MessageBox

MessageBox a convenience dialog class  << see MBHelloWorld.cs >>

class MBHelloWorld { // from C. Petzold, 2001
    public static void Main()
    {
    }
}

DialogResult Show (string str)
    (... string title)
    (... MessageBoxButtons mbb)
    (... MessageBoxIcon mbi)
    (... MessageBoxDefaultButton mbdb)
    (... MessageBoxOptions mbo)

Enumerations are used for results and to set display arguments options
Show returns DialogResult enumeration {0..7}
None, Ok, Cancel, Abort, Retry, Ignore, Yes, No

Enumerations are used for MessageBoxButtons
OK, OKCancel, AbortRetryIgnore, YesNoCancel, YesNo, RetryCancel
Class Hierarchy

- Object
- MarshalByRefObject
- Component
- Control
- ScrollableControl
- ContainerControl
- Form

Object

remote object reference across application domains (IPC)

components

visual components, size, UI events

scroll capability

Container, focus & component control

Basic container for apps & dialogs
When forms are the base container for an application use:

```csharp
Application.Run(new aForm());
```

Application is a static class (like MessageBox).

Run takes a form object, makes it visible, and passes control to Window's message loop.

The form class can post a "quit" message to terminate the program.

FormBorderStyle property (enumeration)

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>none</td>
</tr>
<tr>
<td>1</td>
<td>FixedSingle</td>
</tr>
<tr>
<td>2</td>
<td>Fixed3D</td>
</tr>
<tr>
<td>3</td>
<td>FixedDialog</td>
</tr>
<tr>
<td>4</td>
<td>Sizable (default)</td>
</tr>
<tr>
<td>5</td>
<td>FixedToolWindow</td>
</tr>
<tr>
<td>6</td>
<td>SizableToolWindow</td>
</tr>
</tbody>
</table>

smaller caption bar, no control box
Example Form Properties  often set by Designer

- **Text**  
  title

- **ForeColor**  
  foreground color

- **BackColor**  
  background  -- also BackgroundImage

- **Width, Height**  
  size

- **Cursor**  
  see Cursors class for static values

- **StartPosition**  

- **Anchor**  
  Bottom, Left, None, Top, Right (Top, Left default)

- **Bounds**  
  location and size of control

- **ClientSize**  
  Area of client (! TitleBar and Borders)

- **ClientRect**  
  Area of client (! title, menu, borders, scrollbars)

- **Icon**  
  image used in task bar and control box

- **Parent**  
  The container control of the current control

- **Owner**  
  Owned forms min/max w/ & on top of Owner

- **Location**  
  position of origin
Events via Painting

Form class has delegate as a property

```csharp
public delegate void PaintEventHandler(object obj,
                                        PaintEventArgs pea);
```

In a subclass of Form class add the handler to the Paint property

```csharp
aForm.Paint += new PaintEventHandler(aPaintHandler);
```

and write the handler

```csharp
static void aPaintHandler(object obj, PaintEventArgs pea)
{
    Graphics g = pea.Graphics;
    g.DrawString("Hi CS 432!", Font, Brushes.Black,20,20);
}
```

Paint events occur on open, resize, minimize, restore, exposure

To explicitly cause a Paint event use

```csharp
aForm.Invalidate();
```
inherit event handlers

Inherit Form's OnPaint (a virtual protected method)
takes 1 argument – PaintEventArgs

OnPaint is called when a Control is invalid (a paint event is raised)

The OnPaint method in Control is overridden to call HiAgain's OnPaint.

class HiAgain : Form {
    public static void Main() {
        Application.Run(new HiAgain());
    }

    protected override void OnPaint(PaintEventArgs pea) {
        Graphics g = pea.Graphics;
        g.DrawString("Hi again", Font, Brushes.Black,
                      40, 40);
    }
}
OnResize method is invoked when the form's window is resized. This method is often overridden on overridden methods you often need to call the base class's method first, the keyword "base" is used (like java's super)

```csharp
protected override void OnResize(EventArgs ea) {
    base.OnResize(ea);
    // do class specific resize behaviors
    ...
}
```

The property ResizeRedraw can be set true to force the entire client area to be invalidated on resize events.

