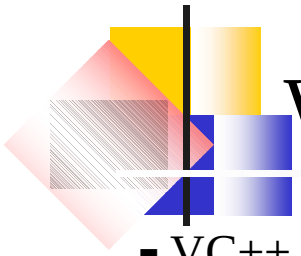
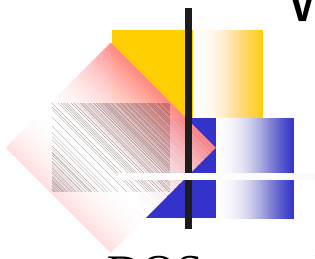
Rapid Application Development



What is VC++

- VC++ is extremely powerful tool for Windows Programming.
- VB and VC++ both are GUI based.
- It is a collection of many tools all wrapped together into one dynamic package.
- It provides us with one integrated design environment.
- It is a case- sensitive language.
- VC++ includes in MFC.
- MFC is a extraordinary package of pre written ready to use code.

Why VC++ comes into Existence

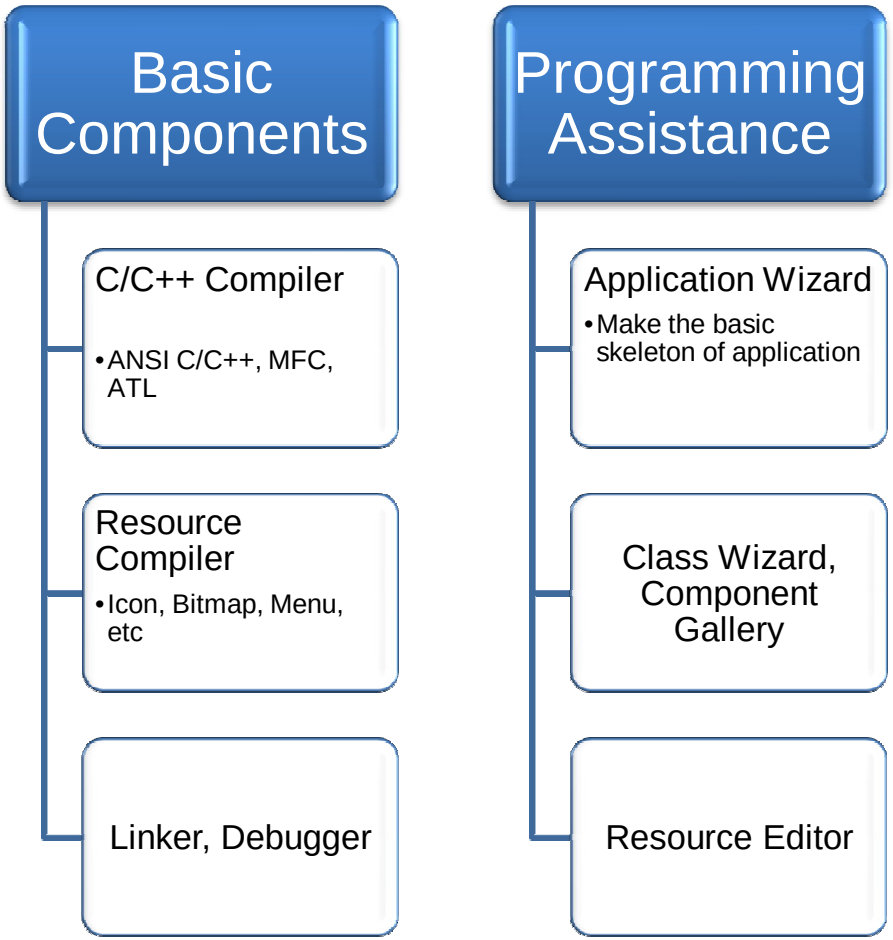


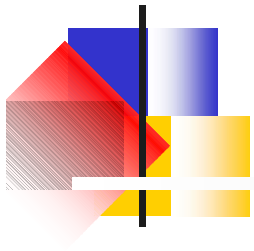
- DOS was the first OS introduced.
- Each and every command had to be written using programming language in C.
- C++ was introduced to make the data & program more manageable using class concept.
- VC++ came into existence under the visual studio 2000 package. It supports MFC.
- Microsoft Foundation Classes that are basically built-in ready to use code.
- VB doesn’t support MFC, but VC++ supports MFC.

Procedural Programming	Event-Driven Programming
1. It supports CUI, i.e. character user interface	1. It supports GUI, i.e. Graphical User Interface
2. It follows the top-down approach, i.e. For e.g. If a DOS program has three functions A(), B() and C(), then we know the order in which they are going to be called.	2. It does not follow the top-down approach. It is Event-Driven in nature, i.e. the functions get executed on the occurrence of event. They may follow any order.
3. It is sequence driven in nature.	3. In Windows, programs are driven by the event. The events may be keyboard events (Striking a key on keyboard) & mouse events (left-click, right-click, scroll, double click , etc.) that can be directed at numerous user interface objects such as menus and buttons.

