATFOR4DIR)/include**.

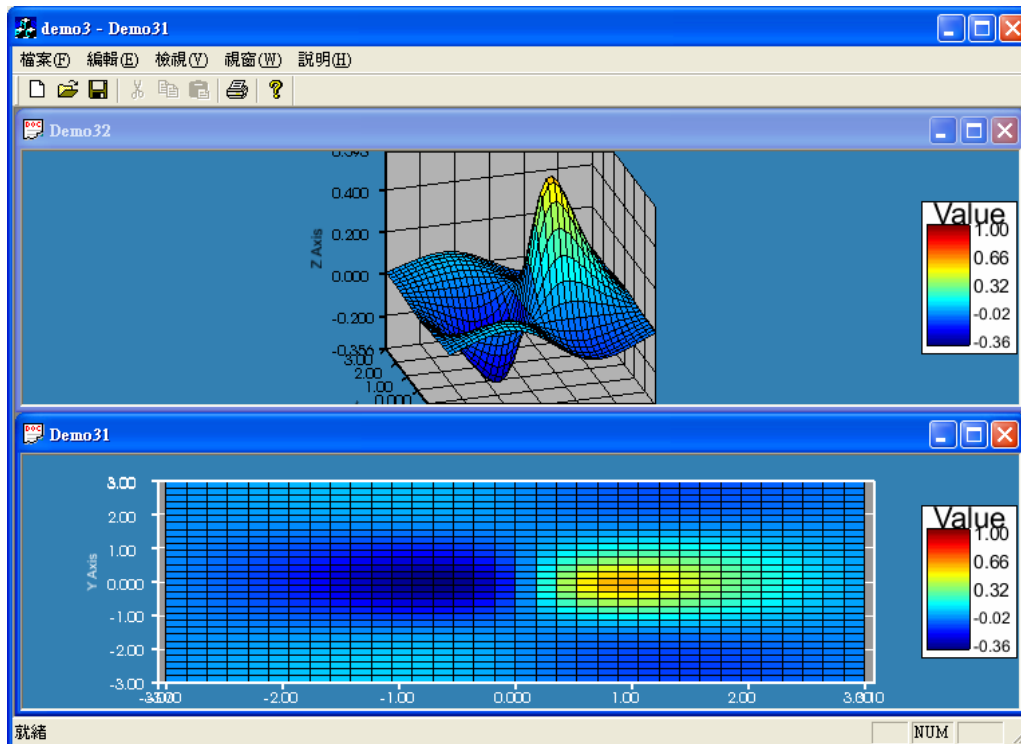


12. Click on the Link label and select **Input** from the Category pull-down menu.
13. Under Additional library path, type in **\$(MATFOR4DIR)/lib/vc6**.



Compile and Run

Click on **New** button once to plot Demo 32.



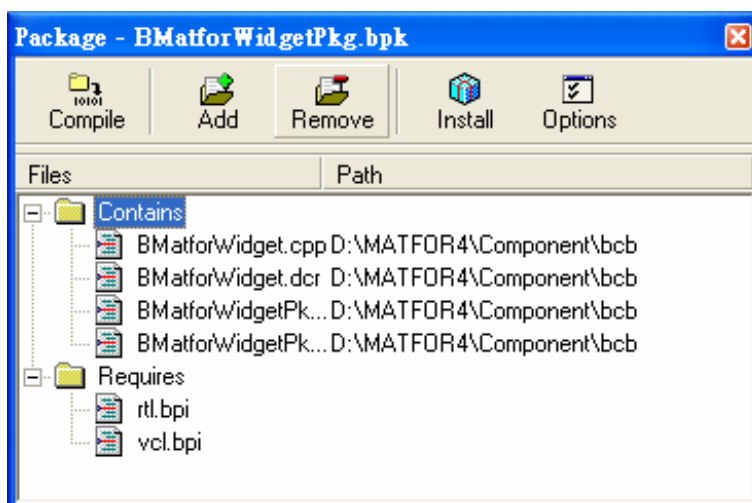
Integration with Borland C++ Builder (ch5-3)