To force repainting of the form after an invalidate call Update().

```csharp
    ... 
    Invalidate(); // posts paint event  
    Update(); // executes pending paint events
```
Timer class

Send an event after counting down for an interval

Time methods
  Timer timer = new Timer ();
  void Start ();
  void Stop ();

Timer Properties

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>operations</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>Interval</td>
<td>get / set</td>
<td>tick time in msecs default 100</td>
</tr>
<tr>
<td>bool</td>
<td>Enabled</td>
<td>get / set</td>
<td>true when timer is running</td>
</tr>
</tbody>
</table>

Time event

<table>
<thead>
<tr>
<th>event</th>
<th>method</th>
<th>delegate</th>
<th>arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tick</td>
<td>OnTick</td>
<td>EventHandler</td>
<td>EventArgs</td>
</tr>
</tbody>
</table>

<< see RandomRectangle.cs >>
Common Dialogs

Component

- CommonDialog (abstract)
- colorDialog
- FileDialog (abstract)
- OpenFileDialog
- SaveFileDialog
- FontDialog
- PageSetupDialog
- PrintDialog

Use Static show methods
Can add to controls using Designer
Use properties to get values

Open dialogs in event handler method

```csharp
DialogResult dr = aDialog.ShowDialog();
if (dr == DialogResult.OK)
    // get property of
    // aDialog
```
ColorDialog
Properties

Color   Color   get / set
bool   FullOpen   get / set   false   open wide custom color
bool   AllowFullOpen   get / set   true   allow custom color open
bool   SolidColorOnly   get / set   false
bool   AnyColor   get / set   false
bool   ShowHelp   get / set   false
int[]   CustomColors   get / set

![ColorDialog Screenshot]
FileDialog

An abstract class with properties shared by Open and Save

FileDialog properties

string FileName get / set "" filename set by dialog
string[] FileNames get / set "" array of filenames (save ?)
string Filter get / set
bool AddExtension get / set true add extension from Filter ?
string InitialDirectory get / set ""
bool RestoreDirectory get / set restore current directory ?
bool CheckPathExists get / set true check before closing ?
bool CheckFileExists get / set true check before closing ?

dialogFilter = "Source files | *.cpp;*.cc; *.c" + "Header files | *.h ";

filter displays string up to "|" in filter combo box
string to the right of "|" is the actual extensions separated with ";"; terminated with "|
Designer doesn't put filter string in "quotes", designer will add "quotes"
OpenFileDialog properties

bool Multiselect get / set
bool ShowReadOnly get / set
bool ReadOnlyChecked get / set

for user to create new file with OpenFileDialog set CheckFileExists to false.

SaveFileDialog properties

bool CreatePrompt get / set false set true to prompt
bool OverwritePrompt get / set true ask for confirmation

SaveFileDialog will append the first extension listed in the Filter property filter line that user selects in "Save As Type" combo box.
Font Dialog

Properties

Font    Font    get / set
Color   Color   get / set
bool    ShowEffects    get / set    true    underline, strikeout ?
bool    ShowColor      get / set    false    combo box for color
bool    ShowApply      get / set    false    apply button ?
bool    ShowHelp       get / set    false    help button ?
bool    FixedPitchOnly get / set    false

FontDialog events

Apply    OnApply    EventHandler    EventArgs
HelpRequest    OnHelpRequest    EventHandler    EventArgs

Apply button semantics forces application to "apply" effects of dialog before the ShowDialog method returns. (see Petzold example pg 766)
Simple font usage

Control Properties related to fonts

Font Font get / set font
Font DefaultFont get font
int FontHeight get / set height

FontFamily class is indentified by string ie. "Times New Roman"

Font is a combination of a font family (object or string identifier), attributes (bold, italic) and a point size

Font (Font font, FontStyle fs);
Font (string family, float points);
Font (string family, float points, FontStyle fs);

FontStyle enumeration {Regular, Bold, Italic, Underline, Strinkout }
Custom Dialogs

<< see AuthorDialog.cs >>

Visual Studio .NET
Project | add Windows Form  eg: Author.cs
Use Toolbox to add controls to new form  // PictureBox, Button, Labels

In owner (application) form class source file (Hello585.cs)
declare object of new form's Class source file

private Author author;

in "initator's" handler  // usually a menu item's handler

private void authorMenuMenuItem_Click(object sender,
                                     EventArgs e) {
    author.Show(); }  // for modal use ShowDialog()
ShowDialog

Display or hide a Form (good for dialogs)

```csharp
    aForm.Show();  // or modeless dialog -- void
    aForm.Hide();

    aForm.Visible = true;  // or false
```

```csharp
    DialogResult ShowDialog();  // modal
    DialogResult ShowDialog (IWin32Window owner);
```

Based on DialogResult in application's source get dialog's properties if desired or write accessor methods in dialog class to return properties of desired controls.
Menus

Component

Menu (abstract)

MainMenu

ContextMenu

MenuItem

MainMenu is the menu bar typically below the Form's title.