4. It does not support multi tasking	4. It supports multitasking
5. In this programming model, the OS simply executes the program and then waits for it to finish.	5. In this programming model, OS not only executes the program, it also communicates with the program and does not sit idle.
6. e.g. COBOL, FORTRAN, C	6. e.g. VB, VC++

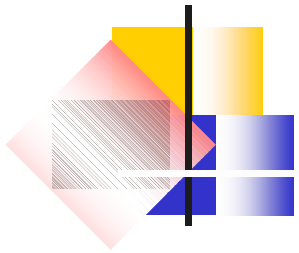
Components of Visual C++





Part 1

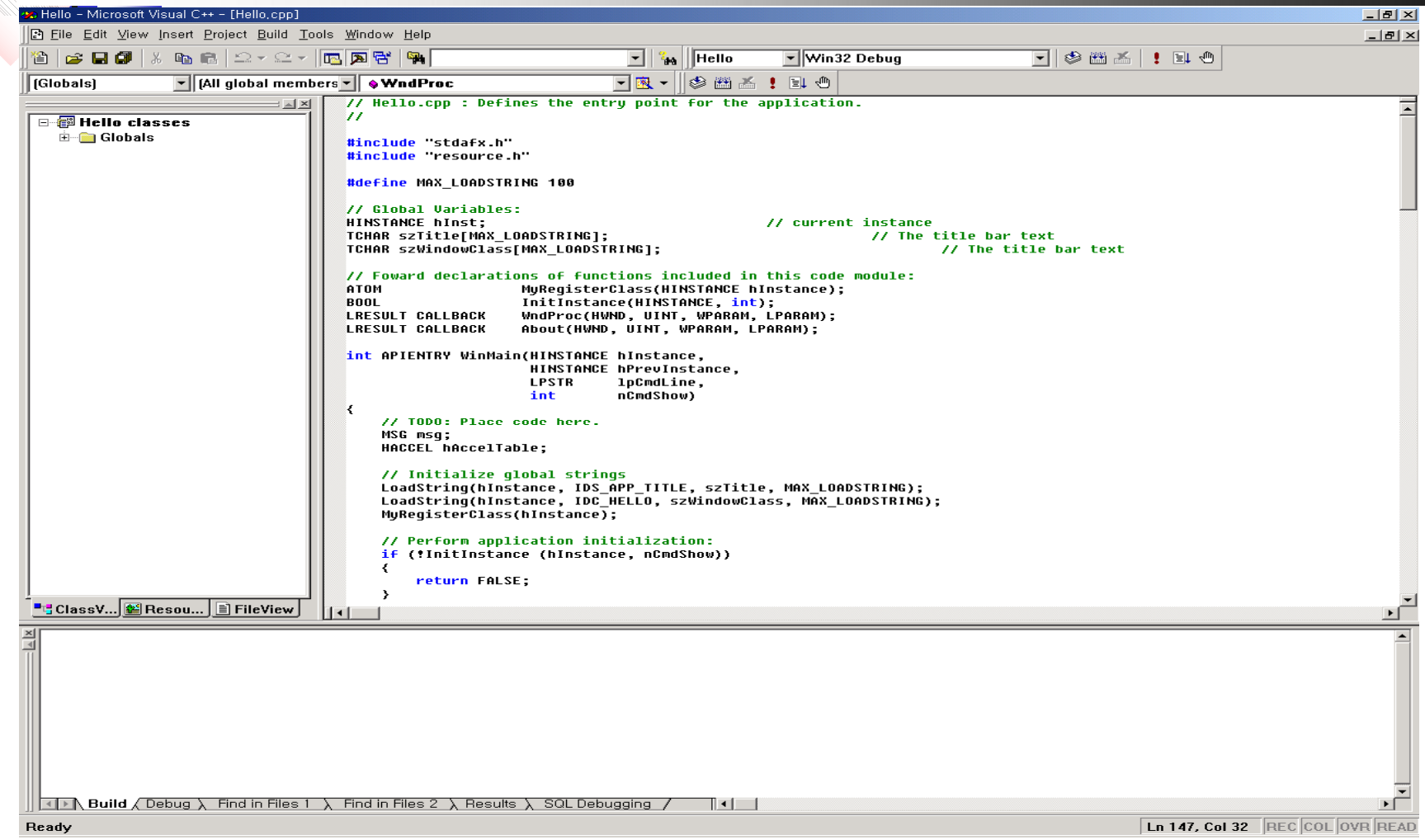
Setting the skeleton of the Application using AppWizard



Overview

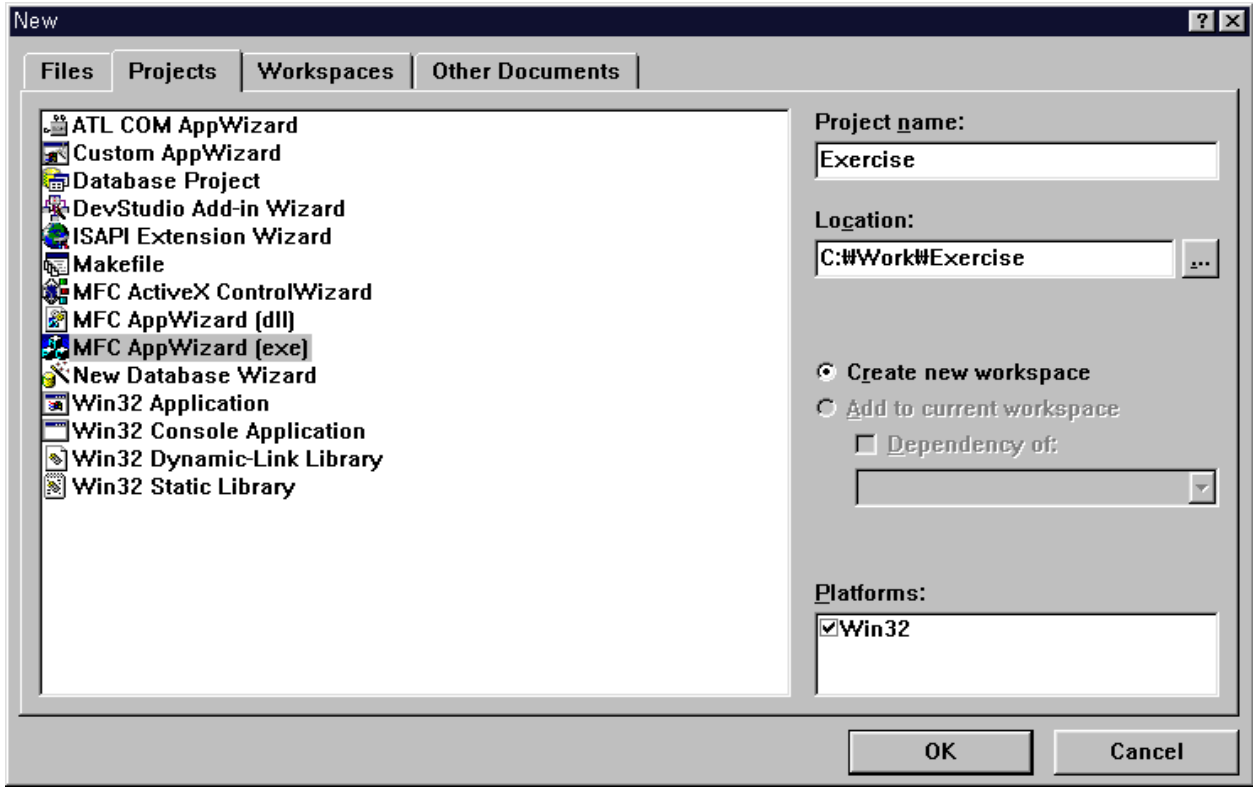
- Building Application using AppWizard
- Classes that AppWizard generated
- Exercise
- building the console applications

