GUI design in C++

Using MFC (Microsoft Foundation Classes)

By: Rishinder Paul

Introduction to Visual Studio 2010 and MFC

Contents

- Some basics about Windows Applications
- Introduction to MFC
- Document Interface in MFC
- Creating 1st MFC Application
- Writing your 1st MFC Code
- Running your Code

Windows and Console Applications

- Windows Applications are different from Console Applications.

  ◦ They let us use the features of the Windows OS.
    - We don't create a line by using – (Dashes) or _ (Underscores) but by using functions like MoveTo() and LineTo().
  ◦ Allows the implementation of events like Mouse click, button clicks, etc.

GUI and Win32 API

▸ It provides us with the lowest level of interaction with Windows OS.

| WIN–32 | → | WIN–OS |

▸ For GUI design, it provides all the necessary support needed in the form of functions.

▸ It does not makes the use of Object Oriented Framework.

GUI and MFC

▸ It's a higher level of interaction to Windows OS which uses Win32 API as an interface.

| MFC | → | WIN-32 | → | WIN-OS |

▸ It lets you use Object Oriented Framework.
▸ Less code, more work!

How MFC works?

- MFC application uses the concept of Document and View.
  - Document is the name given to the collection of data in our application.
  - Document interface describes how many documents does our application deals with.
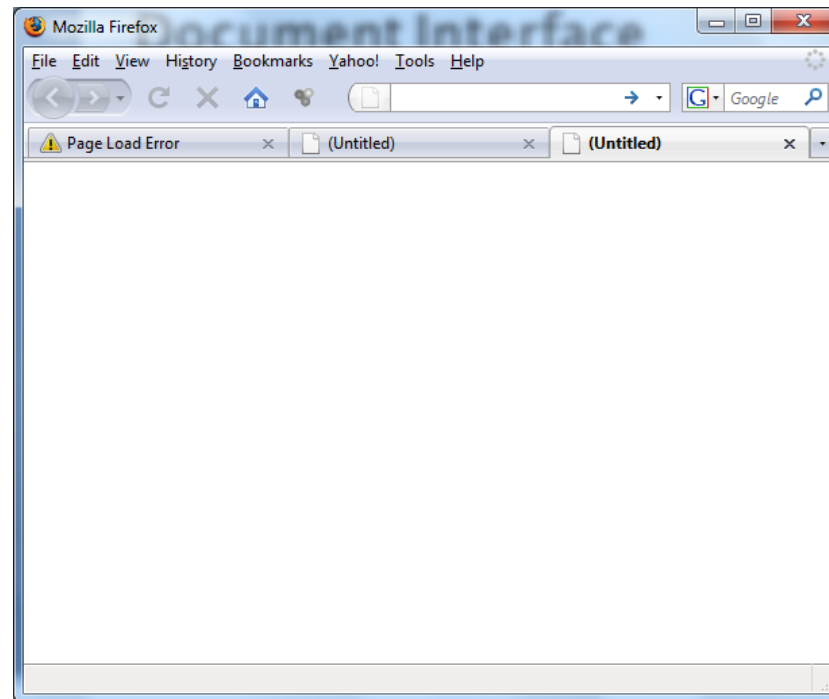 - View defines how that data is to be displayed.

Document Interface

- SDI – Single Document Interface applications have just one document, with which they have to deal with e.g. Notepad.
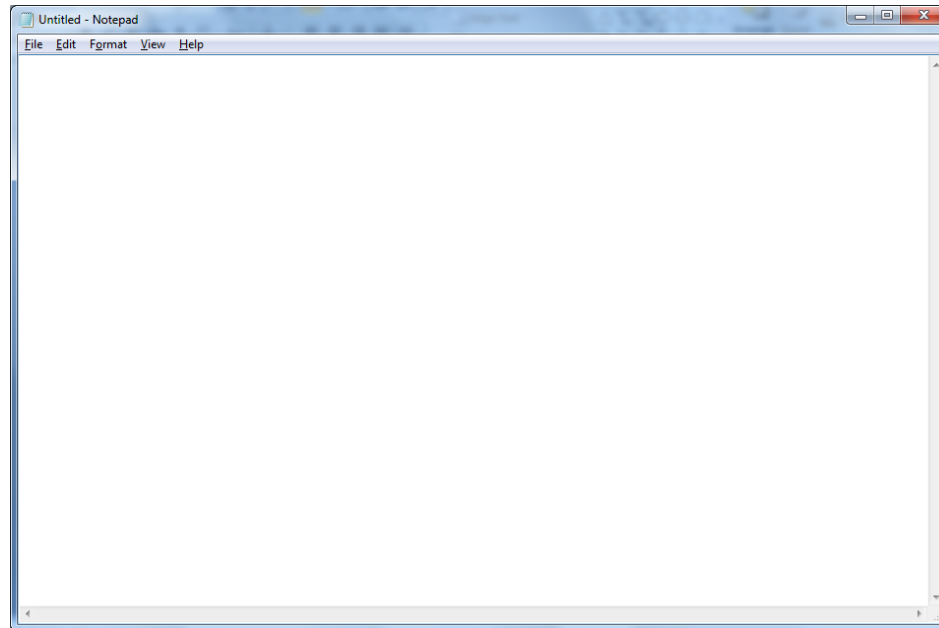
Document Interface

