Designing Interactive Systems II

Computer Science Graduate Programme SS 2010

Prof. Dr. Jan Borchers
RWTH Aachen University

http://hci.rwth-aachen.de

Review: Conviviality (Igoe)

- rules for networking
- role of physical objects in remote collaboration
- participation vs. consumption
- objects as communication starter
- sociable objects
- pachube, asthmapolis, patientslikeme


CNN

Windows

WPF

Windows

GDI

WDDM

DXGI

UDDI

DWM

CIL

DCE

WCF

WIC

MFC

OLE

Windows 7: Architecture

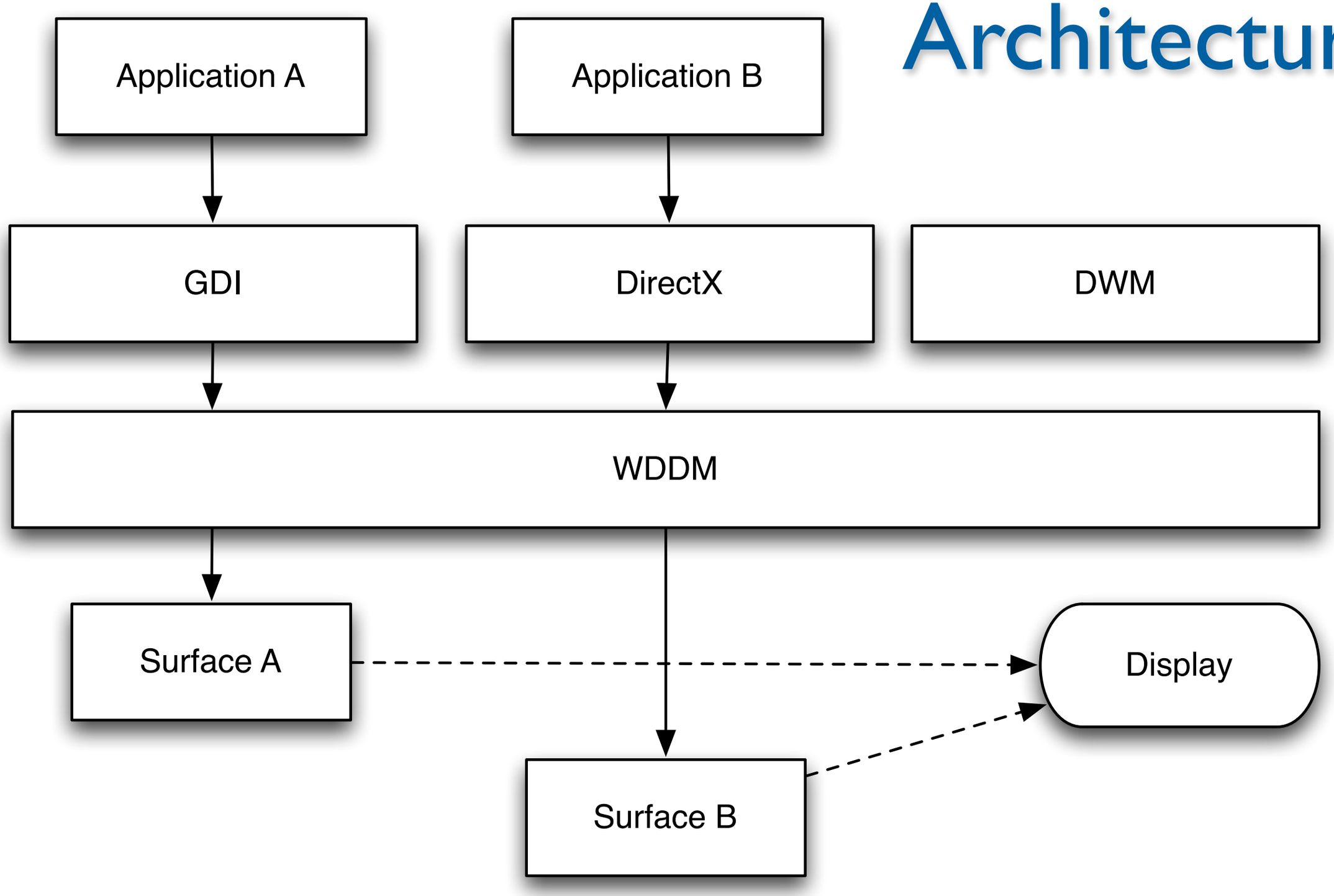
Apps	
UITK	.NET, MFC, Win32
WM	DWM
BWS	DWM
GEL	WPF, Win32
HW	

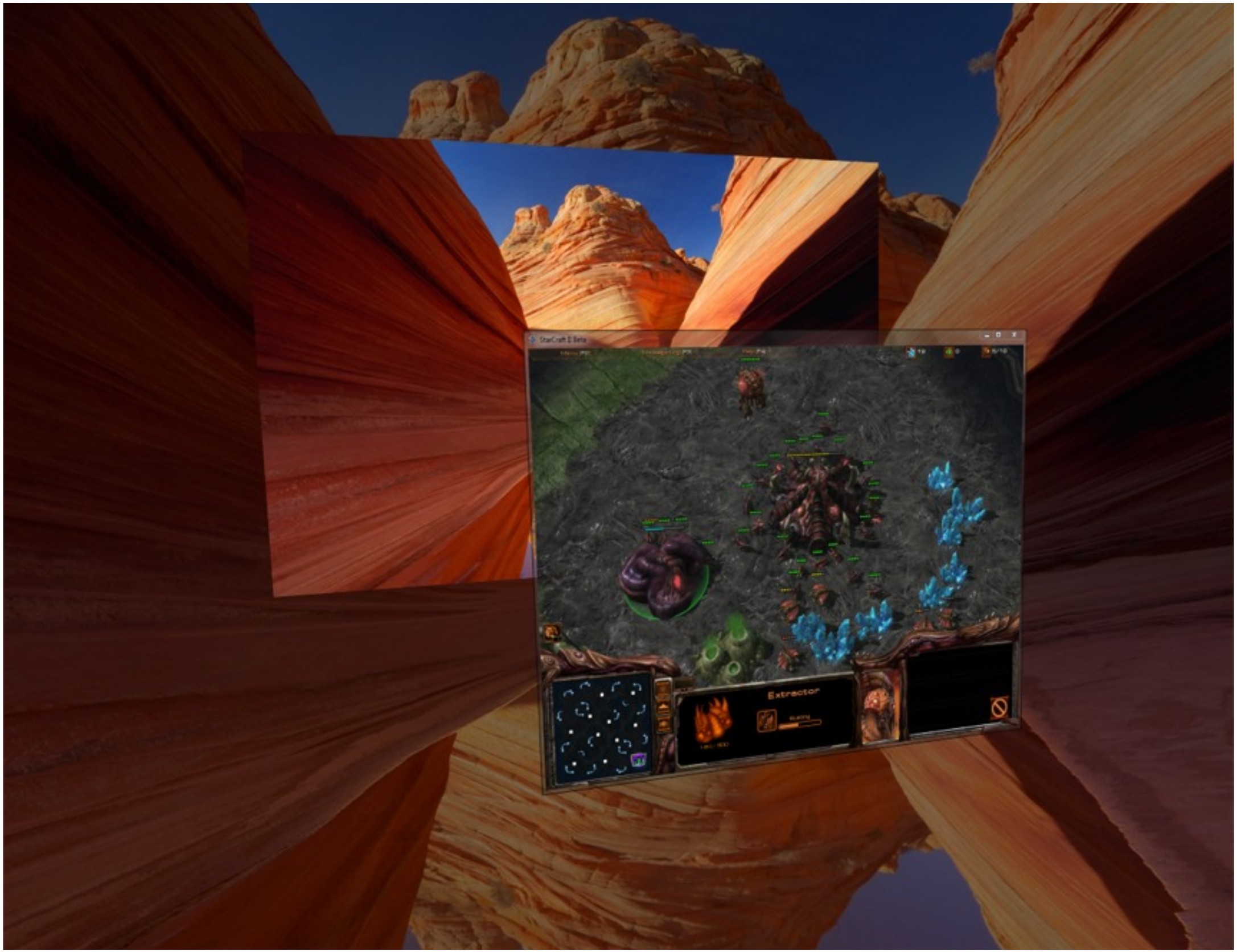
- Microsoft Foundation Classes
- .NET 4.0
- Win32: potpourri
  - *Graphics Device Interface:* drawing functions
  - *User Interface:* user input, windowing, "look-and-feel"