Building Application using AppWizard



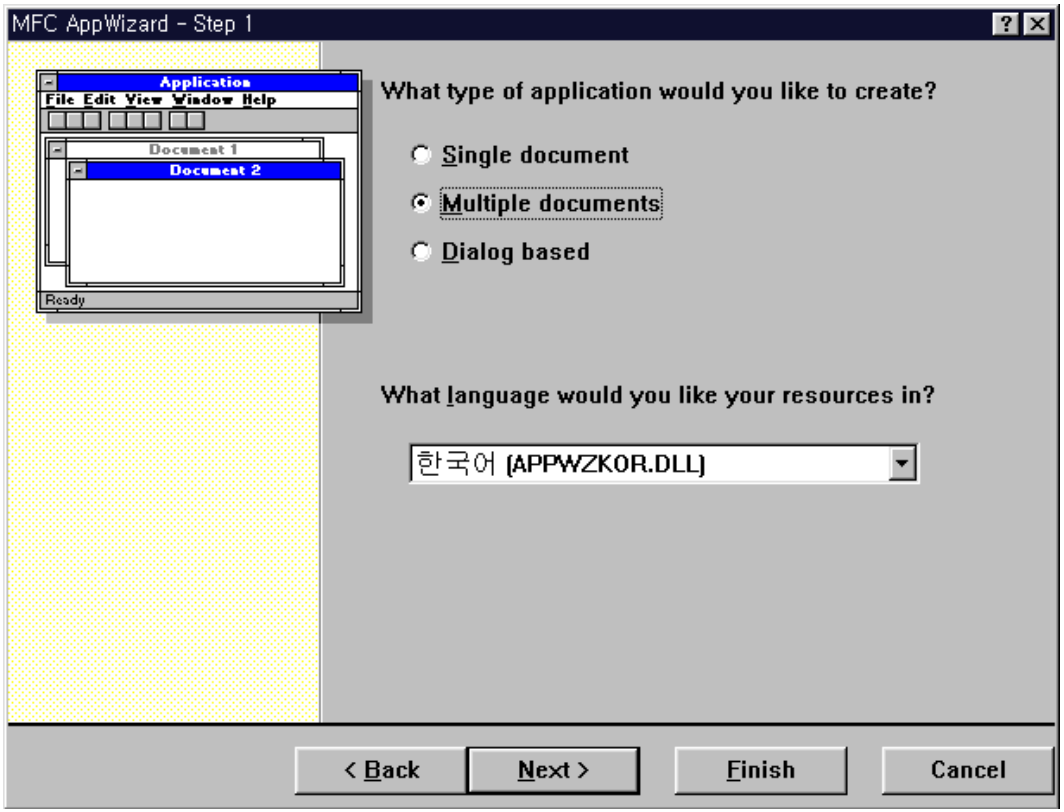
Building Application using AppWizard

- Select File / New
- Select MFC AppWizard (exe)