ContextMenu is a pop up menu, often shown at the mousepoint with a right click.

Conceptually the MainMenu is a set of MenuItems that each contain a set (sub menu of) MenuItems...

Menus can be easily designed and implemented with Designer.

<< do demo >>

menu icons not done inside designer
Menus by hand

Constructors

MainMenu ();
MainMenu (MenuItem[] menuItems);

ContextMenu ();
ContextMenu (MenuItem[] menuItems);

MenuItem ();
MenuItem (string str);
MenuItem (... Event Handler eventHandlerName);
MenuItem (... ShortCut shortCutEnumeration);
MenuItem (string str, MenuItem[] menuItems);

Form Properties the menu for a Form

MainMenu Menu get/set

Control Properties the menu for a Control

ContextMenu ContextMenu get/set
// probably in a constructor ....
// create menu items for a File menu
MenuItem Open = new MenuItem("&Open...",
    new EventHandler(doFileOpen), Shortcut.Ctrl0);
MenuItem Save = new MenuItem("&Save",
    new EventHandler(doFileSave), Shortcut.CtrlS);
MenuItem SaveAs = new MenuItem("Save &As...",
    new EventHandler(doFileSaveAs));
MenuItem miDash = new MenuItem("-");
MenuItem miExit = new MenuItem("E&xit",
    new EventHandler(doExit));
// create the File menu
MenuItem miFile = new MenuItem("&File",
    new MenuItem[] {miOpen, miSave, miSaveAs, miDash, miExit})
// repeat for remaining menuItems of each menu
...

<< see menu.cs >>
Parent - Child

Form (or any container control) can have child controls. The child is shown within the boundaries of the parent.

Properties

Control.ControlCollection Controls get

Methods

void Add (Control aControl);
void AddRange (Control[] aControl);
void Remove (Control aControl);
void RemoveAt (int index);
bool Contains (Control aControl);
int getChildIndex (Control aControl);
void setChildIndex (Control aControl, int index);

as child controls are implicitly added to a form with
   aControl.Parent = theForm;

the form's Control.Count property is incremented as controls are added.
Managing child controls

Geometry Management: Position or Constraint Hierarchy

Position: controls have Location & Size properties

    in resize event handler (or override OnResize)

    Use variables, arrays, expressions to set control's Location and Size
    get parent's clientRect (or Location and ClientSize)
    recompute component's position.

Constraint Hierarchy

    dock {Top, Left, Fill, Right, Bottom, None} how control's size
    responds to parent's resize.

    anchor {Top, Left, Right, Bottom, None } default Top|Left
    how control positions relative to sides of the form.

    when anchored to opposite sides control's size changes
Containment Heirarchy

<< see contentHierarch.cs >>

Form1

statusBar - d = B, a = T | L
bottomPanel - d = F, a = T | L
leftPanel - d = L, a = T | L
buttonPanel - d = N, a = L | R
redButton - d = N, a = T | L | R
greenButton - d = N, a = L | R
blueButton - d = N, a = B | L | R
fillPanel - d = N, a = T | B | L | R

Alternately create a panel just to scroll its child panel (Grid.cs).
Scrolling

ScrollableControl get/set properties

bool  AutoScroll, HScroll, VScroll
Size   AutoScrollMargin, AutoScrollMinSize, AutoScrollPosition

Scrolling in a panel

Panel is a control used to contain other controls or to paint into

public aFormConstructor() {
    ...
    AutoScroll = true;
    ... // create a panel
    Panel panel = new Panel();
    panel.Parent = this; // adds the panel to the form
    panel.Paint += new PaintEventHandler(PanelOnPaint)
    panel.Size = new Size (...);
    ...
}

void PanelOnPaint(object obj, PaintEventArgs pea) {
    ... // do all the painting operations into the panel
}
Scroll in the Form

In the constructor
    set AutoScroll true
    set AutoScrollMinSize to the necessary size for all controls

In OnPaint
    perform all drawing requests

This is inefficient

    often drawing into an area that is clipped (by the ClientArea).

or, need to calculate what part of form is displayed with
    AutoScrollPosition and then draw only parts visible (pg 160 - 161)