Desktop Window Manager

- enables Windows Aero graphical user interface
- applications render in off-screen buffer
- DWM is DirectX application

Windows 7: Graphics Architecture




Extractor

DirectX

DirectX

- Game programmers preferred working with DOS
  - direct access to video card, sound devices, ...
- First Release of DirectX with Windows 95
  - DirectDraw, DirectSound and DirectPlay

Microsoft® DirectX® 11

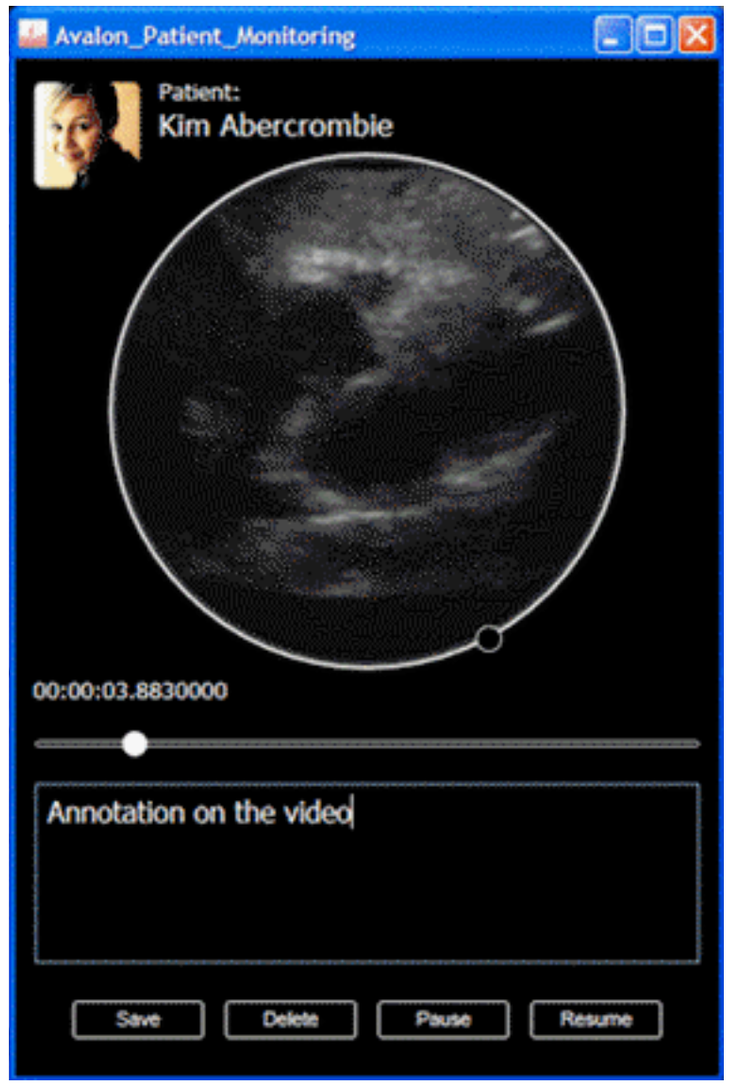
DirectX

- Large rewrite of DirectX for Vista
- DirectWrite
  - Support for rendering text
- Direct2D
  - rendering 2D graphics
  - based on Direct3D

Software development for Windows

Windows Presentation Foundation

- DirectX based graphics subsystem
- also: development framework
  - Goal: "help developers create attractive and effective user interfaces"
  - Combine strengths of various existing technologies to create coherent foundation



Microsoft .NET Framework

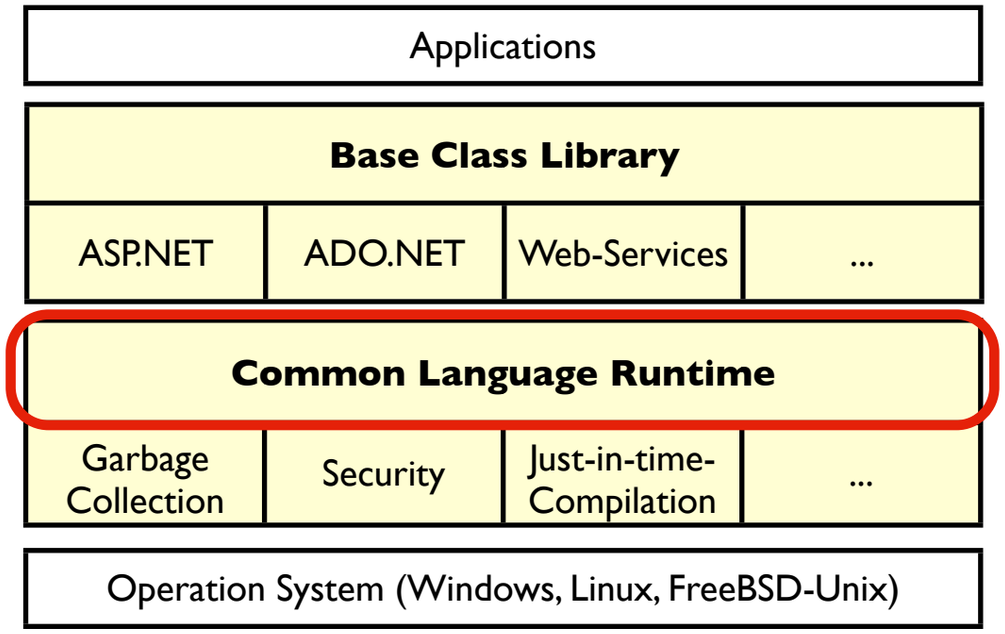
Microsoft .NET Framework

- Software platform developed by Microsoft
- Layer on top on Windows (and other systems)
- Vision: Join all existing software systems and platforms
- History
  - 2000: Bill Gates presents .NET-"Vision"
  - 2002: .NET v1.0 released with SDK and Visual Studio .NET 2002
  - 2003: .NET v1.1 + Visual Studio 2003
  - 2004: .NET v2.0 + Visual Studio 2005
  - 2006: .NET v3.0
  - 2007: .NET v3.5 + Visual Studio 2008
  - 2008: Source disclosed (reason: "simplified debugging")
  - 2010: .NET 4.0 + Visual Studio 2010

.NET Architecture

Applications			
Base Class Library			
ASP.NET	ADO.NET	Web-Services	...
Common Language Runtime			
Garbage Collection	Security	Just-in-time-Compilation	...
Operating System (Windows, Linux, FreeBSD-Unix)			