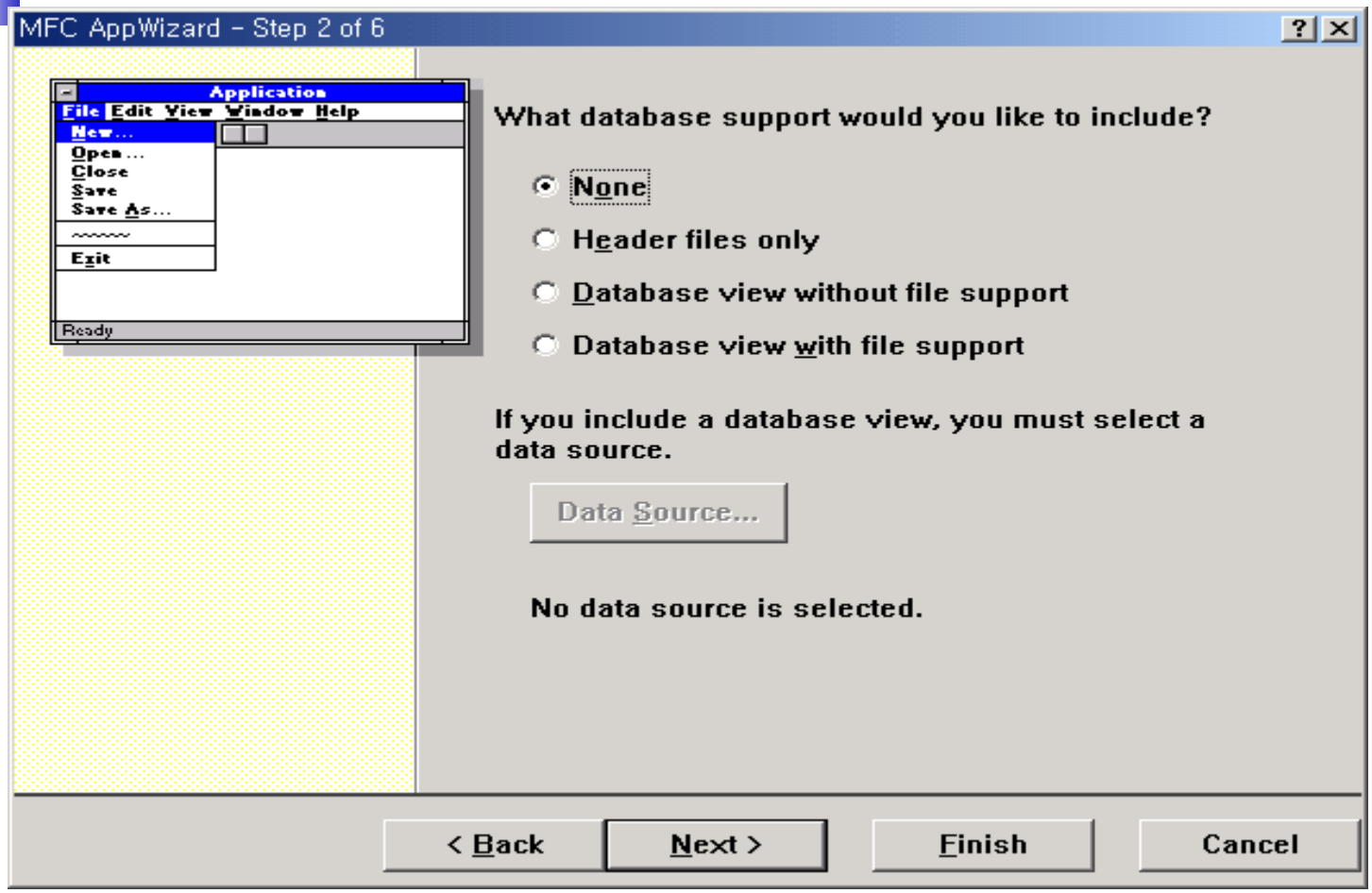


Building Application using AppWizard

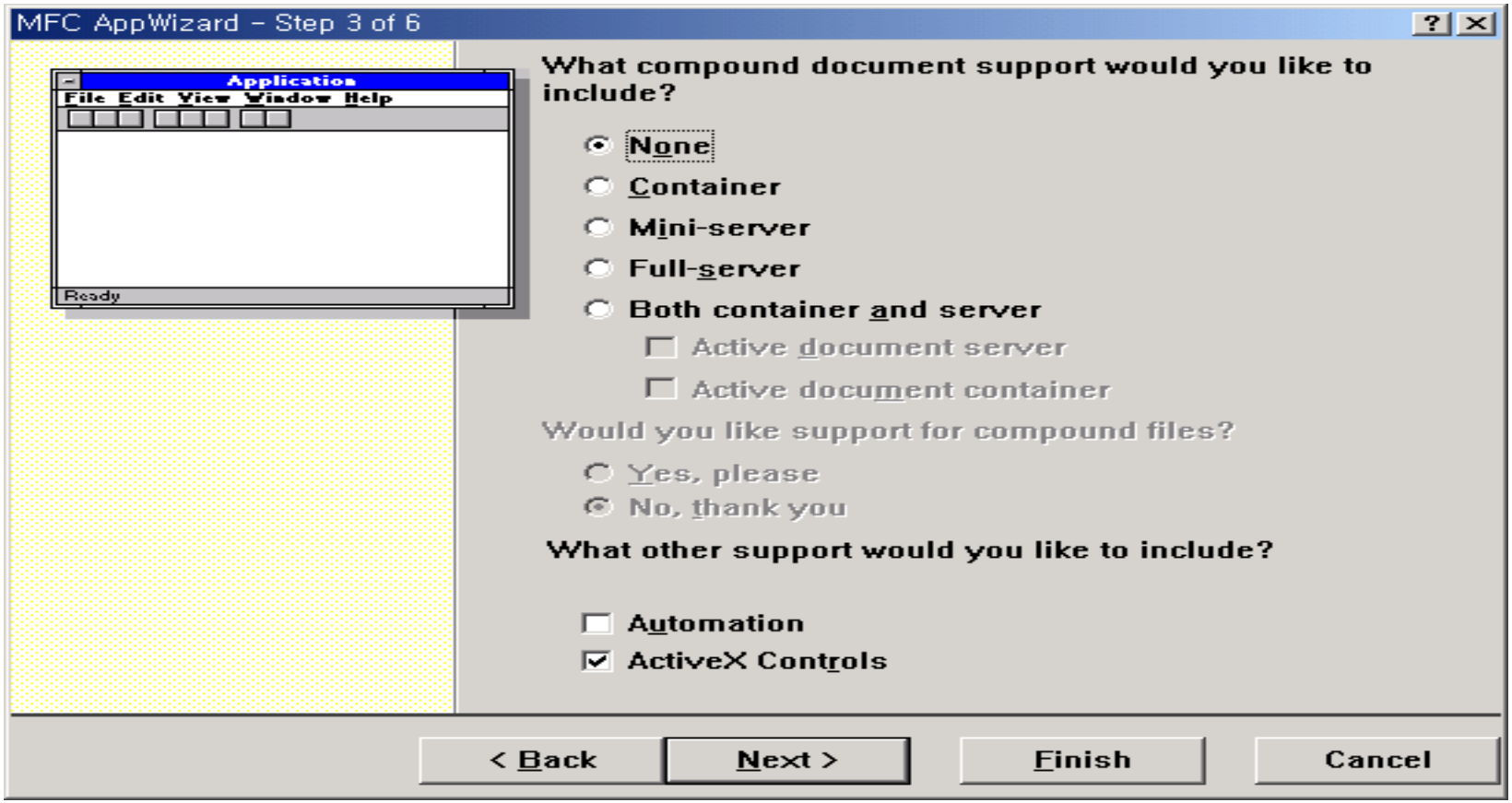
- Select “Single Document”
  - Single Document / Multiple Document / Dialog based



