



# Visual Basic Programming

An Introduction

# Why Visual Basic?

- ★ Programming for the Windows User Interface is extremely complicated.
- ★ Other Graphical User Interfaces (GUI) are no better.
- ★ Visual Basic provides a convenient method for building user interfaces.
- ★ Visual Basic can interface with code written in C, for efficiency.



# What Visual Basic is not

- ★ Visual Basic is *not*, a powerful programming language that enables you to do anything you want.
- ★ Visual Basic is *not*, elegant or fast.
- ★ Visual Basic is *not*, a replacement for C.
- ★ Visual Basic is *not*, anything like any other programming language you have ever used.



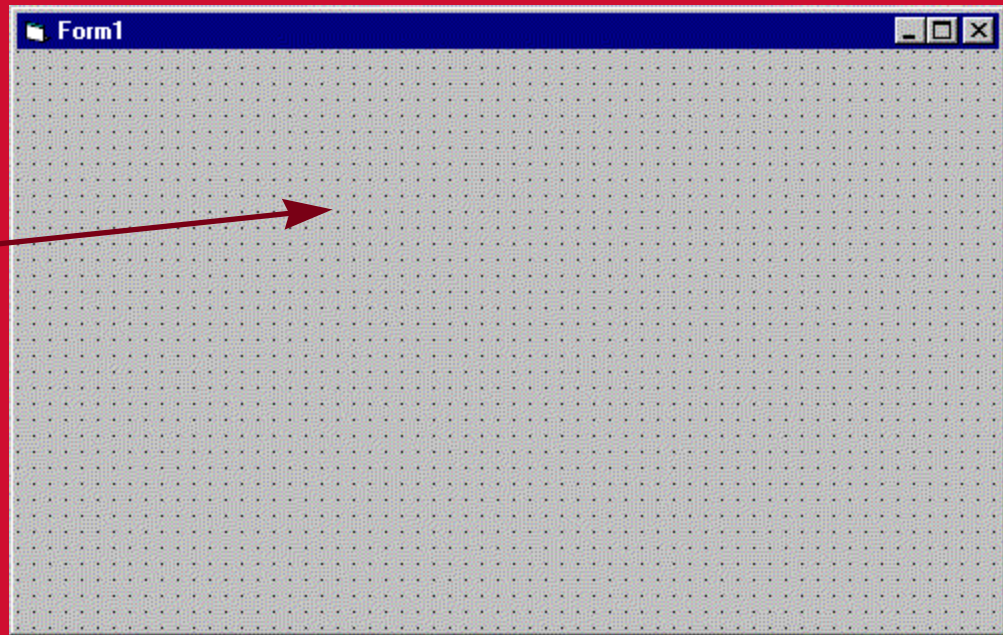
# When You Program in VB:

- ★ You draw pictures of your user interface.
- ★ You draw buttons, text boxes, and other user-interface items.
- ★ You add little snippets of code to handle the user interaction.
- ★ You add initialization code, usually as the last step.
- ★ If you like, you can code more complex functions. (But many do not.)



# The Visual Basic Interface

Draw Your  
Program  
Here!