Common Language Runtime



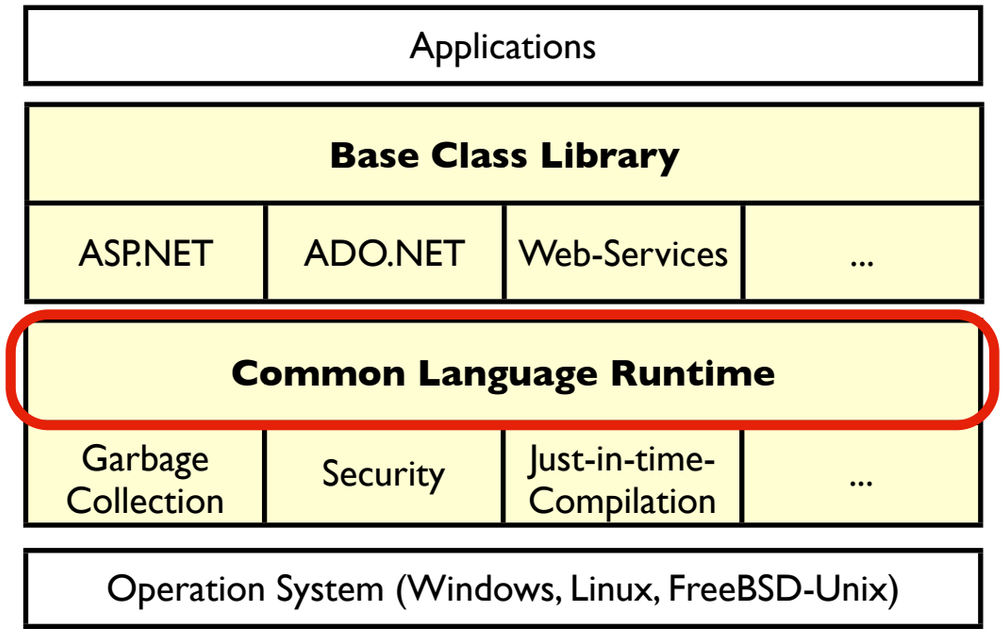
- Runtime environment for all .NET programs
- Virtual machine, just-in-time compilation

  → Platform and language independence
- Common Intermediate Language (CIL)

  Instruction set for virtual machine
- Common Type System (CTS)

  Specifies how classes, interface, elementary types look like

Common Language Runtime

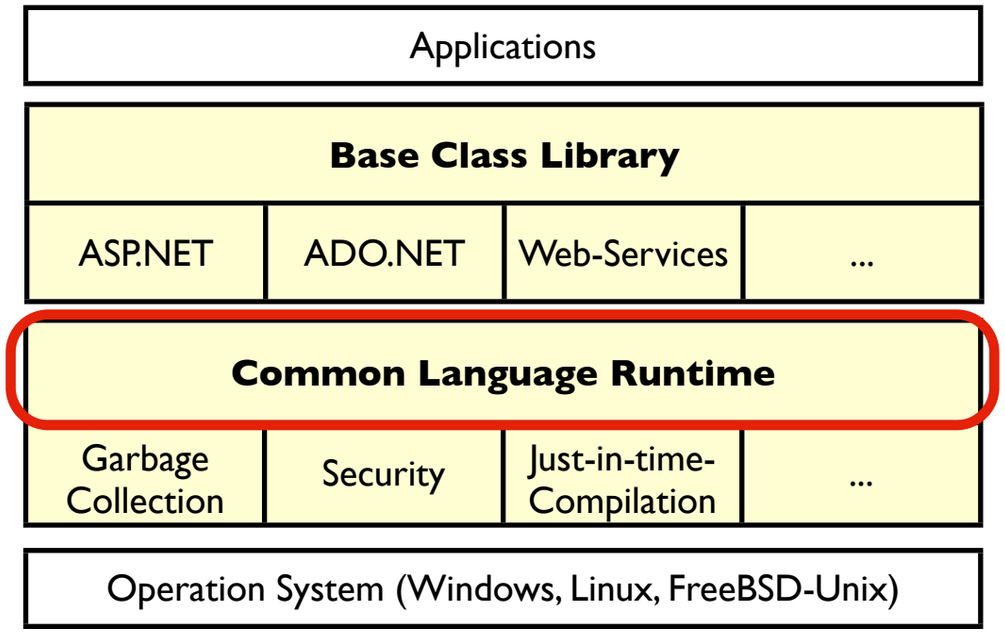


- Common Language Specification (CLS)
  - smallest set of CTS which must be fulfilled by all languages
- Garbage collection
- Languages:

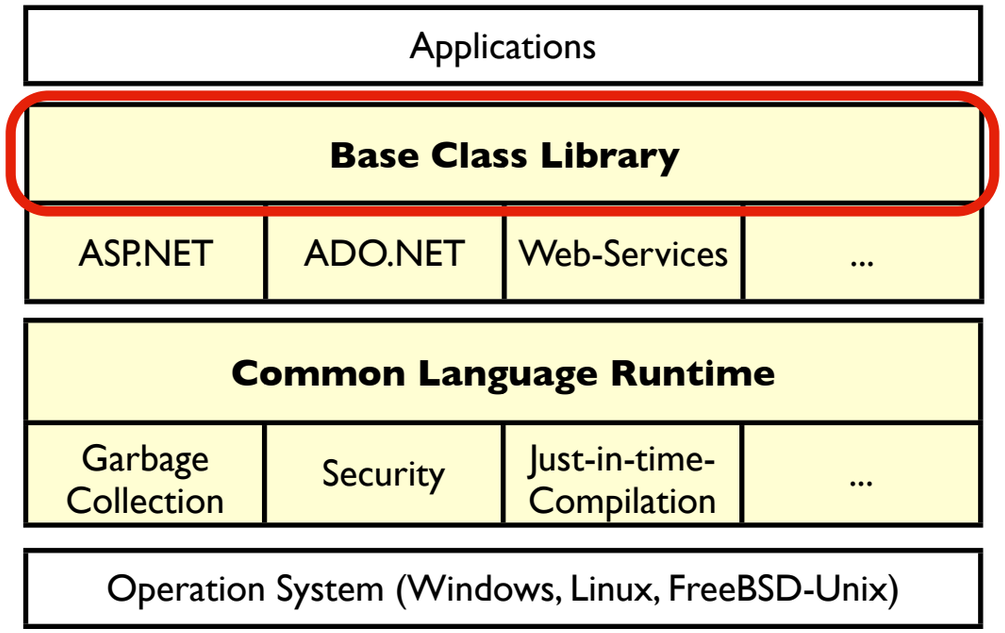
  C#, Visual Basic .NET, Managed C++, Fortran, Java, Pascal, Perl, Python, Smalltalk, etc.