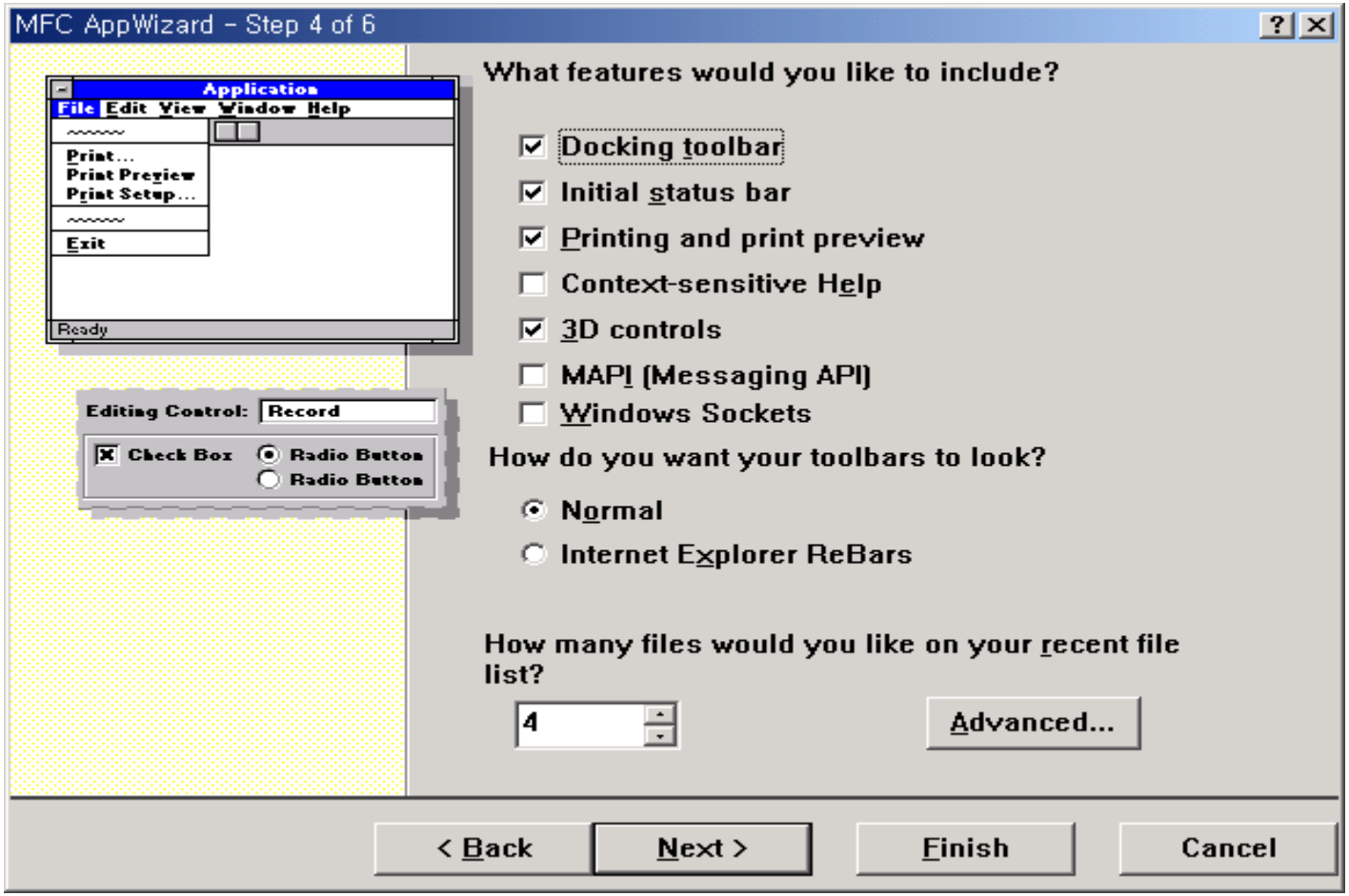
Building Application using AppWizard



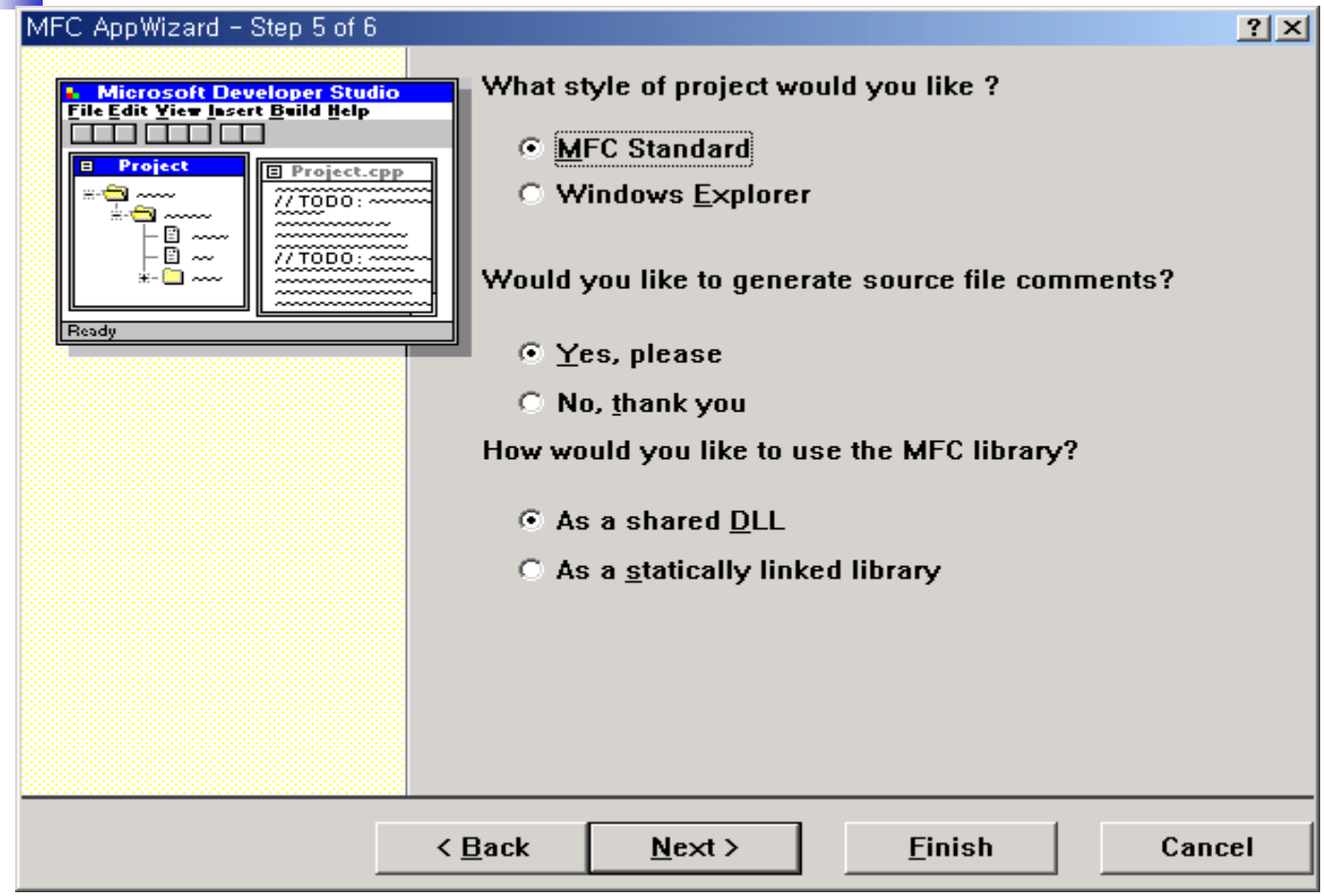
Building Application using AppWizard



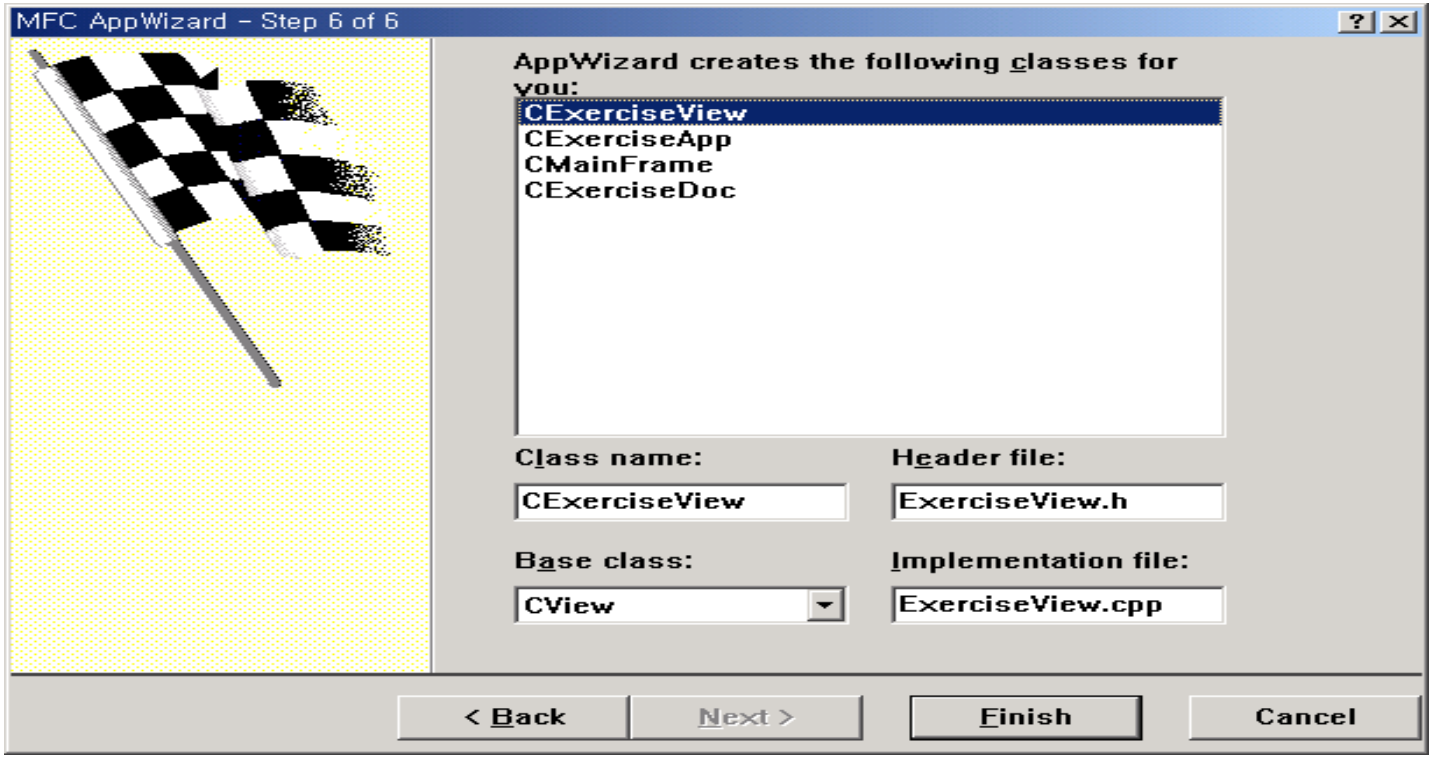
Building Application using AppWizard



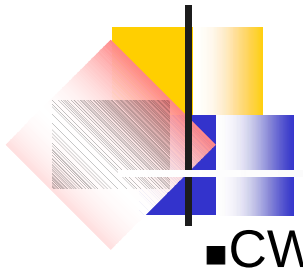
Building Application using AppWizard



Building Application using AppWizard

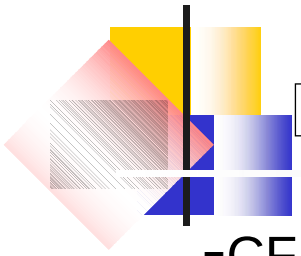


- Push Finish for the End of AppWizard



Classes that AppWizard generated

- CWinApp
  - Capsulate WinMain(), WinProc()
  - Windows Application Instance
  - theApp (Global Variable)
  - MFC Program must have an instance of the Class that inherited from this Class
  - InitInstance()
    - Initialize Application
    - Analyze the command line
    - Create and Show the main Window