- MDI – Multiple Document Interface applications are used for programs needing several documents to be open at one time e.g. Firefox.

Enough of Basics ☺

Visit http://www.winprog.org/tutorial/apivsmfc.html for more on this discussion.

▶ The following slides are concerned with the tasks mentioned below:

- Creation of First MFC application.
  - Using Visual Studio IDE for project creation.
- 1st Step before beginning to code.
- Your 1st Code
  - Understanding your window coordinates.
  - Drawing a line
  - Output Screenshot
  - Explanation

Creating a new MFC project

- Open Visual Studio
- If it prompt you about the development settings, just Choose "Visual C++ Development".

- On the menu bar, go to File->New->Project.
- Select MFC form the installed templates on Left-hand panel.
- Select MFC Application from the Right-hand Panel.
- Enter a Project Name e.g. "BloodB" and Click OK.

New Application Wizard
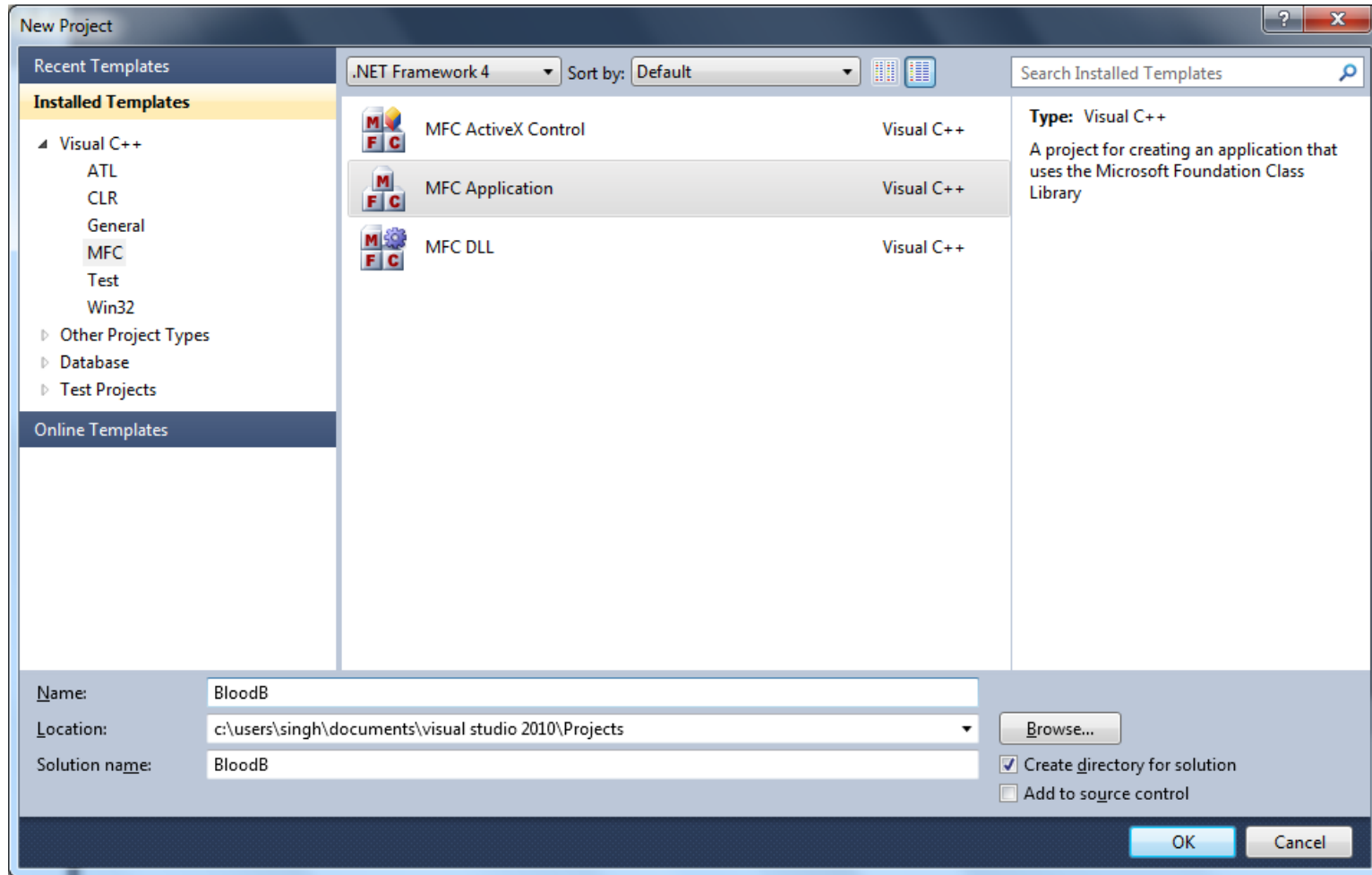
MFC Application Setup