# Drawing The Program

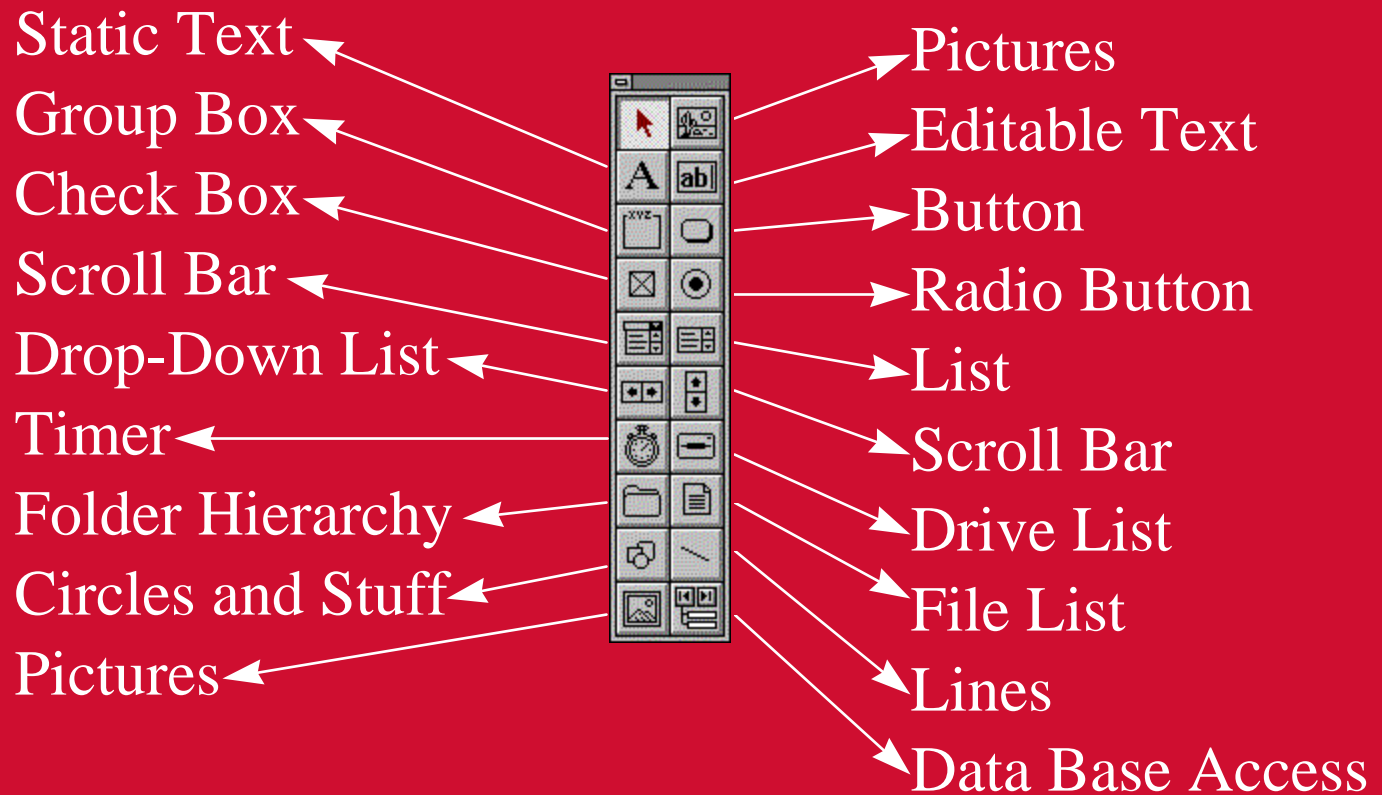
Select A Control From Here

(Click on the appropriate button)

