

## Computer/Human Interaction

### Lecture 9

#### Overview:

- Event-driven programming
- Visual Basic form designer
- Visual Basic event handlers

#### References:

- VBMS, VBCL

## Windowing Systems

- WIMP interface: Window, Icons, Menus, Pointers
- Abstract actual hardware into notions of screen, keyboard, and mouse
- Device driver translates abstract commands into actual hardware commands
- USB has become a meta-abstraction for many peripherals

## Window Management

- Several ways to organize
  - Each application could be responsible for its own windows. Not efficient.
  - Operating system manages windows. E.g. Windows, old MacOS
  - Separate management server, e.g. X Windows. Generic across operating systems, e.g. Unix, MacOS X.
- Generally, client-server architecture; often network-based

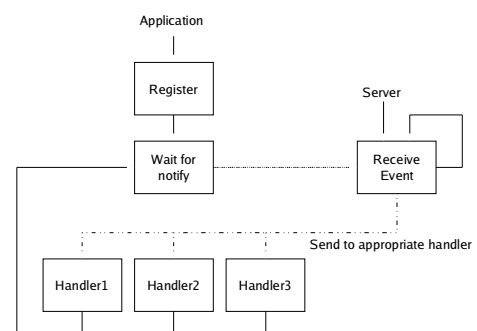
## Read-Eval Loop

- Read-evaluation loop - Win 3.1, old MacOS
  - Client maintains a queue of interaction events. Server notices events and places them on client (foreground process) queue.
  - Client handles events in queue order

```
Loop
  event = getEvent()
  Select on event.type
    Case type1: do type1 processing
    Case type2: do type2 processing
    :
  End Loop
```

## Event-Driven Processing

- Notification-based - Win, MacOS X, X, Java GUI programs
- Each program registers a handler for events it is interested in. E.g., keypress, mouse click, mouse move...
- Handlers also are called *call-back* functions



## Visual Basic

- Due to network security features, cannot run a program from the network drive at UE, so create projects on local (C:) drive.
- Starts with the default main form named `Form1` with programmer code stored in file named `Form1.vb`. Can change by changing file name.
- Also creates file `Form1.Designer.vb` that stores automatically generated designer code. Listed under `Form1.vb` in the solution tree.

## Getting Started

- Design interface first. Every GUI application has a base **form**
- Drop interface elements from a **toolbox** using the **form designer**. Pull on the handles to set size of element Drag to place element.
- Framework for adding handlers is automatically generated when interface element is double-clicked in the designer.

## Properties

- **Property** is anything that is static with respect to element creation.
- Appearance: `BackColor`, `ForeColor`, `BorderStyle`, `Font`, `Text`, `Scrollbars`, etc.
- Behavior: `Multiline`, `ReadOnly`, `TabIndex`, `TabStop`, etc.
- Design: (Name) - program variable name for the element, set before writing code
- Layout: `Size`, `Location`, etc. - usually set using form designer

## Form Elements

- Labels: text, no interaction
- Textboxes: single and multiline text, with or without scrollbars, text is `String` property
- Radio buttons: `checked` is `Boolean` property, all radio buttons tied together unless grouped, text is button label
- Check boxes: also has `checked` property, text is box label
- Command buttons: text is button label

## Adding Handlers

- Double-click on the element. Application is a class that inherits from `Form`. GUI elements are data members.
- Stub for handler is automatically generated as a private subroutine of the application class

```
Private Sub <element>_<event> _  
    (ByVal sender As System.Object, _  
     ByVal e As System.EventArgs) _  
        Handles <element>.<event>  
  
    'Handler code goes here  
End Sub
```

## Handlers, cont'd

- Assumed handlers when double-clicking element in designer
  - Command buttons - `Click`
  - Radio buttons - `ClickChanged`
  - Checkboxes - `ClickChanged`
  - Textboxes - `TextChanged`
  - Forms - `FormLoaded`
- All form elements are friends by default, so can access any property using dot notation

## Purple Pizza Parlor, v1

- Create a new VB project on local (C:) drive.
- Rename `Form1.vb` to `pizza1.vb`; save and close solution
- Download `pizza1.vb` and `pizza1.designer.vb` from course webpage into project folder overwriting the original files
- Open project; right-click on `pizza1.vb` in solution tree, select View Code

## VB coding notes

- Not case sensitive, auto-capitalization
- Comments start with ' (single quote)
- All statements must be "one" line. Use \_ (underscore) to continue on next line.
- Variable declarations are  
`Dim <name> As <type>`
- String concatenation operator is &
- No semicolons! :-)
- If - Then - [ Elseif - Then - Else ] - End If

## More VB notes

- Message boxes are a separate pop-up window for short messages. Default is OK only.
- `vbCrLf` is a defined constant VB string for carriage return and linefeed.
- Placing & in a Text property will cause the following character to become the keyboard hotkey for that element.
- `Me` is the application object reference.  
`Me.Dispose()` deletes the object, thus program terminates.

## In-class Exercise

- Add a phone number field
- Add another size choice
- Add other topping choices
- Add error checking to the submit button handler that pops up a message box when the name or address box is empty. The code to terminate a function and go back to wait state is:  
`Exit Sub`
- Copy `.vb` files to network drive!