Assemblies

- Assembly = smallest programming building block which can be released (.exe or .dll)
- Contains:
  - Resources (images, etc.)
  - Meta data (complete interface description of classes, fields, methods, etc.)
  - Manifest (table of contents)
  - Multistage version number
- Advantages:
  - No registration required
  - End of "DLL hell"



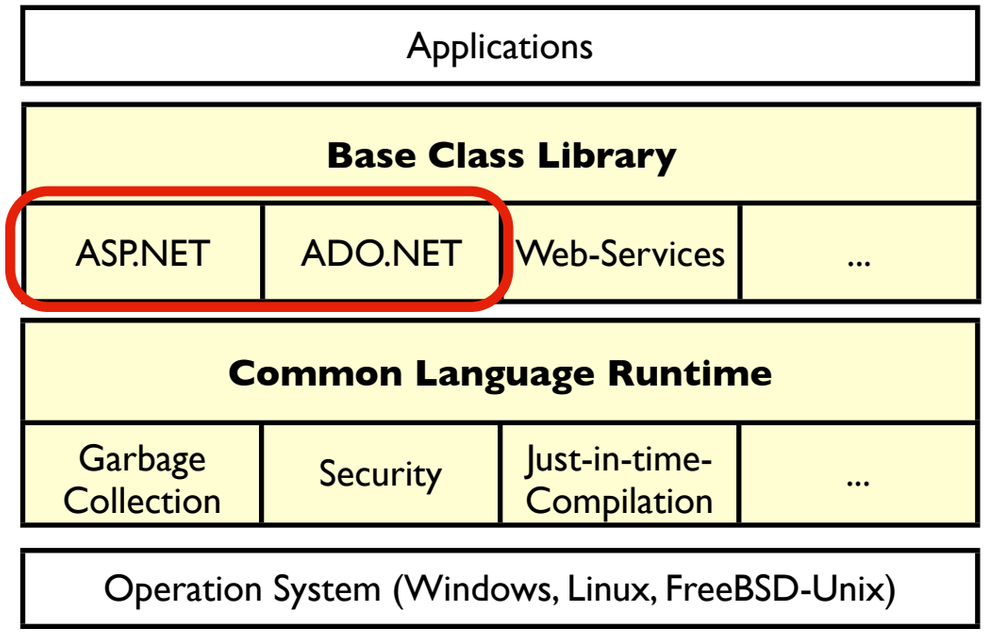
Base Class Library



- Class library of .NET
  - System.Collections, System.Collections.Generic
  - System.IO
  - System.Threading
  - System.Net
  - System.Reflection
  - System.Windows.Forms
  - System.Xml

ASP.NET

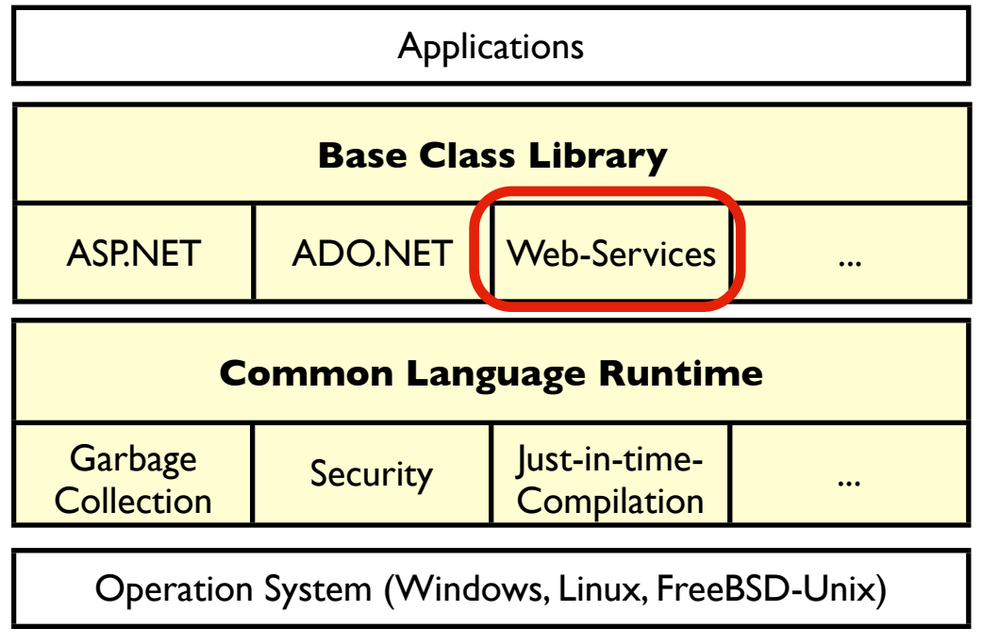
- Programming of dynamic websites
- Complete object oriented model (C#, Visual Basic .NET)
- Rich library of GUI elements
- Easy handling of validators, authentication
- Drag-and-drop design of websites



ADO.NET

- Access to database and other data source (e.g. XML files)
- Support relational databases with transactions and lock mechanisms
- Independent of concrete database architecture

Web-Services



- New concept for distributed applications
- Remote procedure calls over HTTP or SOAP
- Desktop application retrieve information via Web-Services w/o noticing network protocol
- UDDI = Universal Description, Discovery and Integration


Microsoft

Win32 API

```
#define PROG_NAME "Win32 Hello World"

HWND hWnd = NULL;
HANDLE hThread = NULL;

unsigned int __stdcall thread_main(void*) {
   MessageBox(NULL, "hello, world", PROG_NAME, MB_OK | MB_TOPMOST);
   hThread = NULL;
   PostMessage(hWnd, WM_CLOSE, 0, 0);
   return 0;
}

HANDLE start_thread() {
   unsigned int id;
   hThread = (HANDLE)_beginthreadex(NULL, 0, thread_main, NULL, 0, &id);
   if (hThread == NULL) {
      // error
   }
   return hThread;
}

static LRESULT CALLBACK win_proc(HWND hwnd, UINT msg, WPARAM wp, LPARAM lp) {
   switch (msg) {
      case WM_CREATE:
         hWnd = hwnd;
         if (start_thread() == NULL) {
            PostMessage(hwnd, WM_CLOSE, 0, 0);
         }
         return 0;
      case WM_CLOSE:
         if (hThread != NULL) {
            WaitForSingleObject(hThread, INFINITE);
            CloseHandle(hThread);
         }
         DestroyWindow(hwnd);
         return 0;
      case WM_DESTROY:
         PostQuitMessage(0);
         return 0;
      default:
         return DefWindowProc(hwnd, msg, wp, lp);
   }
}
```

```
int WINAPI WinMain(HINSTANCE hi, HINSTANCE hp, LPSTR cmdline, int cmdshow) {
   if (!hp) {
      WNDCLASS wc;
      wc.style = 0;
      wc.lpfnWndProc = win_proc;
      wc.cbClsExtra = 0;
      wc.cbWndExtra = 0;
      wc.hInstance = hi;
      wc.hIcon = LoadIcon(NULL, IDI_APPLICATION);
      wc.hCursor = LoadCursor(NULL, IDC_ARROW);
      wc.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);
      wc.lpszMenuName = NULL;
      wc.lpszClassName = PROG_NAME;
      if (! RegisterClass(&wc)) {
         // error
         return 0;
      }
   }

   HWND wnd = CreateWindow(PROG_NAME, PROG_NAME,
                           WS_POPUP, 0, 0, 0, 0, NULL, NULL, hi, NULL);
   if (wnd == NULL) {
      // error
      return 0;
   }
   ShowWindow(wnd, SW_SHOW);
   UpdateWindow(wnd);
   MSG msg;
   while (GetMessage(&msg, NULL, 0, 0)) {
      TranslateMessage(&msg);
      DispatchMessage(&msg);
   }
   return msg.wParam;
}
```


