Today’s Lecture

- Input of human factors
  - Learning styles
- C# Tips / Tricks
  - Necessity of Windows.Resources
- XML
  - Read / Write in C#
  - XAML-based XML parsing

Reminders
- Blog Post (Week)
- Homework 5
Where are we going?

- First half
  - WPF

- Second half
  - Surface projects
  - XML data
  - HCI – MIT 6.831 – Fall 2004 Variant
    - User – Psychology
      - Understanding the user
    - Business Functions
      - More on task analysis
    - Organization / Structure
      - How should things be laid out
Highlights

☐ Guest lectures
  ■ Nov 9 / 11
  ☐ Lockdown security visualization
    ■ Qi Liao, Dirk VanBruggen
  ☐ Observations from S/W industry
    ■ Prof. Mike Villano from Psychology

☐ Experimentation
  ■ Learning styles
  ■ Security terminology
Learning Styles

☐ Sensing Learner (S)
  - Focus on external input (see, hear)
  - Observant – notice details
  - Repetition / practice
  - Practical
  - Connection to the real world

☐ Intuitive Learner (I)
  - Imaginative
  - Meanings over details – fundamentals / theory
  - Work with concepts
  - Complaint: Too much plug & chug
Learning Styles

- Visual Learner (Vi)
  - Show me
    - Pictures, diagrams, sketches

- Verbal Learner (Vb)
  - Explain it to me
    - Written words
    - Symbols
    - Spoken word
Learning Styles

- Active Learner (A)
  - Physically interact with material
  - Think out loud
  - Let’s try it and see how it goes
  - Group work

- Reflective Learner (R)
  - Think about the material
  - Work introspectively
  - Like solo or pair work
  - Let’s think it through and then try
Learning Styles

- Sequential Learner (Sq)
  - Logical sequential steps
  - Work with partial information
  - Steady progress
  - Trees

- Global Learner (G)
  - Absorb randomly, synthesize big picture
  - Need big picture
  - Large leaps in understanding
  - Synthesis, holistic thinking (the forest)
Relate to HCI

☐ What are you?
☐ What are most people?
☐ How might this drive interface design?
C# Tips / Tricks

Q: Do I really have to use the Window.Resources to link in external classes as a source to an items?
- Why bother if I am just replacing it later?

A: You technically do not have to put it in there
- We did it for illustration
Example

<ListBox>
  <ListBox.ItemTemplate>
    <DataTemplate>
      <StackPanel>
        <TextBlock Text="{Binding Path=Name}"
      </StackPanel>
    </DataTemplate>
  </ListBox.ItemTemplate>
</ListBox>

Remember, XAML binding fails silently
<Grid x:Name="gridGadget" OnLoaded="loadGadget">
  <ListBox>
    <ListBox.ItemTemplate>
      <DataTemplate>
        <StackPanel>
          <TextBlock Text="{Binding Path=Name}"
        </StackPanel>
      </DataTemplate>
    </ListBox.ItemTemplate>
  </ListBox>
</Grid>
public void loadGadget (object sender, RoutedEventArgs theArgs)
{
    gridGadget.DataContext = MyObject;
}

Data Context works hierarchically like all XAML

Grid
ListBox
DataTemplate
StackPanel

Monday: Be bad with quasi-global objects
XML Paper Specification

- Why PowerPoint / Office 2007 docs come up as zip files
  - Because they really are
    - Text / descriptor
    - Binaries

- Already seen it too
  - FlowDocument
  - FixedDocument
Data, data, data

☐ Two aspects
  ■ How do I save my data?
  ■ How do I load my data?

☐ Previously
  ■ XMLTextReader – bare bones version
  ■ .NET full featured
    ☐ XMLReader
    ☐ XMLWriter
    ☐ Which to use?
      ■ Depends on size of file
      ■ Need for schema validation

■ XAML
How do we write?

- Assume we have a list of stuff
  - ListWiimoteValues

- Every card has properties / data
  - WiimoteValue.Xaccel
  - WiimoteValue.Yaccel
  - WiimoteValue.Zaccel
  - WiimoteValue.Time
Two parts

- Settings for the writer
  - XMLWriterSettings
  - Simple mods
    - Indent
    - How much to indent?
  - XMLWriter
    - How we actually write
Example Code

XmlWriterSettings settings = new XmlWriterSettings();
settings.Indent = true;
settings.IndentChars = ("    ");

try
{
    XmlWriter writer = XmlWriter.Create(sFile, settings);

    foreach (WiimoteValue theVal in this)
    {
        writer.WriteStartElement("WiimoteValue");
        writer.WriteAttributeString("Xaccel", theVal.Xaccel);
        writer.WriteAttributeString("Xaccel", theVal.Yaccel);
        writer.WriteAttributeString("Xaccel", theVal.Zaccel);
        writer.WriteAttributeString("Xaccel", theVal.Time);
        writer.WriteEndElement();
    }
}
What did we do?