1. Go to File and select **Close All**.
2. Go to File and select **Open Project**.

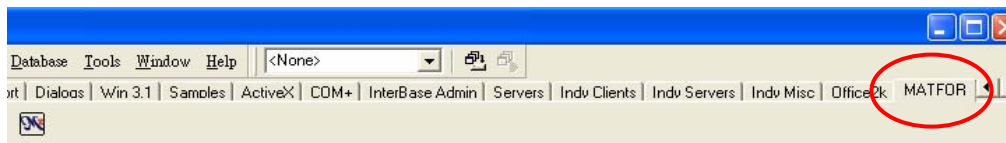
Select `$(MATFOR4DIR)\Component\bcb\BMXWidgetPkg.bpk`.

Click **Compile** and then **Install**.

3. Save and Close.



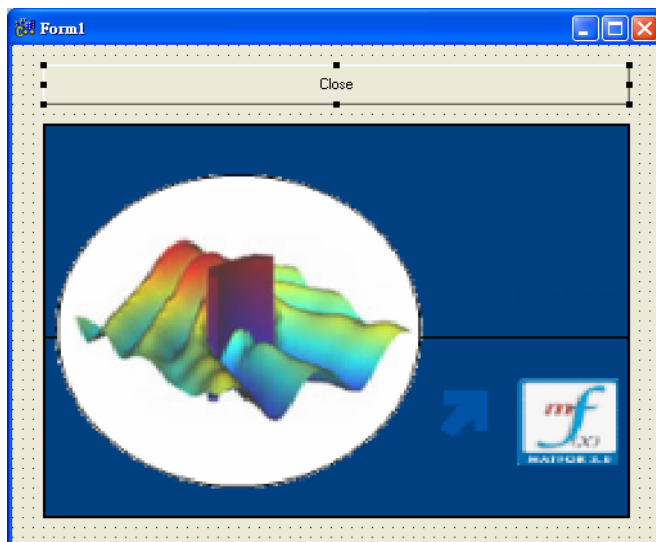
4. Go to File and select **New** → **Application** to open a new application.

5. Add **Button** and **BMatforWidget** to **Form1**.6. Select **Button** and enter **Close** for **Caption**.

Select **OnClick** from **Events** and set it to **Button1Click**.

7. Add the following codes:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    Close();
}
```

8. Press **F9** to compile and run.9. Select **OnCreate** from **Events** and set it to **FormCreate**.

10. Add the following codes:

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{
    mfArray x,y,z;
    mfCreateSurfData( mfOut(x, y, z), 1, 30, 35 );
    mfSurf( x, y, z );
    mfDrawNow();
}
```

11. Add the following headers to “Unit1.cpp”:

```
#include "fml.h"
#include "fgl.h"
#pragma link "fml.lib"
#pragma link "fgl.lib"
```

12. Go to Project→Options and select Directories/Conditionals to add paths.

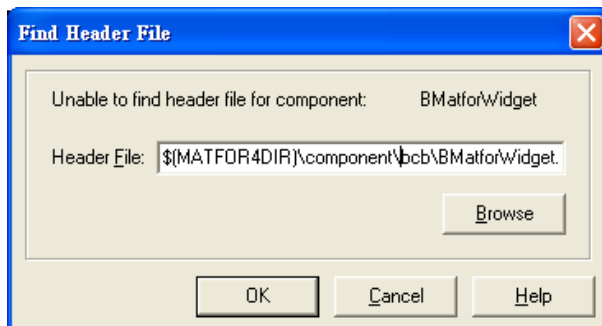
13. In Include Path, add $\$(MATFOR4DIR)\component\bc\$.

In Library Path, add $\$(MATFOR4DIR)\component\bc\$.