Win32 API

```
#define PROG_NAME "Win32 Hello World"

HWND hWnd = NULL;
HANDLE hThread = NULL;

unsigned int __stdcall thread_main(void*) {
   MessageBox(NULL, "hello, world", PROG_NAME, MB_OK | MB_TOPMOST);
   hThread = NULL;
   PostMessage(hWnd, WM_CLOSE, 0, 0);
   return 0;
}

HANDLE start_thread() {
   unsigned int id;
   hThread = (HANDLE)_beginthreadex(NULL, 0, thread_main, NULL, 0, &id);
   if (hThread == NULL) {
      // error
   }
   return hThread;
}

static LRESULT CALLBACK win_proc(HWND hwnd, UINT msg, WPARAM wp, LPARAM lp) {
   switch (msg) {
      case WM_CREATE:
         hWnd = hwnd;
         if (start_thread() == NULL) {
            PostMessage(hwnd, WM_CLOSE, 0, 0);
         }
         return 0;
      case WM_CLOSE:
         if (hThread != NULL) {
            WaitForSingleObject(hThread, INFINITE);
            CloseHandle(hThread);
         }
         DestroyWindow(hwnd);
         return 0;
      case WM_DESTROY:
         PostQuitMessage(0);
         return 0;
      default:
         return DefWindowProc(hwnd, msg, wp, lp);
   }
}
```

```
int WINAPI WinMain(HINSTANCE hi, HINSTANCE hp, LPSTR cmdline, int cmdshow) {
   if (!hp) {
      WNDCLASS wc;
      wc.style = 0;
      wc.lpfnWndProc = win_proc;
      wc.cbClsExtra = 0;
      wc.cbWndExtra = 0;
      wc.hInstance = hi;
      wc.hIcon = LoadIcon(NULL, IDI_APPLICATION);
      wc.hCursor = LoadCursor(NULL, IDC_ARROW);
      wc.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);
      wc.lpszMenuName = NULL;
      wc.lpszClassName = PROG_NAME;
      if (! RegisterClass(&wc)) {
         // error
         return 0;
      }
   }

   HWND wnd = CreateWindow(PROG_NAME, PROG_NAME,
                           WS_POPUP, 0, 0, 0, 0, NULL, NULL, hi, NULL);
   if (wnd == NULL) {
      // error
      return 0;
   }
   ShowWindow(wnd, SW_SHOW);
   UpdateWindow(wnd);
   MSG msg;
   while (GetMessage(&msg, NULL, 0, 0)) {
      TranslateMessage(&msg);
      DispatchMessage(&msg);
   }
   return msg.wParam;
}
```

Win32 API

```
#define PROG_NAME "Win32 Hello World"

HWND hWnd = NULL;
HANDLE hThread = NULL;

unsigned int __stdcall thread_main(void*) {
   MessageBox(NULL, "hello, world", PROG_NAME, MB_OK | MB_TOPMOST);
   hThread = NULL;
   PostMessage(hWnd, WM_CLOSE, 0, 0);
   return 0;
}

HANDLE start_thread() {
   unsigned int id;
   hThread = (HANDLE)_beginthreadex(NULL, 0, thread_main, NULL, 0, &id);
   if (hThread == NULL) {
      // error
   }
   return hThread;
}

static LRESULT CALLBACK win_proc(HWND hwnd, UINT msg, WPARAM wp, LPARAM lp) {
   switch (msg) {
      case WM_CREATE:
         hWnd = hwnd;
         if (start_thread() == NULL) {
            PostMessage(hwnd, WM_CLOSE, 0, 0);
         }
         return 0;
      case WM_CLOSE:
         if (hThread != NULL) {
            WaitForSingleObject(hThread, INFINITE);
            CloseHandle(hThread);
         }
         DestroyWindow(hwnd);
         return 0;
      case WM_DESTROY:
         PostQuitMessage(0);
         return 0;
      default:
         return DefWindowProc(hwnd, msg, wp, lp);
   }
}
```

```
int WINAPI WinMain(HINSTANCE hi, HINSTANCE hp, LPSTR cmdline, int cmdshow) {
   if (!hp) {
      WNDCLASS wc;
      wc.style = 0;
      wc.lpfnWndProc = win_proc;
      wc.cbClsExtra = 0;
      wc.cbWndExtra = 0;
      wc.hInstance = hi;
      wc.hIcon = LoadIcon(NULL, IDI_APPLICATION);
      wc.hCursor = LoadCursor(NULL, IDC_ARROW);
      wc.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);
      wc.lpszMenuName = NULL;
      wc.lpszClassName = PROG_NAME;
      if (! RegisterClass(&wc)) {
         // error
         return 0;
      }
   }

   HWND wnd = CreateWindow(PROG_NAME, PROG_NAME,
                           WS_POPUP, 0, 0, 0, 0, NULL, NULL, hi, NULL);
   if (wnd == NULL) {
      // error
      return 0;
   }
   ShowWindow(wnd, SW_SHOW);
   UpdateWindow(wnd);
   MSG msg;
   while (GetMessage(&msg, NULL, 0, 0)) {
      TranslateMessage(&msg);
      DispatchMessage(&msg);
   }
   return msg.wParam;
}
```

Win32 API

```
#define PROG_NAME "Win32 Hello World"

HWND hWnd = NULL;
HANDLE hThread = NULL;

unsigned int __stdcall thread_main(void*) {
   MessageBox(NULL, "hello, world", PROG_NAME, MB_OK | MB_TOPMOST);
   hThread = NULL;
   PostMessage(hWnd, WM_CLOSE, 0, 0);
   return 0;
}

HANDLE start_thread() {
   unsigned int id;
   hThread = (HANDLE)_beginthreadex(NULL, 0, thread_main, NULL, 0, &id);
   if (hThread == NULL) {
      // error
   }
   return hThread;
}

static LRESULT CALLBACK win_proc(HWND hwnd, UINT msg, WPARAM wp, LPARAM lp) {
   switch (msg) {
      case WM_CREATE:
         hWnd = hwnd;
         if (start_thread() == NULL) {
            PostMessage(hwnd, WM_CLOSE, 0, 0);
         }
         return 0;
      case WM_CLOSE:
         if (hThread != NULL) {
            WaitForSingleObject(hThread, INFINITE);
            CloseHandle(hThread);
         }
         DestroyWindow(hwnd);
         return 0;
      case WM_DESTROY:
         PostQuitMessage(0);
         return 0;
      default:
         return DefWindowProc(hwnd, msg, wp, lp);
   }
}
```

```
int WINAPI WinMain(HINSTANCE hi, HINSTANCE hp, LPSTR cmdline, int cmdshow) {
   if (!hp) {
      WNDCLASS wc;
      wc.style = 0;
      wc.lpfnWndProc = win_proc;
      wc.cbClsExtra = 0;
      wc.cbWndExtra = 0;
      wc.hInstance = hi;
      wc.hIcon = LoadIcon(NULL, IDI_APPLICATION);
      wc.hCursor = LoadCursor(NULL, IDC_ARROW);
      wc.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);
      wc.lpszMenuName = NULL;
      wc.lpszClassName = PROG_NAME;
      if (! RegisterClass(&wc)) {
         // error
         return 0;
      }
   }

   HWND wnd = CreateWindow(PROG_NAME, PROG_NAME,
                           WS_POPUP, 0, 0, 0, 0, NULL, NULL, hi, NULL);
   if (wnd == NULL) {
      // error
      return 0;
   }
   ShowWindow(wnd, SW_SHOW);
   UpdateWindow(wnd);
   MSG msg;
   while (GetMessage(&msg, NULL, 0, 0)) {
      TranslateMessage(&msg);
      DispatchMessage(&msg);
   }
   return msg.wParam;
}
```