- XMLWriterSettings
  - Affected how it gets written

- XMLWriter
  - Created a new file
  - Added elements one at a time
  - Added attributes

- Key points
  - Nest things as we see fit
<?xml version="1.0" encoding="utf-8"?>

<WiimoteVal XAccel="0" YAccel="0" ZAccel="0"
    Time="09-10-09 09:48:00.25" />

<WiimoteVal XAccel="0" YAccel="0.05" ZAccel="0"
    Time="09-10-09 09:48:00.45" />

<WiimoteVal XAccel="0" YAccel="0.01" ZAccel="0.02"
    Time="09-10-09 09:48:00.60" />

Loose listing of stuff
Can organize via “root” element
Example Code

XmlWriterSettings settings = new XmlWriterSettings();
settings.Indent = true;
settings.IndentChars = ("    ");

try
{
    XmlWriter writer = XmlWriter.Create(sFile, settings);

    Writer.WriteStartElement("WiiValList");

    foreach (WiimoteValue theVal in this)
    {
        writer.WriteStartElement("WiimoteValue");
        writer.WriteAttributeString("Xaccel", theVal.Xaccel);
        writer.WriteAttributeString("Yaccel", theVal.Yaccel);
        writer.WriteAttributeString("Zaccel", theVal.Zaccel);
        writer.WriteAttributeString("Time", theVal.Time);
        writer.WriteEndElement();
    }

    writer.WriteEndElement();
For the read example, we’ll use the loose list of stuff
Got it out, bring it in

☐ Reading in the XML
  ■ XMLReaderSettings
    ☐ Ignore comments
  ■ XMLReader
    ☐ Nearly identical to earlier code
<?xml version="1.0" encoding="utf-8"?>
<WiimoteVal XAccel="0" YAccel="0" ZAccel="0"
    Time="09-10-09 09:48:00.25" />
<WiimoteVal XAccel="0" YAccel="0.05" ZAccel="0"
    Time="09-10-09 09:48:00.45" />
<WiimoteVal XAccel="0" YAccel="0.01" ZAccel="0.02"
    Time="09-10-09 09:48:00.60" />

Convert this back into object
Settings / Creation Code

XmlReaderSettings settings = new XmlReaderSettings();
settings.ConformanceLevel = ConformanceLevel.Fragment;
settings.IgnoreWhitespace = true;
settings.IgnoreComments = true;

try
{
    XmlReader reader = XmlReader.Create(sFile, settings);

    while (reader.Read())
    {
        reader.MoveToElement();

        if(reader.Name == "WiimoteVal")
        {
        }
Not bad, how to make the object

- Do the standard, new / set
  - WiimoteVal theVal;
    theVal = new WiimoteVal();

    theVal.Xaccel = reader.GetAttribute("Xaccel");

- Make it a constructor
  - Really 😊
public WiimoteVal (XMLReader reader) 
{
    Xaccel = reader.GetAttribute("Xaccel");
    ...
}

// In the code
if(reader.Name == "WiimoteVal")
{
    WiimoteVal theVal;
    theVal = new WiimoteVal(reader);
    // Validation might be good here too
    Add(theVal);
}
Recap

☐ Input / output of our classes
  ■ ~ 40 lines of code
  ■ A bit trickier with nesting
    ☐ Track which level we are on as we go between elements / end of elements

☐ Other notes
  ■ Case sensitive
  ■ Always exception handle
Data Providers

- Data provider
  - External outside of a class
    - Chunk of memory
    - RSS feed
  - XML
    - XmlDataProvider
    - XPath
      - Query language for XML

```xml
<Window.Resources>
  <XmlDataProvider x:Key="DataProvider"
                   XPath="GameStats" Source="GameStats.xml" />
</Window.Resources>
```
<Window.Resources>
    <XmlDataProvider x:Key="DataProvider"
        XPath="GameStats">
        <x:XData>
            <GameStats xmlns="">
                <GameStat Type="Beginner">
                    <HighScore>1203</HighScore>
                </GameStat>
                <GameStat Type="Intermediate">
                    <HighScore>1089</HighScore>
                </GameStat>
                <GameStat Type="Advanced">
                    <HighScore>541</HighScore>
                </GameStat>
            </GameStats>
        </x:XData>
    </XmlDataProvider>
</Window.Resources>
Add each HighScore element to the list box as an item

```xml
<ListBox ItemsSource="{Binding Source={StaticResource dataProvider}, XPath=GameStat/HighScore}" />
```
Add each HighScore element to the list box as an item
<Window.Resources>
   <HierarchicalDataTemplate DataType="GameStats"
      ItemSource="{Binding XPath=*}">
      <TextBlock FontStyle="Italic"
         Text="All Game Stats" />
   </HierarchicalDataTemplate>
   <HierarchicalDataTemplate DataType="GameStat"
      ItemSource="{Binding XPath=*}">
      <TextBlock FontWeight="Bold" FontSize="20"
         Text="{Binding Xpath=@Type}" />
   </HierarchicalDataTemplate>
   <DataTemplate DataType="HighScore">
      <TextBlock Foreground="Blue"
         Text="{Binding XPath=.}" />
   </DataTemplate>
</Window.Resources>
Hierarchical is for special controls
Menu
TreeView
Expander

Filter = *

Element Name

HierarchicalDataTemplate
DataType="GameStats"
ItemSource="{Binding XPath=*}">
<TextBlock FontStyle="Italic"
Text="All Game Stats"/>
</HierarchicalDataTemplate>
HOP ON THE ELEMENT TRAIN

<HierarchicalDataTemplate DataType="GameStat"
    ItemSource="{Binding XPath=*}"
    >
    <TextBlock FontWeight="Bold" FontSize="20"
        Text="{Binding Xpath=@Type}"
    />
    </HierarchicalDataTemplate>

Filter = *

Attribute Value

GameStat is nested in GameStat
And the caboose

<DataTemplate DataType="HighScore">
  <TextBlock Foreground="Blue"
    Text="{Binding XPath= . }" />
</DataTemplate>

Content, bind to thyself

DataTemplate when you have nothing more underneath that is nested

Could place a container instead of just one control -> StackPanel, Grid, etc. Will always fall through until success
Final Thoughts

☐ Lots to digest
  ■ Look for more examples

☐ Binding can be external
  ■ XmlDataProvider as a RSS feed
Questions?

- Weekly Blog
- Homework 5 (Posted Wed)