    The ClipRectangle property of the PaintEventArgs returns the
    smallest Rectangle in client area that is the invalid region.
Control Auto Scroll

Properties

<table>
<thead>
<tr>
<th>Type</th>
<th>Property</th>
<th>get / set</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td>AutoScroll</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>Size</td>
<td>AutoScrollMargin</td>
<td></td>
<td>0, 0</td>
</tr>
<tr>
<td>Size</td>
<td>AutoScrollMinSize</td>
<td></td>
<td>0, 0</td>
</tr>
<tr>
<td>Point</td>
<td>AutoScrollPosition</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Margin is between contained controls and scrollable parent

MinSize of the scrollable control.
Scroll bars visible when ClientSize < MinSize

AutoScrollPosition used to adjust display of child controls in scrolled parent.

As scroll thumb is moved right or down AutoScrollPosition.X or Y increases as a negative offset. 0 to -(AutoScrollMinSize - ClientSize)

Also see panel in panel scroll example – using containment hierarchy for transparent scrolling.
Mouse

Mouse events

- mouse: input device
- mouse cursor: on screen pointer
- hotspot: actual active (selection) location of mouse pointer

SystemInformation properties

<table>
<thead>
<tr>
<th>Type</th>
<th>Property</th>
<th>Get</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>DoubleClickTime</td>
<td>get</td>
<td>msec</td>
</tr>
<tr>
<td>Size</td>
<td>DoubleClickSize</td>
<td>get</td>
<td>area in pixel</td>
</tr>
<tr>
<td>bool</td>
<td>MousePresent</td>
<td>get</td>
<td></td>
</tr>
<tr>
<td>int</td>
<td>MouseButtons</td>
<td>get</td>
<td>1 to 5</td>
</tr>
<tr>
<td>bool</td>
<td>MouseButtonsSwapped</td>
<td>get</td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td>MouseWheelPresent</td>
<td>get</td>
<td></td>
</tr>
<tr>
<td>int</td>
<td>MouseWheelScrollLines</td>
<td>get</td>
<td>lines / scroll turn</td>
</tr>
</tbody>
</table>
Mouse events

<table>
<thead>
<tr>
<th>Event</th>
<th>Method</th>
<th>Delegate</th>
<th>Argument</th>
</tr>
</thead>
<tbody>
<tr>
<td>MouseDown</td>
<td>OnMouseDown</td>
<td>MouseEventArgsHandler</td>
<td>MouseEventArgs</td>
</tr>
<tr>
<td>MouseUp</td>
<td>OnMouseUp</td>
<td>MouseEventArgsHandler</td>
<td>MouseEventArgs</td>
</tr>
<tr>
<td>MouseMove</td>
<td>OnMouseMove</td>
<td>MouseEventArgsHandler</td>
<td>MouseEventArgs</td>
</tr>
<tr>
<td>MouseWheel</td>
<td>OnMouseWheel</td>
<td>MouseEventArgsHandler</td>
<td>MouseEventArgs</td>
</tr>
</tbody>
</table>

MouseEventArgs properties

- int X get location in client area
- int Y get
- MouseButtons Button get mouse buttons used
- int Clicks get 2 for double clicks
- int Delta get mouse wheel movement
  - 1 click is ≈ ± 120

MouseButton enumeration (bit flags, can be combined "and")

None, Left, Right, Middle, XButton1, XButton2

(IntelliMouse Explorer has 5 buttons)
Mouse events (Control events) continued

<table>
<thead>
<tr>
<th>Event</th>
<th>Method</th>
<th>Delegate</th>
<th>Argument</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click</td>
<td>OnClick</td>
<td>EventHandler</td>
<td>EventArgs</td>
</tr>
<tr>
<td>DoubleClick</td>
<td>OnDoubleClick</td>
<td>EventHandler</td>
<td>EventArgs</td>
</tr>
<tr>
<td>MouseEnter</td>
<td>OnMouseEnter</td>
<td>EventHandler</td>
<td>EventArgs</td>
</tr>
<tr>
<td>MouseLeave</td>
<td>OnMouseLeave</td>
<td>EventHandler</td>
<td>EventArgs</td>
</tr>
<tr>
<td>MouseHover</td>
<td>OnMouseHover</td>
<td>EventHandler</td>
<td>EventArgs</td>
</tr>
</tbody>
</table>

EventArgs does not have any mouse specific information.