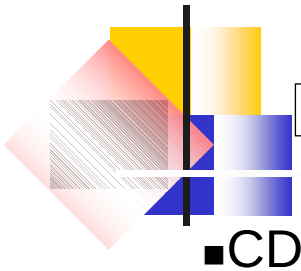


Classes that AppWizard generated

- CFrameWnd
  - Frame Window
  - Container of Other Window
  - The main windows of SDI, MDI Project
  - Created in CWinApp:InitInstance()
  - OnCreate( )
  - PreCreateWindow( )

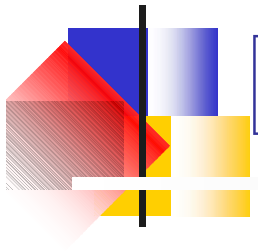
Classes that AppWizard generated

- CView (View Class)
  - Attached in Frame Window (as Child Window)
  - Presentation of Application Data (Display, Printer)
  - Combined with Document Class (Generally)
  - Many child class
    - CScrollView, CFormView, CEditView, CListView, CTreeView, …
  - OnInitialUpdate( )
  - OnDraw( )
  - GetDocument( )
  - OnUpdate( )



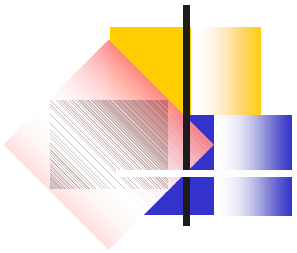
Classes that AppWizard generated

- CDocument (Document Class)
  - Management of Application Data
    - Read/Write, Input/Output
  - Combined with View Class
  - OnNewDocument( )
  - Serialize( )
  - DeleteContents( )
- Document/View Structure
  - Separation of data/display



Part 2

Basic Concepts

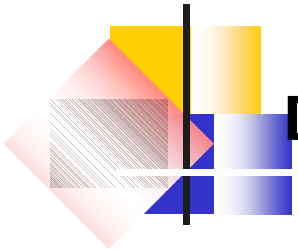


API and DLL

❖If we want to use C for windows programming then we require SDK .
❖ API : It has in build functions.
All internal working are through API.
It is used by O.S.

Disadvantages of API
❖ Size increase as a memory usage increase.
❖ Compilation time also increases

Functions in DLL are in executable form. It overcome all the disadvantages Of API
❖ They are loaded into memory.
❖ Independent from any application.

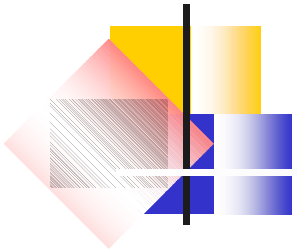


Device Context

It is an area of memory used to represent an image or an object on the screen by the window system.