▸ Click Next on the MFC Application wizard and chose the following settings:

◦ <u>Application Type</u>: Multiple Documents.
  • As our project may need more documents in future. You can even choose Single Document if you want.
◦ <u>Project Style:</u> MFC Standard.
  • This is a simpler style to work with in beginning.
◦ Click Finish. *(See the Screenshot in Next Slide)*
◦ You are ready to start coding now.

MFC Application Wizard
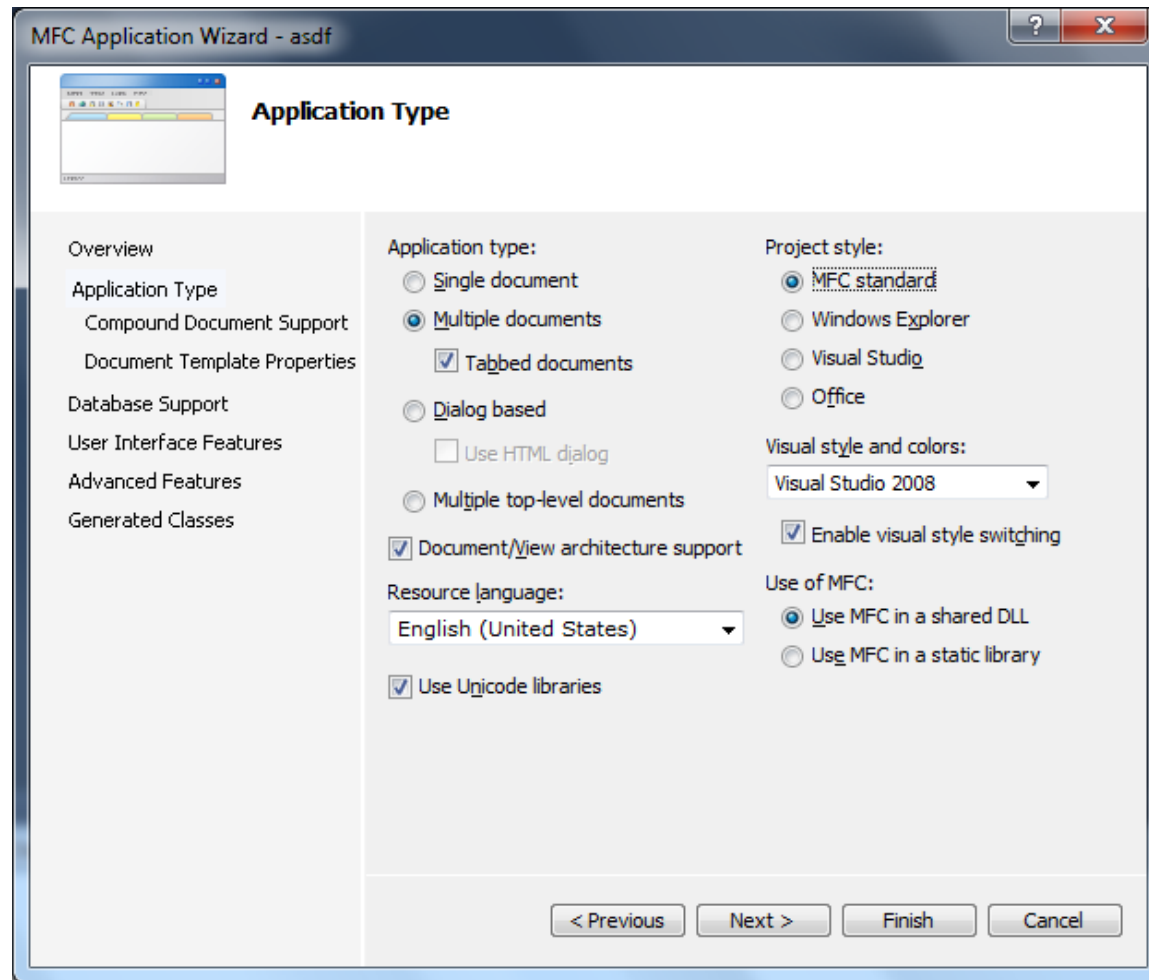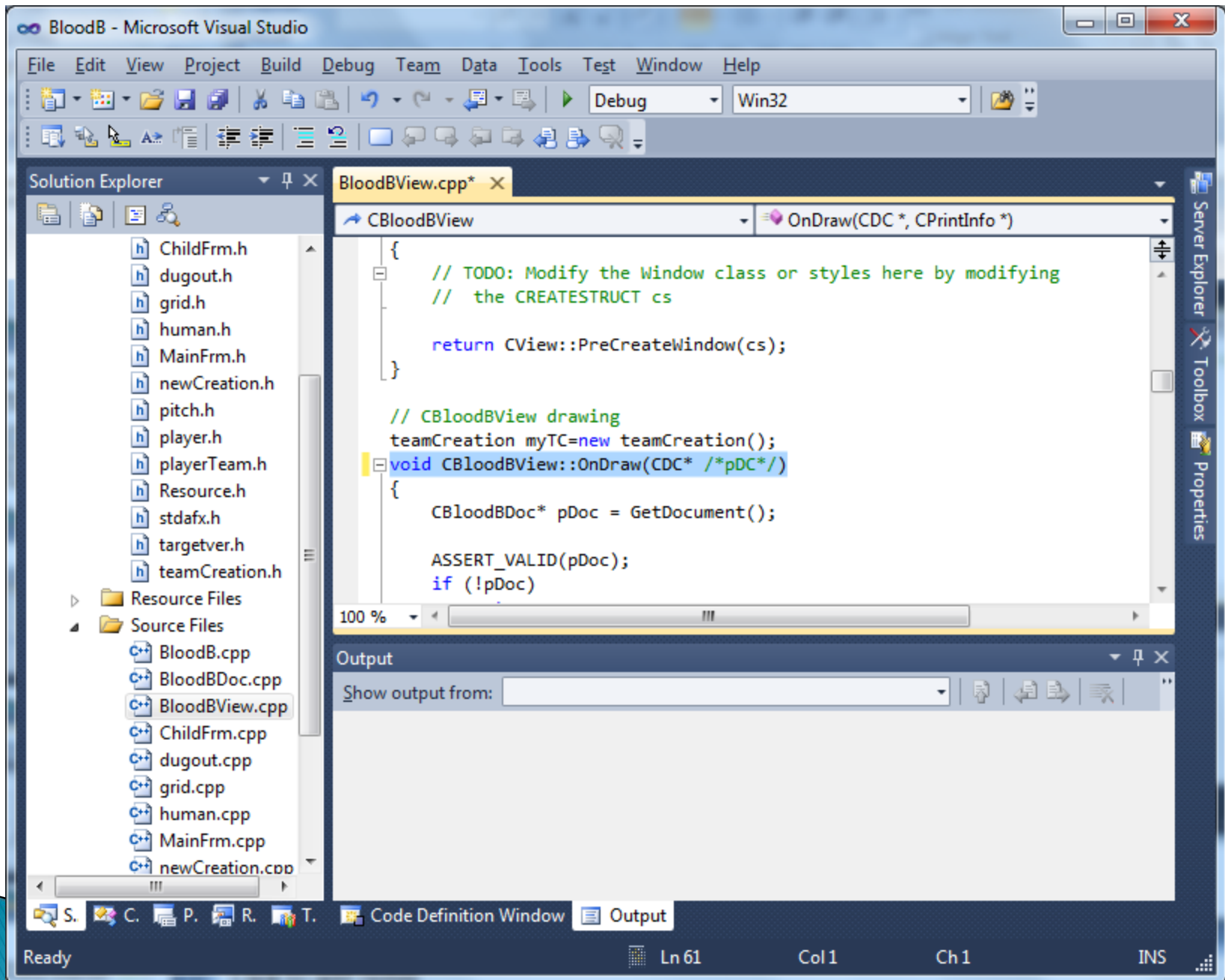
1ˢᵗ thing to do…
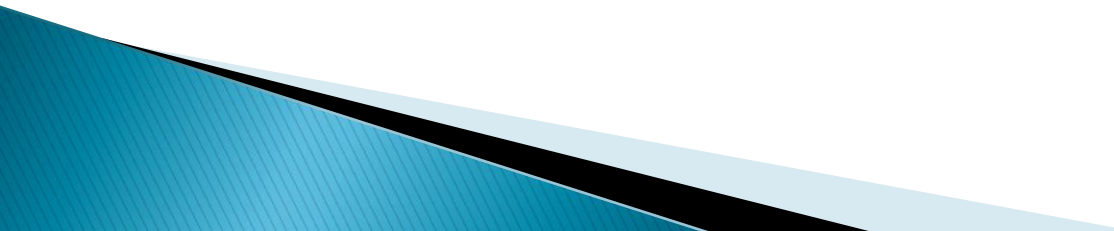
- Before beginning to draw, we need the drawing class object.