Control class does have properties

Point MousePosition get location in screen coordinates
MouseButtons MouseButtons get buttons currently pressed
Keys ModifierKeys get status of Shift, Cntrl, Alt keys

```csharp
if (MouseButtons == MouseButtons.Left &&
aControl.ModifierKeys == Keys.Shift)
```

Use PointToClient to convert screen coordinates to client area coordinates.

```csharp
aControl.PointToClient(aControl.MousePosition);
```
Mouse tracking

Three events: `mouseDown`, `mouseMove`, and `mouseUp` are used to track mouse movement events.

`mouseDown`, `OnMouseDown` set up the state (or store) to record mouse move events

`mouseMove`, `OnMouseMove` store information on movement (usually X, Y or Points)

if drawing is done need to

```csharp
Graphics g = CreateGraphics ();
// do drawing
g.Dispose ();
```

`mouseUp`, `OnMouseUp` stop or unset the state for recording mouse move events use `Invalidate ()` to have `OnPaint (...)` called.
Cursor and Cursors class

Static Cursors properties can be used to set the control's cursor shape

Cursor properties

Cursor       Current       get/set       set the cursor
Point        Position      get/set       position the cursor !advised
Rectangle    Clip          get/set

Cursor methods      void Show ();  void Hide ();

To set the cursor its better to use the Control's Cursor property

\[
a\text{Control}.\text{Cursor} = \text{Cursors}.\text{Hand};
\]

ContextMenu Show (menu property set with Designer)

(pop ups) are initiated with mouse events and displayed with

\[
\text{Show}(\text{Control } c, \text{ Point } p); \quad \text{Control is } "\text{associated}" \text{ with menu}
\]

private void contextMenu_Popup(object s, EventArgs e) {
    if (MouseButtons == MouseButtons.Right)
        contextMenu.Show(this, MousePosition);
}
Point, a 2D coordinate point, has two properties X and Y

```csharp
Point pt = new Point(30, 20);
Point pt2 = new Point(); pt2.X = 40; pt2.Y = 60;
Console.WriteLine(pt2);
if (pt == pt2) ... // same as if (pt.Equals(pt2)) ...

Point[] pts4 = { new Point(10, 10), new Point(20, 10),
                 new Point(20, 20), new Point(10, 20); }
```

Size has Width and Height properties.

sizes and points can be cast into each other and used to construct each other.
"+" and "−" operators are overridden for both points and sizes

There are also float versions: PointF and SizeF.
Form and Client

Client area is the internal form area (excludes caption bar, borders, menu bar, scroll bars)

ClientSize property returns the Size value

ClientRectRectangle property returns a Rectangle value

Can't set a property of a property

ClientSize.Width += 100;  // doesn't work

ClientSize += new Size(100,0);  // works

Drawing in the Client (outside of an OnPaint, PaintHandler or constructor method)

...  
Graphics g = CreateGraphics();
// drawing statements here
  g.Dispose();
...
Rectangular struct is a point (origin) with a size (extent). Also RectangleF

```csharp
    Rectangle(Point p, Size s);
    Rectangel(int x, int y, int w, int h);
```

Rectangle properties:
- get/set Point, Size, X, Y, Width, Height,
- get Left, Top, Right, Bottom, isEmpty
  - Left is X, Top is Y
  - Right is X + Width, Bottom is Y + Height

Operators are:  ==  !=
Equals(Rectangle)
Color

Color struct represents color with ARGB (alpha is transparency level).
has 141 static color names properties

Pen class used for drawing lines, curves
Brush class used for drawing filled areas and text

Pens and Brushes classes have 141 static read only color properties that are easier usage than Pen and Brush.

SystemColors class has 26 read only properties for most (26 / 29) user settable colors in the Windows interface.

Window (back), WindowText(fore), Control, ControlText, ...

SystemPens and SystemBrushes have 15 and 21 of these properties

KnownColors enumeration has all color names and system colors

Use SystemColors values if you want user settable interface colors to be used in your application.
Math class

constants: Math.PI and Math.E
transcendental functions (sin, cos, ...)
double Math.Sin(double radians);
methods: Abs, Floor, Round, Ceiling, Exp, Pow ...