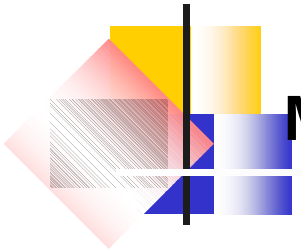
- The CDC is a special object where C represents classname and DC represents device context.
- All drawing or display of data that we do in windows take place in the device context.
- To draw an view object we use a device context corresponding to the view.
- To draw in Device Context we use CDC class methods such as

  CDC
  TextOut
  MoveTo
  Rectangle
  SetTextAlign
  SetTextColor



Continue..

- CDC TextOut() method is used to print in the view.
- pDC -> TextOut(0,0,string);
- CDC is the class name which is used to draw the string.
- pDC : It is a pointer of CDC class .It points to device context(Output window Where we represent the data.
- OnDraw() :It is used to display the program's client area, such as when our program first starts or when its window is closed and then reopened.

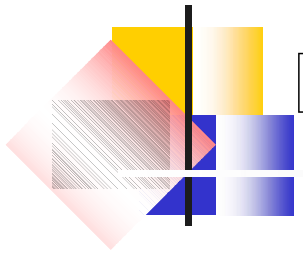


Microsoft Foundation Classes(MFC)

MFC library is a collection of C++ classes created by Microsoft. The baseclass for the class provided by MFC is CObject .It is the C++ class library. It Provides an object oriented wrapper around Windows's API.

- ❖Root class CObject
- ❖MFC application architecture classes.
- ❖Windows, dialog control classes.
- ❖Drawing and painting classes.
- ❖Simple data type classes.
- ❖Array . list and map classes
- ❖File and database classes.
- ❖Internet and networking classes.
- ❖Object linking and embedding(OLE).
- ❖Debugging and exceptional classes.

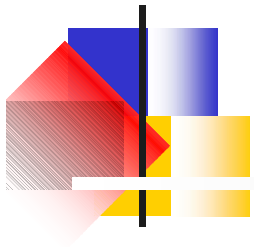
The MFC provides full source code in the form of header files and implementation files.



Hello, World! (Simple Example)

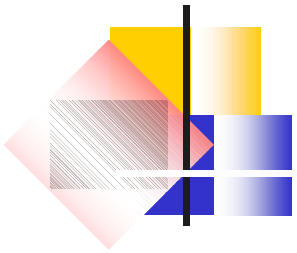
- Simple exercise for Usage of Class Wizard
- Modify CView::OnDraw()

```
pDC->TextOut(0, 0, “Hello, World!”);
```



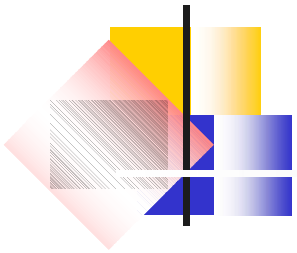
❑Part 3

Parts of Visual C++ program



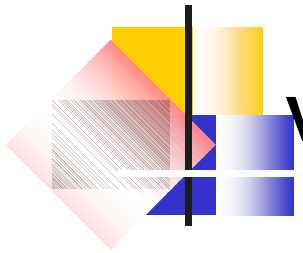
Application Object

- Functions of Application Object:-
- Start the program
- Launches main window on the screen
- Message Passing
- It is supported in .h and .cpp files.



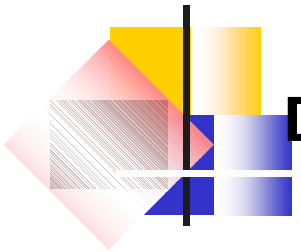
Main Window Object

- It display the programs.
- It handles everything except the client area
- Title bar
- Menu bar
- Tool bar
- Status bar



View Object

- It handles the client area
- It is really a window that appears on top of the client area.
- The data we display in the view object is stored in document object.



Document Object

- It stores the data for our program.
- The four parts of VC++ look like this.
- CFirstApp : Our windows program itself.
- CMainFrame: Our main window
- CFirstView : Handles data display.
- CFirstDoc : Handles our data.

Program :create an application in which user will type text from keyboard and will display that text in center of screen

Step1 :

Create a new SDI named center

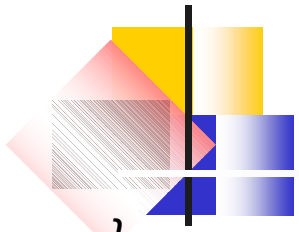
The four classes that will be created automatically are:

- CKeystrokes App
- CMainFrame
- CKeystrokesView
- CKeystrokesDoc

Step 2:

Go to file view -> header files->KeystrokesDoc.h(double click)

```
class KeystrokesDoc:: public CDocument
{
protected:
CKeystrokesDoc();
DECLARE_DYNCREATE(CenterDoc)
CString  StringData;
```



}

Step 3 :

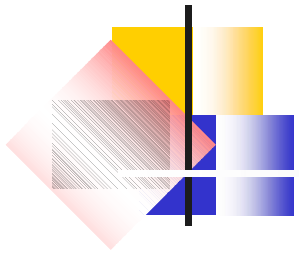
KeytrokesDoc.cpp

```
CkeystrokesDoc::CKeystrokesDoc()
{
StringData=“ “;
}
```

Step 4 :

go to view-> class wizard-> select
message maps-> select CKeystrokesView

```
void CKeystrokesView :: OnChar(UINT
nchar, UINT nflags,UINT nRepCnt)
{
CKeystrokesDoc * pDoc= GetDocument();
ASSERT_VALID(pDoc);
pDoc->StringData+=nchar;
Invalidate();
}
```



Step 5: CKeystrokesView::OnDraw(CDC* pDC)

```
{
        CKeystrokesDoc *pDoc = GetDocument();
        ASSERT_VALID(pDoc);
        pDC-> TextOut(0,0, pDoc-> StringData);
}
```