- For this we need to do the following:
  - From the Solution Explorer open "BloodBView" by double clicking it.
  - You will get the whole source code in front of you.
  - There you will see a function "OnDraw()".
  - Go to that line of code and uncomment "pDC", which is passed as a parameter to OnDraw().

BloodB - Microsoft Visual Studio

File Edit View Project Build Debug Team Data Tools Test Window Help

Debug   Win32

Solution Explorer

BloodBView.cpp*

CBloodBView   OnDraw(CDC *, CPrintInfo *)

```cpp
{
    // TODO: Modify the Window class or styles here by modifying
    //  the CREATESTRUCT cs

    return CView::PreCreateWindow(cs);
}


// CBloodBView drawing
teamCreation myTC=new teamCreation();
void CBloodBView::OnDraw(CDC* /*pDC*/)
{
    CBloodBDoc* pDoc = GetDocument();

    ASSERT_VALID(pDoc);
    if (!pDoc)
```

Solution Explorer files:
- ChildFrm.h
- dugout.h
- grid.h
- human.h
- MainFrm.h
- newCreation.h
- pitch.h
- player.h
- playerTeam.h
- Resource.h
- stdafx.h
- targetver.h
- teamCreation.h
- Resource Files
- Source Files
  - BloodB.cpp
  - BloodBDoc.cpp
  - BloodBView.cpp
  - ChildFrm.cpp
  - dugout.cpp
  - grid.cpp
  - human.cpp
  - MainFrm.cpp
  - newCreation.cpp

100 %

Output

Show output from:

S. C. P. R. T.   Code Definition Window   Output

Ready   Ln 61   Col 1   Ch 1   INS

- The initial code will look like:
  `void CBloodBView::OnDraw(CDC* /*pDC*/)`

- The final code should look like:
  `void CBloodBView::OnDraw(CDC* pDC)`

- Now you are ready with the basic setup. This function OnDraw() will help you with drawing on your document.

Some Important Information!

- Visual Studio provides you with all the basic files and setup, which you need for your application. So you are saved from writing the same files again and again whenever you create a new application.

- It's better to go through the details about these files and their use. It develops more understanding in the topic.

Writing your first MFC code!

- Let's draw a line first.

- We need to know the starting point and the end point.

- What is a point in MFC?

- To understand the concept of point, we need to understand the way in which MFC deals with the screen coordinates.
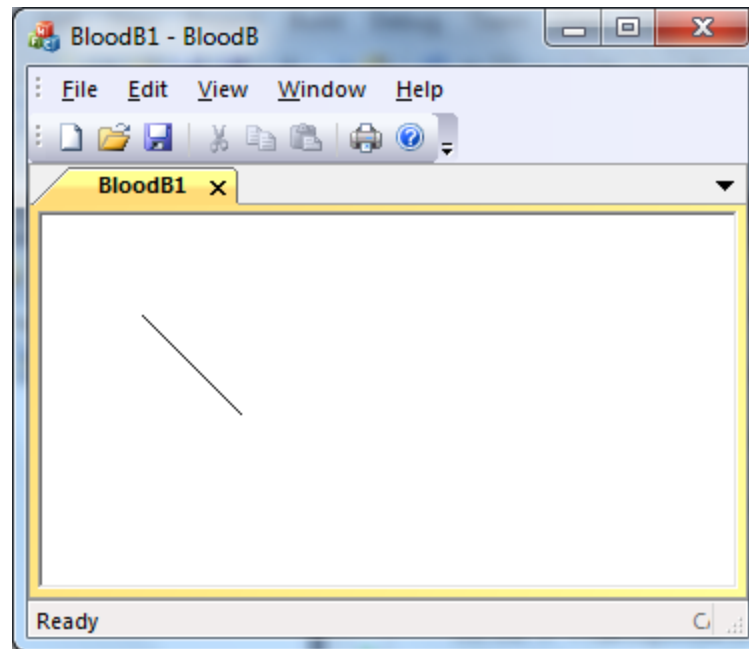
Some Coordinate Geometry