Then Draw the control on the form

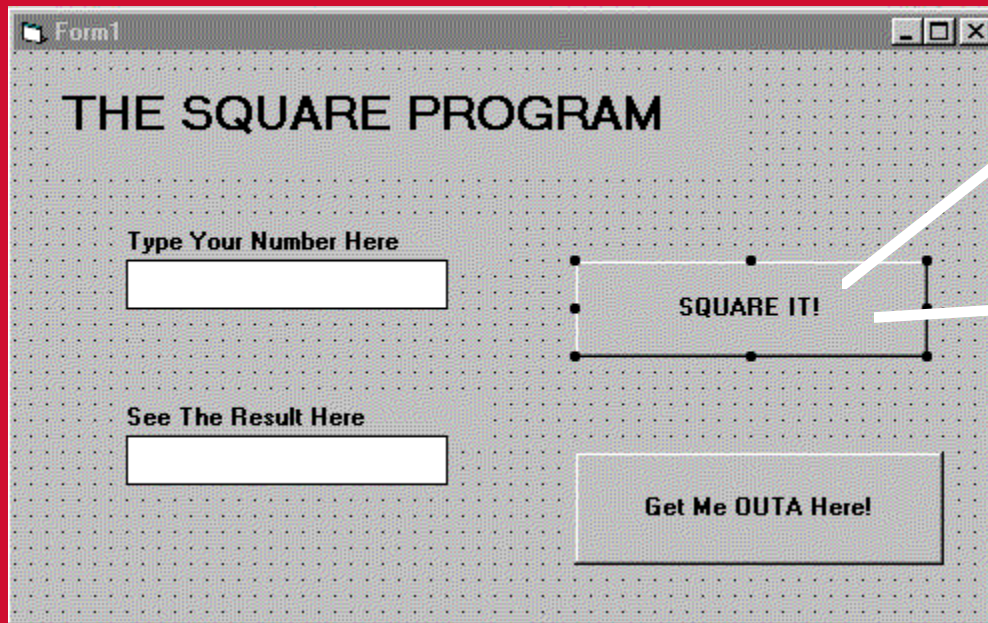


# Types of Controls



And the List Goes On and On ...

# A Simple Program



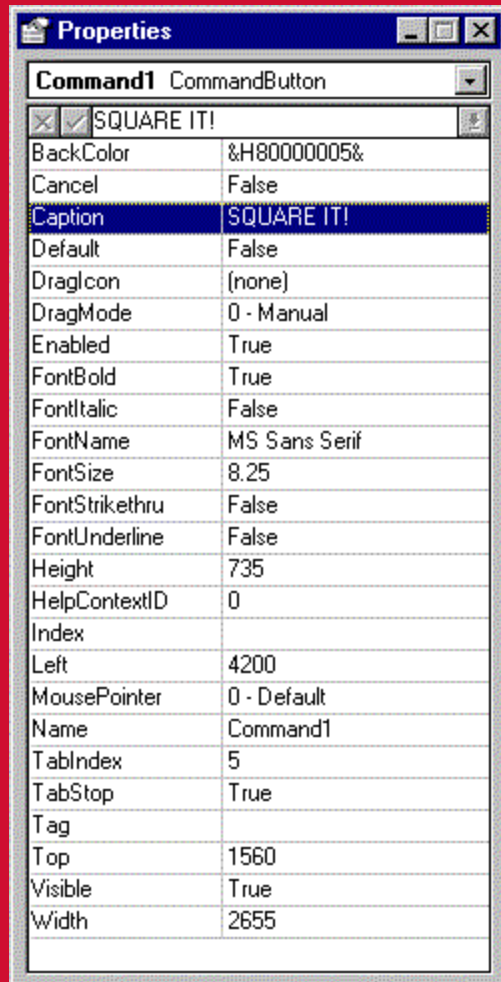
Double-Click to  
Add Code

Single-Click to  
Select and  
Change  
Properties

Using controls: Static Text  
Editable Text  
Buttons



# The Properties Window



List of Properties  
For Currently Selected  
Control

Click on Property, and  
Type In New Value, or  
Select New Value From  
Menu.

# Adding Code

Control  
Name

External Event  
Name

```
Form1.frm
Object: Command1 Proc: Click

Sub Command1_Click ()
  Dim Num As Integer

  Num = Val(Text1.Text)
  Num = Num * Num
  Text2.Text = Format$(Num)
End Sub
```

You must Write  
The Body  
Yourself

What to Do When It Happens

# More Complex Controls

## ★ Complex Controls Have:

- *Action* Properties to Execute Commands
- *Active* Properties that Cause Actions When Values Are Assigned to Them
- Many Types of Events for Program Interaction

## ★ Examples:

- Spreadsheets
- Word Processors
- Web Browsers



# Using C Code

- ★ Write a DLL in C
- ★ Use the `_export` Property on Appropriate Functions
- ★ Write Visual Basic Definitions for each Function
- ★ Add VB Definitions to The (*general*) section of the VB Program
- ★ Use Functions as if they were VB functions



# C Definition vs. VB Definition

**C:**

```
long FAR PASCAL _export HexToLong (char *Hex)
```

**VB:**

```
Declare Function HexToLong Lib "FIRSTONE.DLL"  
    (ByVal InString As String) As Long
```

Function Name Must Be The Same in Both Declarations.  
The Lib keyword Must Give The Name of the Library.  
Argument Name in VB is arbitrary.