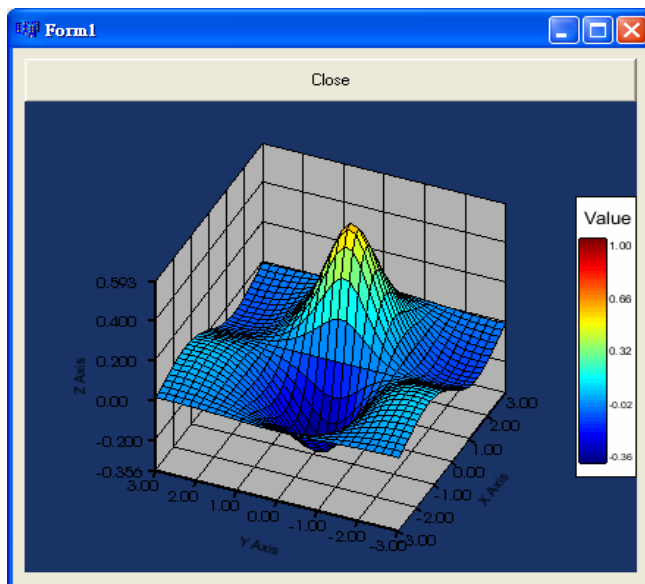
14. Save Project (“Project1.bpr”) and Unit1 (“Unit1.cpp”).

15. If the following dialog box appears, type in:

$\$(MATFOR4DIR)\Component\bc\BMXWidget.h$



Compile and Run



Integration with Win32 SDK (ch5-4)

1. Prepare a Win32 skeleton program and a makefile (see “win32_skeleton.cpp” and “win32_skeleton.mak”) and rename them as “demo7.cpp” and “Makefile.”

2. Add variable declarations in **long FAR PASCAL WndProc(...)** of “demo7.cpp”:

```
static MxWidget *mxWidget;  
static int winSizeX = 420;  
static int winSizeY = 420;
```

3. Add include files in “demo7.cpp”:

```
#include "MxWidget.h"  
#include "fml.h"  
#include "fgl.h"
```

4. Add additional codes to handle **WM_CREATE** in “demo7.cpp”:

```
case WM_CREATE:  
{  
    CreateWindow("button", "Exit",  
                WS_CHILD | WS_VISIBLE | SS_CENTER |  
WS_CLIPSIBLINGS,  
                0,0,420,40,  
                hwnd,(HMENU)1,  
  
(HINSTANCE)GetWindowLong(hwnd,GWL_HINSTANCE),  
                NULL);  
  
    mxWidget = new MxWidget(hwnd);  
    mxWidget->SetPosition(0, 40);  
    mxWidget->SetSize(winSizeX, winSizeY);  
    mxWidget->SetActive();  
  
    mfArray x,y,z;  
    mfCreateSurfData( mfOut(x, y, z), 1, 30, 35 );  
    mfSurf( x, y, z );  
    mfResetCamera();  
    mfDrawNow();  
    return 0;  
}
```

5. Add additional codes to handle **WM_SIZE** in “demo7.cpp”:

```
case WM_SIZE:
    winSizeX = LOWORD(IParam);
    winSizeY = HIWORD(IParam) - 40;
    mxWidget->SetSize(winSizeX, winSizeY);
    return 0;
```

6. Add additional codes to handle **WM_COMMAND** in “demo7.cpp”:

```
case WM_COMMAND:
    switch (wParam)
    {
        case 1:
            PostQuitMessage (0);
            break;
    }
    return 0;
```

7. Modify “Makfile.” For example:

```
SOURCES = demo7.cpp
OBJECTS = demo7.obj
TARGET   = demo7.exe
INCPATH  = -I"$(MATFOR4DIR)\include"
LFLAGS=  /NOLOGO /DEBUG /SUBSYSTEM:windows
LIBPATH:"$(MATFOR4DIR)\lib"
LIBS = kernel32.lib user32.lib gdi32.lib fgl.lib fml.lib mxui.lib
```

Build and Run