Random class

using System;
class RandomDemo {
    [STAThread] // single threaded application
    static void Main(string[] args) {
        Random r = new Random(); // use time dependent seed
        for(int i = 0; i < 10; i++)
        {
            Console.WriteLine("{0} ", r.Next(0, 5));
        }
    }
}
generates 10 integers values {0, 1, 2, 3, 4}
Drawing w/ Pen

Pens are used for lines (not filled drawing)

Pen property float Width get/set

Pen (Color c); // default width of 1
Pen (Color c, float w);

DrawLines

DrawLine (Pen p, int x1, int y1, int x2, int y2);
DrawLine (Pen p, Point p1, Point p2);
also float x, y and PointF's values
includes the second point positions

Assume Graphics g

g.DrawLine (pen, 3, 3, 3, 3); // draws nothing

DrawLines (Pen, Point[] pt);
draws a poly line, connect the points
also array of PointF
class HiYetAgain : Form {

    public static void Main() {
        Application.Run(new HiYetAgain());
    }

    public HiYetAgain() {
        ... 
        BackColor = SystemColors.Window;
        ForeColor = SystemColorsWindowText;
        ... 
    }

    protected override void OnPaint(PaintEventArgs pea) {
        Graphics g = pea.Graphics;
        g.DrawString("Hi yet again", Font,
                    new SolidBrush(ForeColor), 40, 40);
        ...
    }

    << see Petzold's RandomRectangle.cs >>
Shapes

Draw rectangles

```csharp
DrawRectangle (Pen p, Rectangle r);
DrawRectangle (Pen p, int x, int y, int w, int h);
// also w/ floats
```

Draw Polygons

```csharp
DrawPolygon (Pen p, Point[] pt);  // also w/ array of PointF
like DrawLines with array except a closing line is draw between the
first and last points in the array
```

Other draw commands -- there are overloaded methods for each

```csharp
DrawEllipse (Pen p, Rectangle r);
DrawArc (Pen p, Rectangle r,
    float start, float sweep);
DrawPie (Pen p, Rectangle r,
    float start, float sweep);
```

Arc is part of an ellipse. Pie is an Arc with edges to center.
There are several overloaded methods for each filled shape

*FillRectangle (Brush b, Rectangle r)*;

*FillRectangles (Brush b, Rectangle[] r)*;

*FillEllipse(...)*;

*FillPie (...)*

*FillPolygon (Brush b, Point[] pt, FillMode fm)*;

*FillMode has effect if lines defining polygon body overlap!*

*FillMode enumeration:
  - Alternate fill, unfilled, fill, unfilled alternate (default)
  - Winding most interiors are filled (usually fills all)*
The Graphics class provides several transformations and there is a matrix class! Useful for coordinate transformations.

World → Page → Device

TranslateTransform (float dx, dy);
ScaleTransform (float sx, xy);
RotateTransform (float radian);

all transforms can specify a MatrixOrder (for matrix composition)

RotateTransform (float radian, MatrixOrder mo);

MatrixOrder enumeration Prepend, Append

Page coordinates are transformed to Device coordinates with PageTransformations

using Graphics properties PageScale and PageUnit.

Graphics properties, GraphicState class, can be saved and restored

GraphicsState gs = g.Save(); or g.Restore = gs;
Matrices

Matrix class namespace System.Drawing.Drawing2D

Matrix is a 3 by 3 matrix useful for 2D transformations.

Matrix (); an identity matrix
Matrix ( float sx, float ry,
float rx, float sy,
float dx, float dy);

Graphics class has a Transform property that is a Matrix.

Matrices can be composed

To multiply the existing (transform) matrix with Matrix mx

MultiplyTransform(Matrix mx); // default is prepend MatrixOrder
MultiplyTranform(Matrix mx, MatrixOrder mo);

Java has matrix objects in the javax.vecmath package part of the Java 3D API download
Buttons

Bitmaps and Owner-drawn

ButtonBase Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image</td>
<td>Image get / set</td>
</tr>
<tr>
<td>ImageList</td>
<td>ImageList get / set</td>
</tr>
</tbody>
</table>

Bitmaps are *.bmp set in Designer or code with:

```csharp
aButton.Image = new Bitmap(GetType(), "*.bmp");
// new Bitmap (Type extractorClass, string fileName);
```

Program Files\Microsoft Visual Studio .NET\Common7\Graphics\subdirectories: bitmaps, cursors, icons

Owner-drawn buttons

override DrawButton(...) and DrawFocusRectangle(...) methods to be used on paint events.
TextBox usually 1 line
RichTextBox text area w/ editing

TextBase properties:
bool MultiLine, wordwrap, scrollbars

ListControl (abstract) display & selection
ListBox
CheckedListBox
ComboBox - TextBox with List