# A (Very Annoying) Problem

- ★ It is sometimes difficult for VB to *FIND* the .DLL file.
- ★ If this occurs, copy the .DLL file to the WINDOWS directory.
- ★ Remember to Delete the file when you are done.



# Alternative Methods

- ★ Some Versions of VB do not allow DLL function definitions in the (*general*) section of a form.
- ★ To Get Around this Problem, Create a new Module (File Menu)
- ★ Add the declarations to the (*general*) section of the module
- ★ You can add your own VB functions to the (*general*) section of a form or a module.



# Syntax Considerations

- ★ All Functions are Global in VB
- ★ Variables are declared using the syntax:
  - Dim <Name> As <Type>
  - Every variable must have a type
  - Dim A,B,C As <Type> will work, but gives weird results
- ★ Most Common Types: Integer, String, Long





# More VB Syntax

## ★ Use Integers for Booleans

- As in C, 0 = False, everything else = True
- Symbolic constants True and False may be used
- True = -1, False = 0

## ★ Assignments are the same as in C

## ★ The Val function converts strings to integers

## ★ The Format\$ function converts integers to strings




# VB Statements

- ★ Assignments are the Same as in C
- ★ Case is not significant
  - Case will be adjusted for you on keywords
  - For Variable Names, Case is ignored
- ★ The Usual Operators can be used
  - AND is the same as both & and && depending on context
  - OR = | and ||
  - NOT = !



# VB IF Statements



```
If <condition> Then
    <List of Statements>
Else
    <List of Statements>
EndIf
```

```
If <condition> Then
    <List of Statements>
EndIf
```

Comparators: =, <, >, <=, >=, <> (not equal)

Connectives: And, Or, Not

**DON'T FORGET THE ENDIF!**

# VB While Statements

```
While <condition> do  
    <List of Statements>  
Wend
```

The VB Manual Recommends a different structure.  
Use the alternative if you wish.



# VB For Statements

```
For <Variable> = <start> to <finish>  
    <List of Statements>  
Next <Variable>
```

```
For <Variable> = <start> to <finish> Step <increment>  
    <List of Statements>  
Next <Variable>
```

Example:

```
For I = 1 to 10 do  
    A[I] = A[I] + 1  
Next I
```



# VB Arrays

- ★ Indices Always Start With Zero
- ★ *Dim A[10] As Integer* Declares 11 elements, indexed from 0 through 10.
- ★ Multi-Dimensional Arrays are Permitted.
- ★ Arrays can be resized at run time (See VB Help File for *ReDim*)



# VB Strings

- ★ Variable Length
- ★ Compare using standard comparators
- ★ Maximum length is about 64Kb
- ★ Minimum length is zero
- ★ Allocated from VB “String Space”, so may run out of space even on systems with much memory.

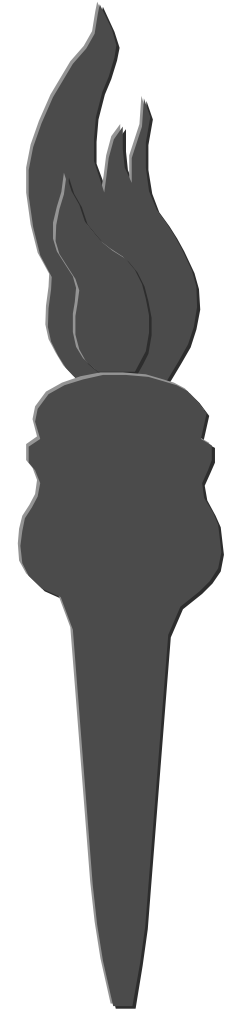


And in Conclusion ...

Go  
Have  
Fun!