Hello, World - MFC

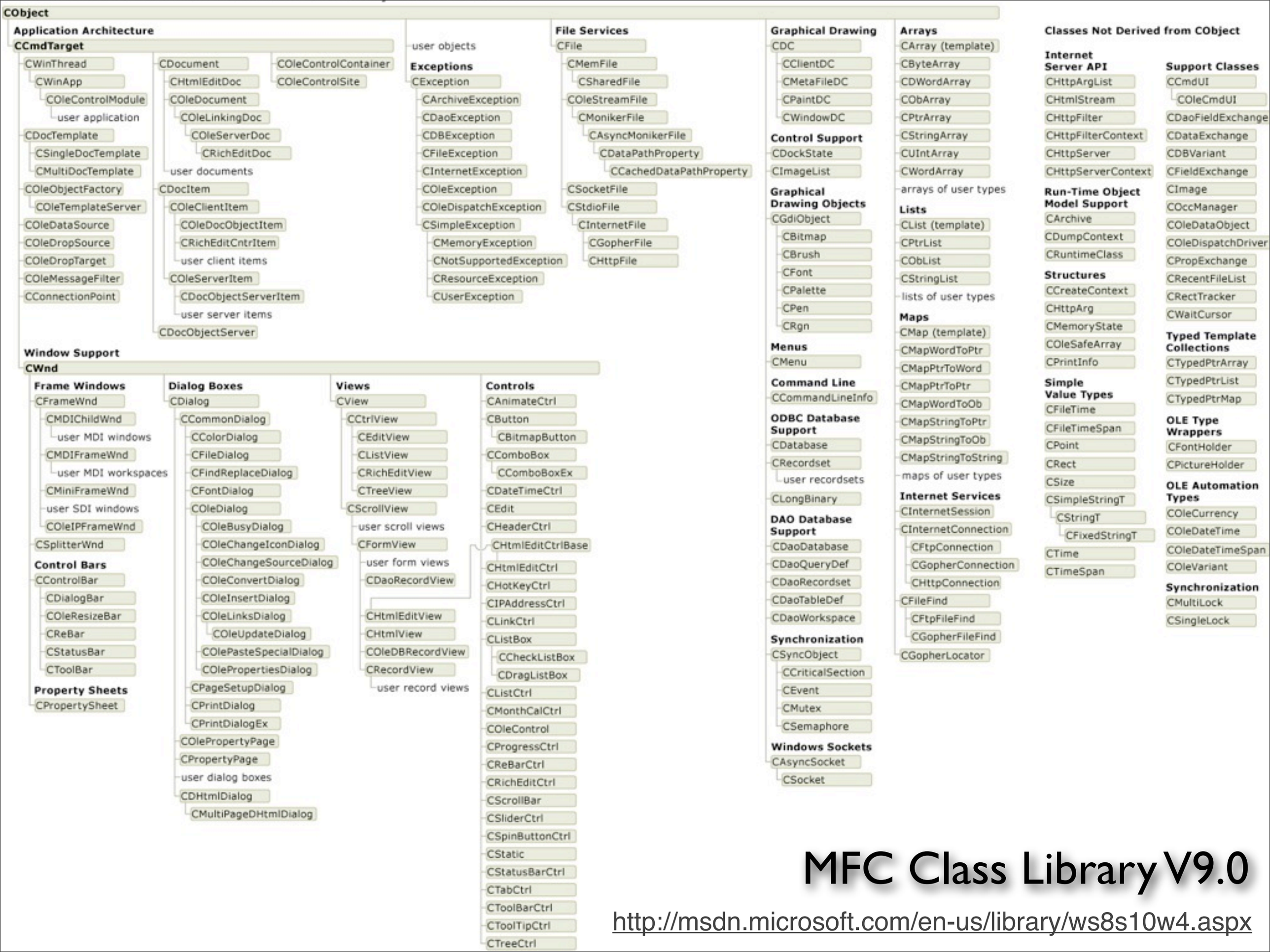
```
class HelloApplication : public CWinApp {
public:
   virtual BOOL InitInstance();
};

HelloApplication HelloApp;

#define BUTTON_ID 1001
class HelloWindow : public CFrameWnd {
public:
   HelloWindow();
protected:
   afx_msg void OnClicked();
   DECLARE_MESSAGE_MAP();
   CButton *m_pHelloButton;
};
```

```
BOOL HelloApplication::InitInstance() {
   m_pMainWnd = new HelloWindow();
   m_pMainWnd->ShowWindow(m_nCmdShow);
   m_pMainWnd->UpdateWindow();
   return TRUE;
}

BEGIN_MESSAGE_MAP(HelloWindow, CFrameWnd)
   ON_BN_CLICKED(BUTTON_ID, OnClicked)
END_MESSAGE_MAP()

HelloWindow::HelloWindow() {
   Create(NULL, _T("Hello World!"), WS_OVERLAPPEDWINDOW, CRect(0,0,160,100));
   m_pHelloButton = new CButton();
   m_pHelloButton->Create(_T("Hello World!"), WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON, CRect(20,20,120,40), this, BUTTON_ID);
}

void HelloWindow::OnClicked() {
   PostMessage(WM_CLOSE);
}
```


Hello, World - MFC

```
class HelloApplication : public CWinApp {
public:
   virtual BOOL InitInstance();
};

HelloApplication HelloApp;

#define BUTTON_ID 1001
class HelloWindow : public CFrameWnd {
public:
   HelloWindow();
protected:
   afx_msg void OnClicked();
   DECLARE_MESSAGE_MAP();
   CButton *m_pHelloButton;
};
```


```
BOOL HelloApplication::InitInstance() {
   m_pMainWnd = new HelloWindow();
   m_pMainWnd->ShowWindow(m_nCmdShow);
   m_pMainWnd->UpdateWindow();
   return TRUE;
}

BEGIN_MESSAGE_MAP(HelloWindow, CFrameWnd)
   ON_BN_CLICKED(BUTTON_ID, OnClicked)
END_MESSAGE_MAP()

HelloWindow::HelloWindow() {
   Create(NULL, _T("Hello World!"), WS_OVERLAPPEDWINDOW, CRect(0,0,160,100));
   m_pHelloButton = new CButton();
   m_pHelloButton->Create(_T("Hello World!"), WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON, CRect(20,20,120,40), this, BUTTON_ID);
}

void HelloWindow::OnClicked() {
   PostMessage(WM_CLOSE);
}
```

MFC Class Library V9.0

http://msdn.microsoft.com/en-us/library/ws8s10w4.aspx

Hello World - WinForms

```
public class MyForm : Form {
   private Button button = new Button();

   MyForm() {
      Text = "Hello WinForms!";

      button.Text = "Hello World!";
      button.Anchor = AnchorStyles.Top | AnchorStyles.Left;
      EventHandler handler = new EventHandler(buttonClicked);
      button.Click += handler;

      this.Controls.Add(button);
   }

   private void buttonClicked(object sender, EventArgs e) {
      Application.Exit();
   }

   public static void Main(string[] args) {
      Application.Run(new MyForm());
   }
}
```