UpDownBase (abstract) Spinners:
NumericUpDown numbers
DomainUpDown strings
CheckBox

normally an on/off checkbox

Properties

<table>
<thead>
<tr>
<th>Type</th>
<th>Property</th>
<th>get / set</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td>Checked</td>
<td>get / set</td>
<td>false</td>
</tr>
<tr>
<td>bool</td>
<td>AutoCheck</td>
<td>get / set</td>
<td>true</td>
</tr>
<tr>
<td>Appearance</td>
<td>Appearance</td>
<td>get / set</td>
<td>{Normal, Button}</td>
</tr>
<tr>
<td>bool</td>
<td>ThreeState</td>
<td>get / set</td>
<td>false</td>
</tr>
</tbody>
</table>
| CheckState | CheckState | get / set | // use instead of Checked  
|           |            |           | {Unchecked, Checked, Indeterminate} |

Events

- CheckedChanged OnCheckChanged EventHandler EventArgs
- CheckStateChanged OnCheckStateChanged ""
Group Box

A container for other controls
  a title and "border" around a set of controls
  used for Radio Buttons

Radio Buttons
  mutually exclusive set of buttons
  similar properties and events as CheckBox (w/o ThreeState).
ScrollBars

Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Get/Set</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>int Value</td>
<td>get/set</td>
<td>Min..Max + 1 - LargeChange</td>
<td></td>
</tr>
<tr>
<td>int Minimum</td>
<td>get/set</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>int Maximum</td>
<td>get/set</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>int SmallChange</td>
<td>get/set</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>int LargeChange</td>
<td>get/set</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

Events

ValueChanged OnValueChanged EventHandler EventArgs
Scroll OnScroll ScrollEventHandler ScrollEventArgs

ScrollEventArgs Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Get/Set</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int NewValue</td>
<td>get/set</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ScrollEventType</td>
<td>Type</td>
<td>get</td>
<td>// how scrolled</td>
</tr>
</tbody>
</table>

many ways to use Scroll and ScrollEventArgs -- see text

TrackBar alternate form of ScrollBar with similar properties
Models & Views

Windows Forms has complex controls for viewing and manipulating collections of data.

Single Collections: NumericUpDown, DomainUpDown, ComboBox, ListBox

Many Collections: TreeView, ListView, and DataGrid

Views of single collections store the model in an collection of objects. The views display, and provide UIs, to the items held in the collection.

ComboBox, NumericUpDown, and DomainUpDown all have an Items property: an ObjectCollection implementation of the IList, ICollection, and IEnumerable interfaces.

ListBox model has 3 collections:
- Items (like the above),
- SelectedItems (UI event result),
- SelectedIndices (UI events result).
Hierarchical collections

The collections used by views of related collections are used to hold the values and relations.

TreeView displays collections that have a hierarchical relationship like directories and files. TreeView is used in File Explorer's Folder pane.

A TreeView has a single TreeNode

bool Sorted get / set
TreeNode string + optional image

TreeNode class has a Nodes property that is a TreeNode Collection

TreeNode Nodes [ ] get / set
int Count get
bool IsReadOnly get

TreeNode Collection implements IList, ICollection and IEnumerable with TreeNodeNodes property []

Each TreeNode can contain other TreeNodes (hierarchy)
Tabular collections

ListView displays information in a single table of rows and columns with column headings. ListView is used in File Explorer's file pane. The first column is the list views Items. The remaining columns are the subitems of the item.

ListView properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>View</td>
<td>View Enumeration</td>
<td>View Enumeration</td>
</tr>
<tr>
<td>SmallImageList</td>
<td>ImageList</td>
<td>SmallImageList</td>
</tr>
<tr>
<td>LargeImageList</td>
<td>ImageList</td>
<td>LargeImageList</td>
</tr>
<tr>
<td>Columns</td>
<td>ColumnHeaderCollection</td>
<td>Columns</td>
</tr>
<tr>
<td>Items</td>
<td>ListViewItemCollection</td>
<td>Items</td>
</tr>
</tbody>
</table>

ColumnHeaderCollection property:

```csharp
[get] ColumnHeader
```
ListViewItem Collection property

[] ListViewItem get / set

ListViewItem property:

SubItems ListViewSubItemCollection get

ListViewSubItemCollection properties

[] listViewSubItem get / set

Other property collections (results of events):

SelectedIndices, SelectedItems

Other controls:

TabControl
DateTime Picker
MonthCalendar
StatusBar w/ panels

Splitter
Database collections

Relational databases are collections of related tables (relations). Each table is identified by a primary key. A table's primary key can be a field (foreign key) in another table.