# Visual Basic Programming

*An Introduction*







# Why Visual Basic?

- 
- ★ Programming for the Windows User Interface is extremely complicated.
  - ★ Other Graphical User Interfaces (GUI) are no better.
  - ★ Visual Basic provides a convenient method for building user interfaces.
  - ★ Visual Basic can interface with code written in C, for efficiency.
- 




# What Visual Basic is not

- 
- ★ Visual Basic is *not*, a powerful programming language that enables you to do anything you want.
  - ★ Visual Basic is *not*, elegant or fast.
  - ★ Visual Basic is *not*, a replacement for C.
  - ★ Visual Basic is *not*, anything like any other programming language you have ever used.
- 

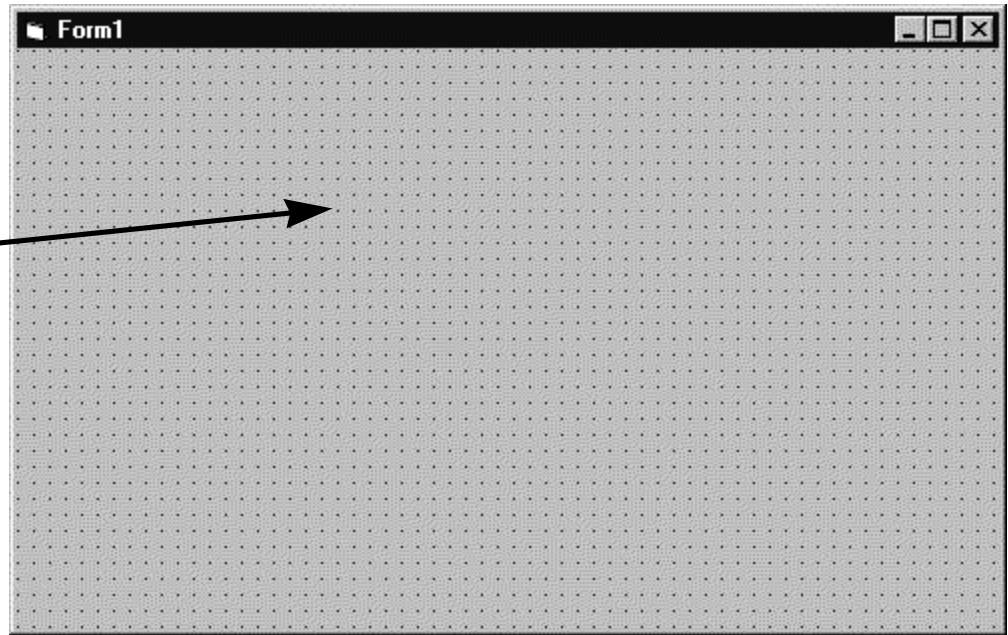


# When You Program in VB:

- ★ You draw pictures of your user interface.
  - ★ You draw buttons, text boxes, and other user-interface items.
  - ★ You add little snippets of code to handle the user interaction.
  - ★ You add initialization code, usually as the last step.
  - ★ If you like, you can code more complex functions. (But many do not.)
- 

# The Visual Basic Interface

Draw Your  
Program  
Here!