Increasing X coordinate

Increasing Y coordinate
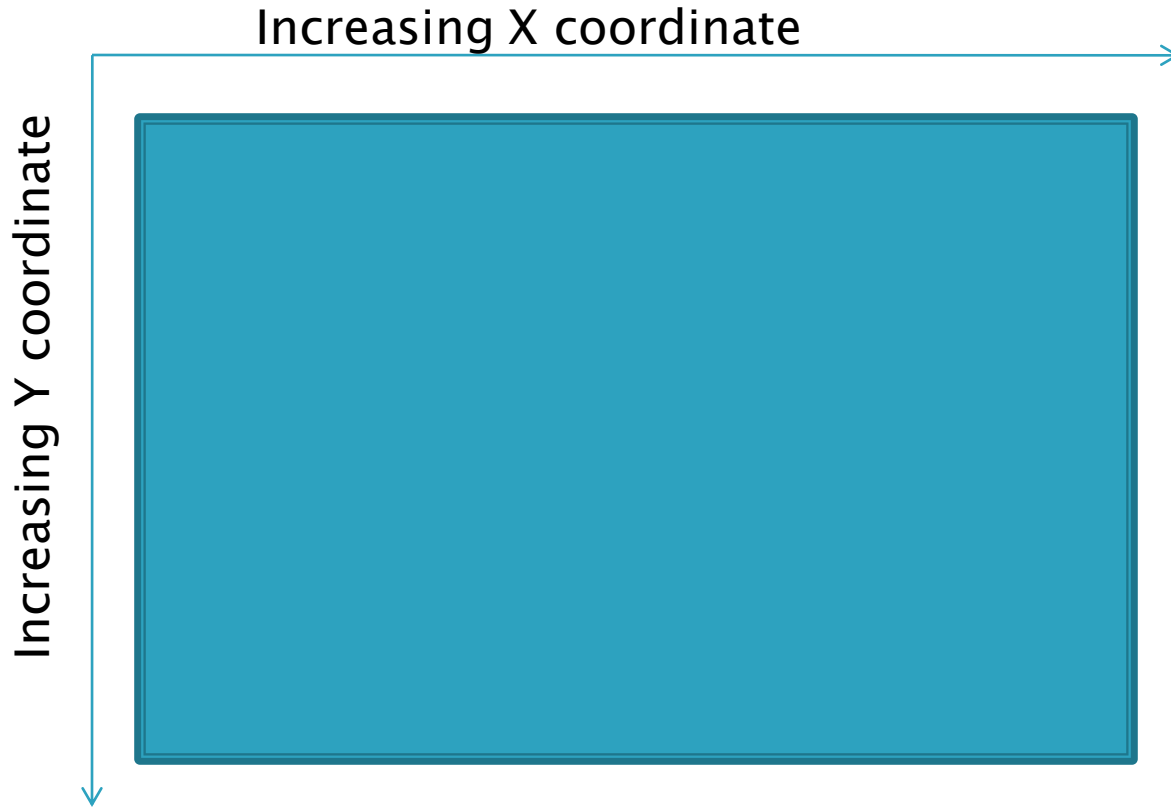
Writing your first MFC code!

▶ Where to write code?
  ◦ Within the OnDraw() function, go to the last line of code, just before the closing brace '}'. Start writing your code from there.

▶ What to write?
  ◦ pDC->MoveTo(50,50);
  ◦ pDC->LineTo(100,100);

▶ Now what?
  ◦ Hit F5 to run

Output of your code.

Explanation

- The function MoveTo(x,y) takes the coordinates of the starting point of line as parameters.
- The function LineTo(x,y) takes the coordinates of the end point of line as parameters.
- Hence it uses the drawing class object pDC to draw it for you.