Hello World - WinForms


```
public class MyForm : Form {
   private Button button = new Button();

   MyForm() {
      Text = "Hello WinForms!";

      button.Text = "Hello World!";
      button.Anchor = AnchorStyles.Top | AnchorStyles.Left;
      EventHandler handler = new EventHandler(buttonClicked);
      button.Click += handler;

      this.Controls.Add(button);
   }

   private void buttonClicked(object sender, EventArgs e) {
      Application.Exit();
   }

   public static void Main(string[] args) {
      Application.Run(new MyForm());
   }
}
```


Hello World - WPF

```
<Window
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    Title="Window with Button"
    Width="250" Height="100">

  <Button Name="button">Click Me!</Button>
</Window>
```

```
public partial class MyWindow : Window
   public MyWindow()
   {
      InitializeComponent();
   }

   void button_Click(object sender, RoutedEventArgs e)
   {
      Application.Exit();
   }
}
```

Hello World - WPF

```
<Window
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    Title="Window with Button"
    Width="250" Height="100">

  <Button Name="button">Click Me!</Button>
</Window>
```


```
public partial class MyWindow : Window
   public MyWindow()
   {
      InitializeComponent();
   }

   void button_Click(object sender, RoutedEventArgs e)
   {
      Application.Exit();
   }
}
```

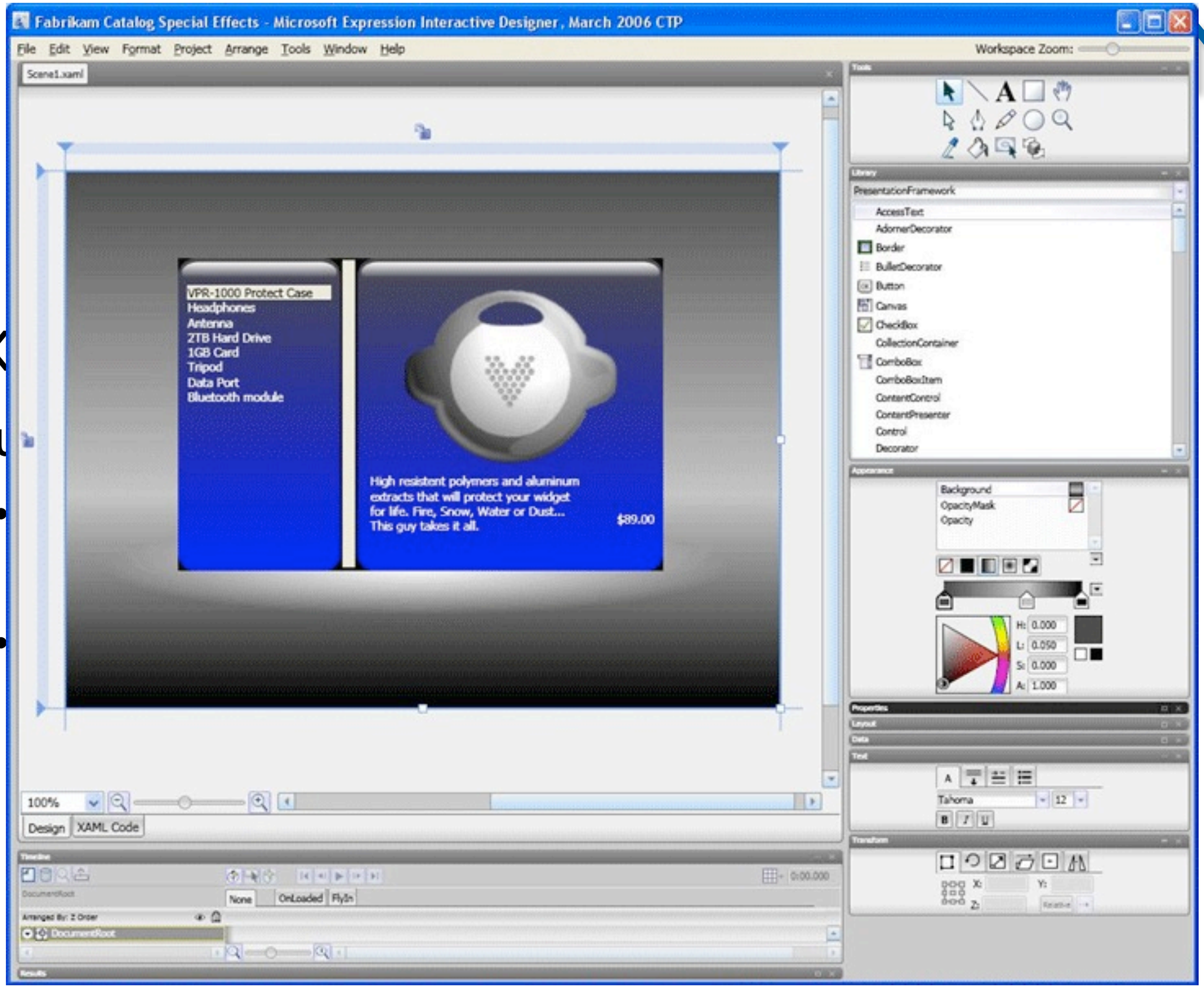

Windows: C#

- Introduced together with .NET in 2000
- Version 3.0 released with Vista
- Design is based on Java and C++
  - Runs on a virtual machine (like Java)
  - Garbage collection
  - Single object hierarchy
  - Reflection
  - Explicit pointer manipulation permitted
  - Versioning