# Drawing The Program

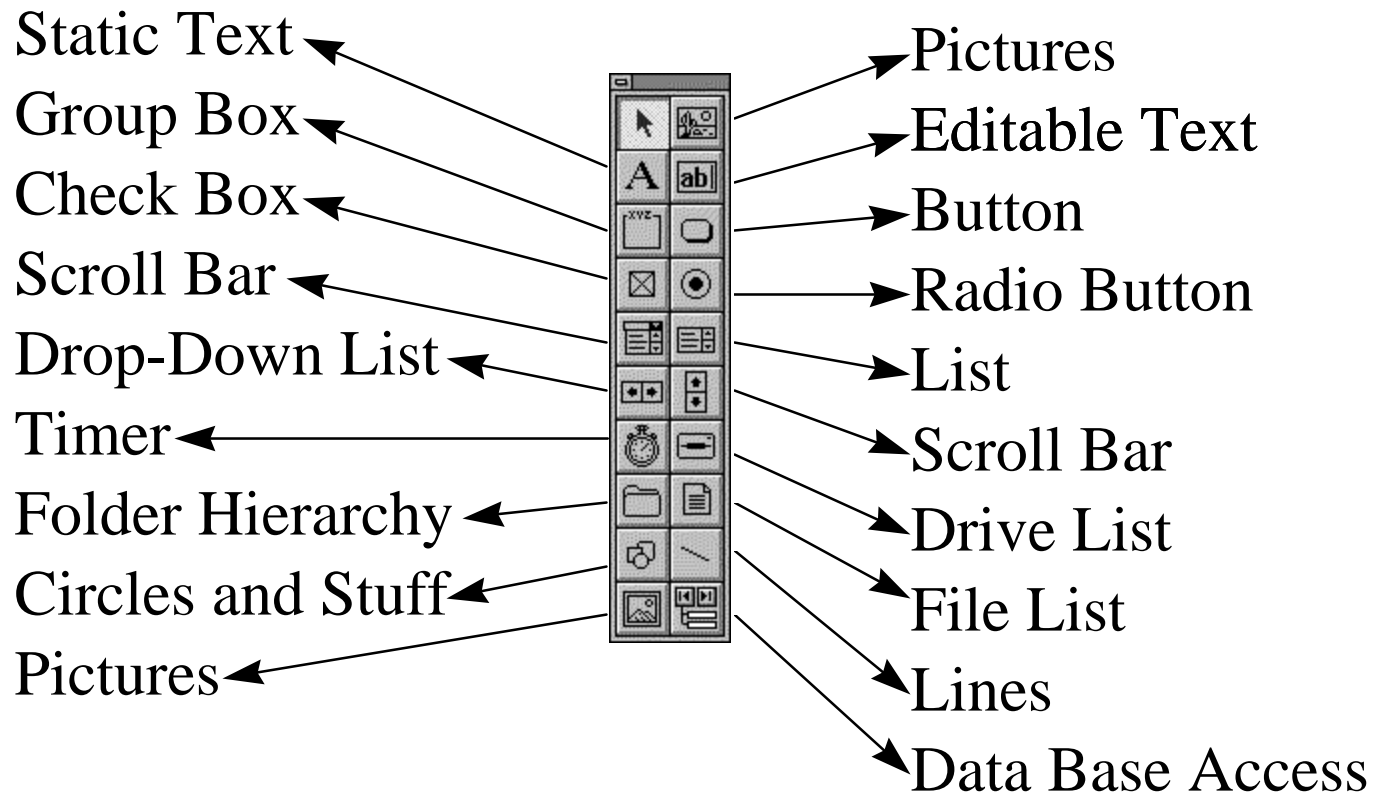
Select A Control From Here

(Click on the appropriate button)

Then Draw the control on the form

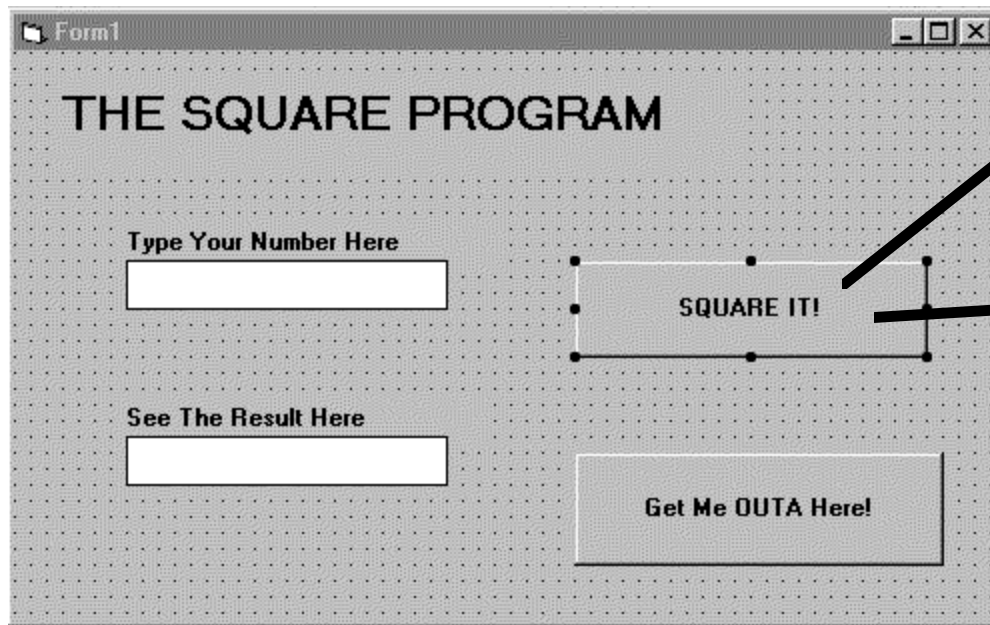


# Types of Controls



And the List Goes On and On ...