ADO.NET object model has: DataSet, DataTable and DataRelation objects

The DataSet is a subset of database cached on local machine. Data connections are used to get and set the DataSet.

A DataSet has:
  - Tables (collection of DataTable)
  - Relations (collection of DataRelation) properties.
Tables

Tables property returns a DataTablesCollection of all DataTable objects.

DataTable properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Collection</th>
<th>Accessors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Columns</td>
<td>DataColumnnsCollection</td>
<td>get</td>
</tr>
<tr>
<td>Rows</td>
<td>DataRowCollection</td>
<td>get</td>
</tr>
<tr>
<td>Constraints</td>
<td>ConstraintCollection</td>
<td>get</td>
</tr>
</tbody>
</table>

DataColumn represents a column in the table using property

item [] get / set

DataRow stores items in the row in property

ItemArray object [] get / set

and individual column items in a row as property

Item [] get set
Relations

DataRelation represents the dependencies between two tables using DataColumn objects.

Relations property of DataSet returns a DataRelationCollection of DataRelations

Data Adapter is a bridge between a DataSet and the data source.

DataAdapter.Fill() retrieve data from a database and fill a data set. DataAdapter.Update() inserts, updates, or deletes rows in the data set.

DBConnection represents a data source
DBCommand sends commands (SQL statements) to the database.

```csharp
SQLDataAdapter myDataAdapter = new SQLDataAdapter(commandString, connectionString);
DataSet myDataSet = new DataSet();
myDataAdapter.Fill(myDataSet, "DataSetName");  // tables
DataGrid

The Window Form control DataGrid provides the view to DataTables

DataGrid has a DataSource property

```csharp
DataGrid myDataGrid = new DataGrid();
myDataGrid.DataSource = myDataSet.Tables("DataSetName");
```

When working with multiple (n) tables you need

1 connection object

n SqlCommand objects for each table

n SqlDataAdapter objects for each table

Create DataRelation object for each dependency between tables, add to DataSet

Create a DataViewManager for the DataSet to provide a view of the DataSet and set the Grid's DataSource. (use DefaultViewManager).

Set the DataMember property to the parent (base) table
one dataset: semesterCourses
four tables: section, profs, students, deptFaculty
three relations: teaches, attends, memberOf

DataGrid shows relations facilitates navigation based on relations
Need to create relations after connections (adapters).
app needs to know semantics of the database.
example

<table>
<thead>
<tr>
<th>Events</th>
<th>56</th>
</tr>
</thead>
</table>

Semester Courses

CSUN:

<table>
<thead>
<tr>
<th>snum</th>
<th>prof-ID</th>
<th>student-ID</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>461</td>
<td>20</td>
<td>261</td>
<td>CS 585</td>
</tr>
<tr>
<td>463</td>
<td>21</td>
<td>118</td>
<td>CS 310</td>
</tr>
<tr>
<td>550</td>
<td>20</td>
<td>115</td>
<td>CS 585</td>
</tr>
<tr>
<td>464</td>
<td>11</td>
<td>118</td>
<td>CS 465</td>
</tr>
<tr>
<td>463</td>
<td>21</td>
<td>115</td>
<td>CS 310</td>
</tr>
<tr>
<td>363</td>
<td>21</td>
<td>115</td>
<td>Math 482</td>
</tr>
</tbody>
</table>

Semester Courses

CSUN:

<table>
<thead>
<tr>
<th>Section</th>
<th>snum</th>
<th>prof-ID</th>
<th>student-ID</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>463</td>
<td>464</td>
<td>11</td>
<td>118</td>
<td>CS 585</td>
</tr>
<tr>
<td>463</td>
<td>21</td>
<td>115</td>
<td>CS 310</td>
<td></td>
</tr>
</tbody>
</table>

Semester Courses

CSUN:

<table>
<thead>
<tr>
<th>Section</th>
<th>snum</th>
<th>prof-ID</th>
<th>student-ID</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>463</td>
<td>21</td>
<td>118</td>
<td>Math 482</td>
<td></td>
</tr>
</tbody>
</table>

Semester Courses

CSUN:

<table>
<thead>
<tr>
<th>Section</th>
<th>snum</th>
<th>prof-ID</th>
<th>student-ID</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>463</td>
<td>21</td>
<td>118</td>
<td>Math 482</td>
<td></td>
</tr>
</tbody>
</table>