# A Simple Program



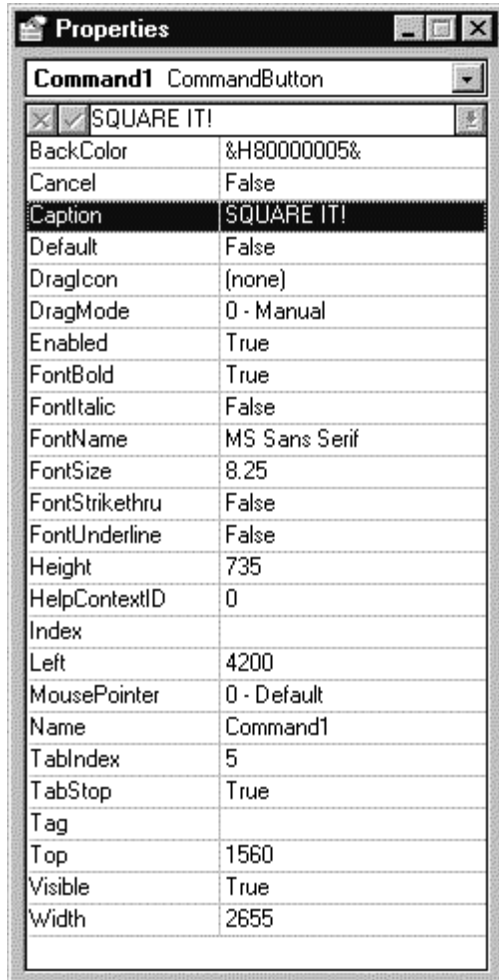
Double-Click to  
Add Code

Single-Click to  
Select and  
Change  
Properties

Using controls: Static Text  
Editable Text  
Buttons



# The Properties Window



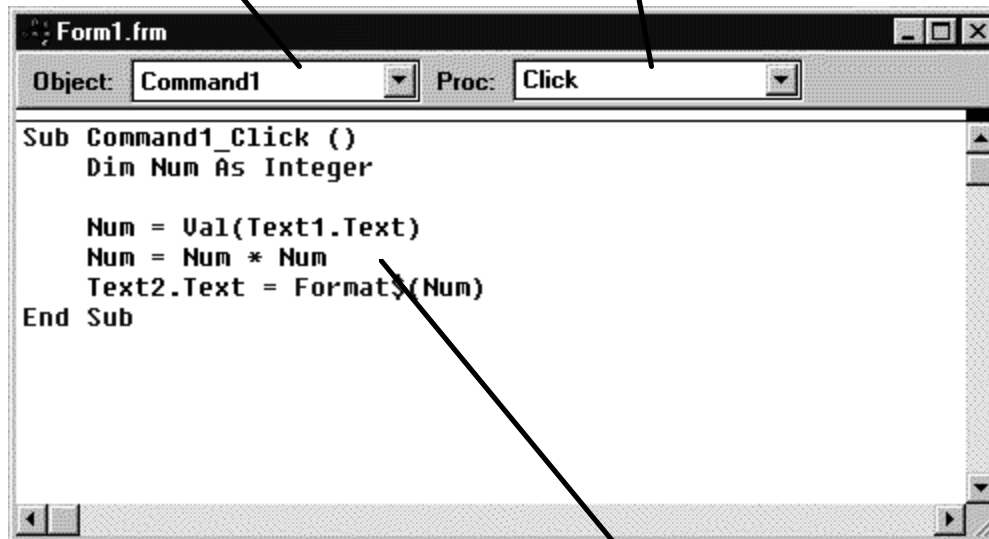
List of Properties  
For Currently Selected  
Control

Click on Property, and  
Type In New Value, or  
Select New Value From  
Menu.

# Adding Code

Control  
Name

External Event  
Name



```
Sub Command1_Click ()  
    Dim Num As Integer  
  
    Num = Val(Text1.Text)  
    Num = Num * Num  
    Text2.Text = Format$(Num)  
End Sub
```

You must Write  
The Body  
Yourself

What to Do When It Happens






# More Complex Controls



## ★ Complex Controls Have:


- *Action* Properties to Execute Commands
- *Active* Properties that Cause Actions When Values Are Assigned to Them
- Many Types of Events for Program Interaction

## ★ Examples:

- Spreadsheets
  - Word Processors
  - Web Browsers
- 



# Using C Code

- ★ Write a DLL in C
  - ★ Use the `_export` Property on Appropriate Functions
  - ★ Write Visual Basic Definitions for each Function
  - ★ Add VB Definitions to The (*general*) section of the VB Program
  - ★ Use Functions as if they were VB functions
- 



# C Definition vs. VB Definition

**C:**

```
long FAR PASCAL _export HexToLong (char *Hex)
```


**VB:**

```
Declare Function HexToLong Lib "FIRSTONE.DLL"  
    (ByVal InString As String) As Long
```

Function Name Must Be The Same in Both Declarations.  
The Lib keyword Must Give The Name of the Library.  
Argument Name in VB is arbitrary.




# A (Very Annoying) Problem

- 
- ★ It is sometimes difficult for VB to *FIND* the .DLL file.
  - ★ If this occurs, copy the .DLL file to the WINDOWS directory.
  - ★ Remember to Delete the file when you are done.






# Alternative Methods

- ★ Some Versions of VB do not allow DLL function definitions in the (*general*) section of a form.
  - ★ To Get Around this Problem, Create a new Module (File Menu)
  - ★ Add the declarations to the (*general*) section of the module
  - ★ You can add your own VB functions to the (*general*) section of a form or a module.
- 



# Syntax Considerations

- ★ All Functions are Global in VB
  - ★ Variables are declared using the syntax:
    - Dim <Name> As <Type>
    - Every variable must have a type
    - Dim A,B,C As <Type> will work, but gives weird results
  - ★ Most Common Types: Integer, String, Long
- 





# More VB Syntax



## ★ Use Integers for Booleans

- As in C, 0 = False, everything else = True
- Symbolic constants True and False may be used
- True = -1, False = 0


## ★ Assignments are the same as in C

## ★ The Val function converts strings to integers

## ★ The Format\$ function converts integers to strings





# VB Statements

- ★ Assignments are the Same as in C
  - ★ Case is not significant
    - Case will be adjusted for you on keywords
    - For Variable Names, Case is ignored
  - ★ The Usual Operators can be used
    - AND is the same as both & and && depending on context
    - OR = | and ||
    - NOT = !
- 



# VB IF Statements



```
If <condition> Then
    <List of Statements>
Else
    <List of Statements>
EndIf
```

```
If <condition> Then
    <List of Statements>
EndIf
```

Comparators: =, <, >, <=, >=, < > (not equal)

Connectives: And, Or, Not

**DON'T FORGET THE ENDIF!**





# VB While Statements





```
While <condition> do  
    <List of Statements>  
Wend
```

The VB Manual Recommends a different structure.  
Use the alternative if you wish.



# VB For Statements




```
For <Variable> = <start> to <finish>  
    <List of Statements>  
Next <Variable>
```

```
For <Variable> = <start> to <finish> Step <increment>  
    <List of Statements>  
Next <Variable>
```


Example:

```
For I = 1 to 10 do  
    A[I] = A[I] + 1  
Next I
```







# VB Arrays

- ★ Indices Always Start With Zero
  - ★ *Dim A[10] As Integer* Declares 11 elements, indexed from 0 through 10.
  - ★ Multi-Dimensional Arrays are Permitted.
  - ★ Arrays can be resized at run time (See VB Help File for *ReDim*)
- 



# VB Strings

- 
- ★ Variable Length
  - ★ Compare using standard comparators
  - ★ Maximum length is about 64Kb
  - ★ Minimum length is zero
  - ★ Allocated from VB “String Space”, so may run out of space even on systems with much memory.
- 



And in Conclusion ...



Go  
Have  
Fun!