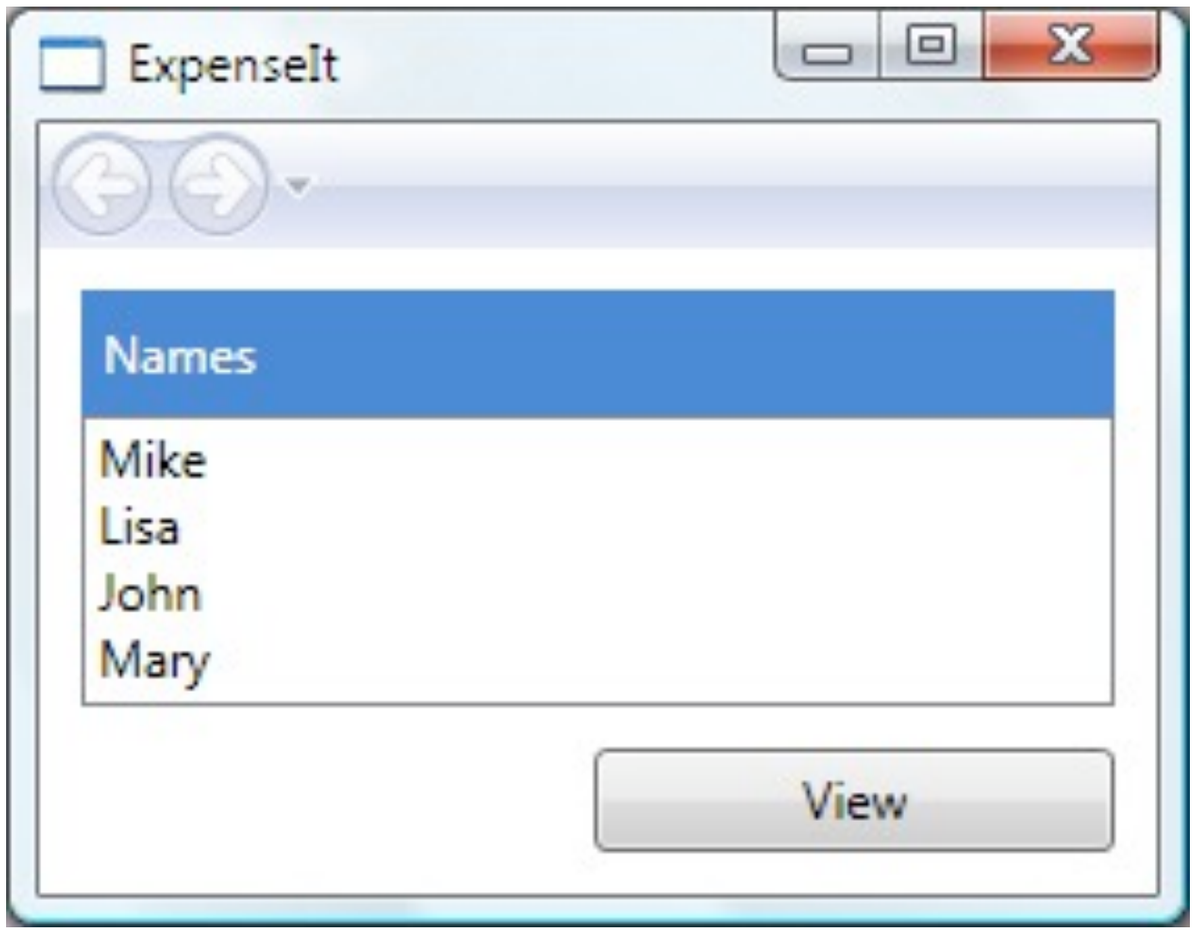
XAML

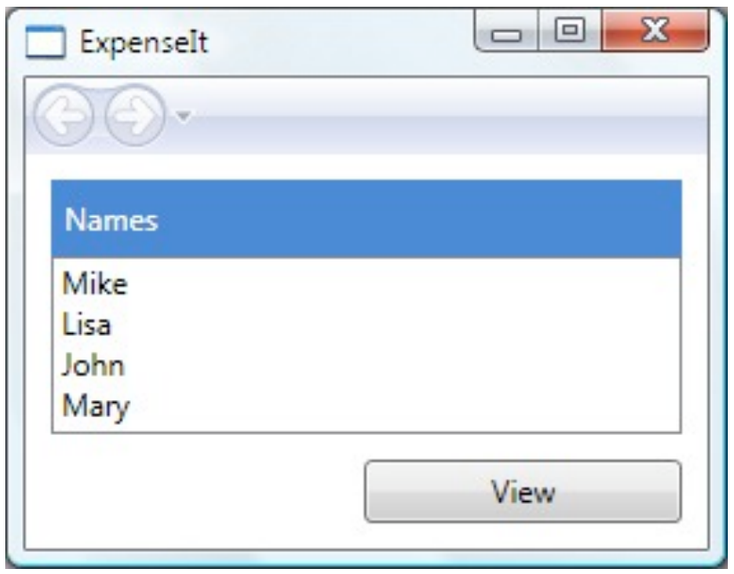
- XML based markup language to describe UI
- support for new development process
  - designer creates UI
    - Expression Interactive Designer
  - programmer adds functionality
    - Visual Studio

- X
- su


Fabrikam Catalog Special Effects - Microsoft Expression Interactive Designer, March 2006 CTP
File Edit View Format Project Arrange Tools Window Help
Workspace Zoom:
Scene1.xaml
VPR-1000 Protect Case
Headphones
Antenna
2TB Hard Drive
1GB Card
Tripod
Data Port
Bluetooth module
High resistent polymers and aluminum extracts that will protect your widget for life. Fire, Snow, Water or Dust... This guy takes it all.
$89.00
100%
Design
XAML Code
Tools
Library
PresentationFramework
AccessText
AdornerDecorator
Border
BulletDecorator
Button
Canvas
CheckBox
CollectionContainer
ComboBox
ComboBoxItem
ContentControl
ContentPresenter
Control
Decorator
Appearance
Background
OpacityMask
Opacity
Timeline
None
OnLoaded
FlyIn
DocumentRoot
Results

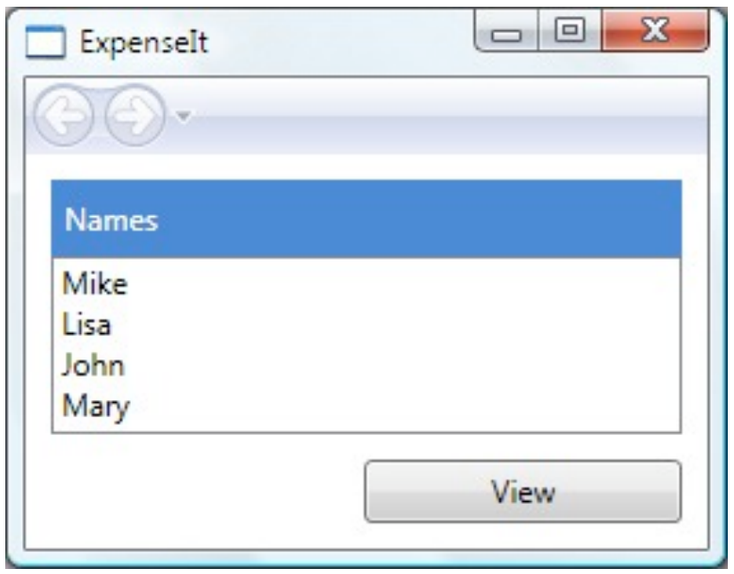
XAML: Example





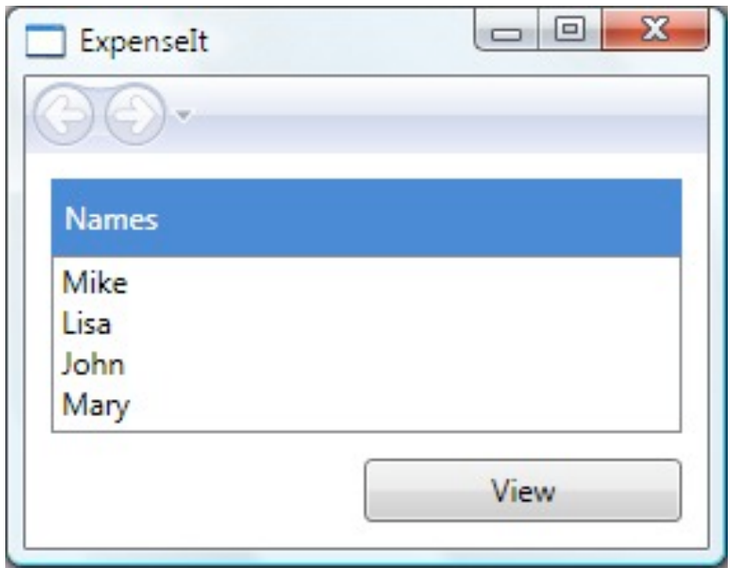
XAML: Example

```
<Border Grid.Column="0" Height="35" Padding="5" Background="#4E87D4">
<Label VerticalAlignment="Center" Foreground="White">Names</Label>
</Border>
<ListBox Name="peopleListBox" Grid.Column="0" Grid.Row="1">
   <ListBoxItem>Mike</ListBoxItem>
   <ListBoxItem>Lisa</ListBoxItem>
   <ListBoxItem>John</ListBoxItem>
   <ListBoxItem>Mary</ListBoxItem>
</ListBox>
<!-- View report button -->
<Button Grid.Column="0" Grid.Row="2" Margin="0,10,0,0" Width="125"
        Height="25" HorizontalAlignment="Right">View
        </Button>
```

XAML: Example

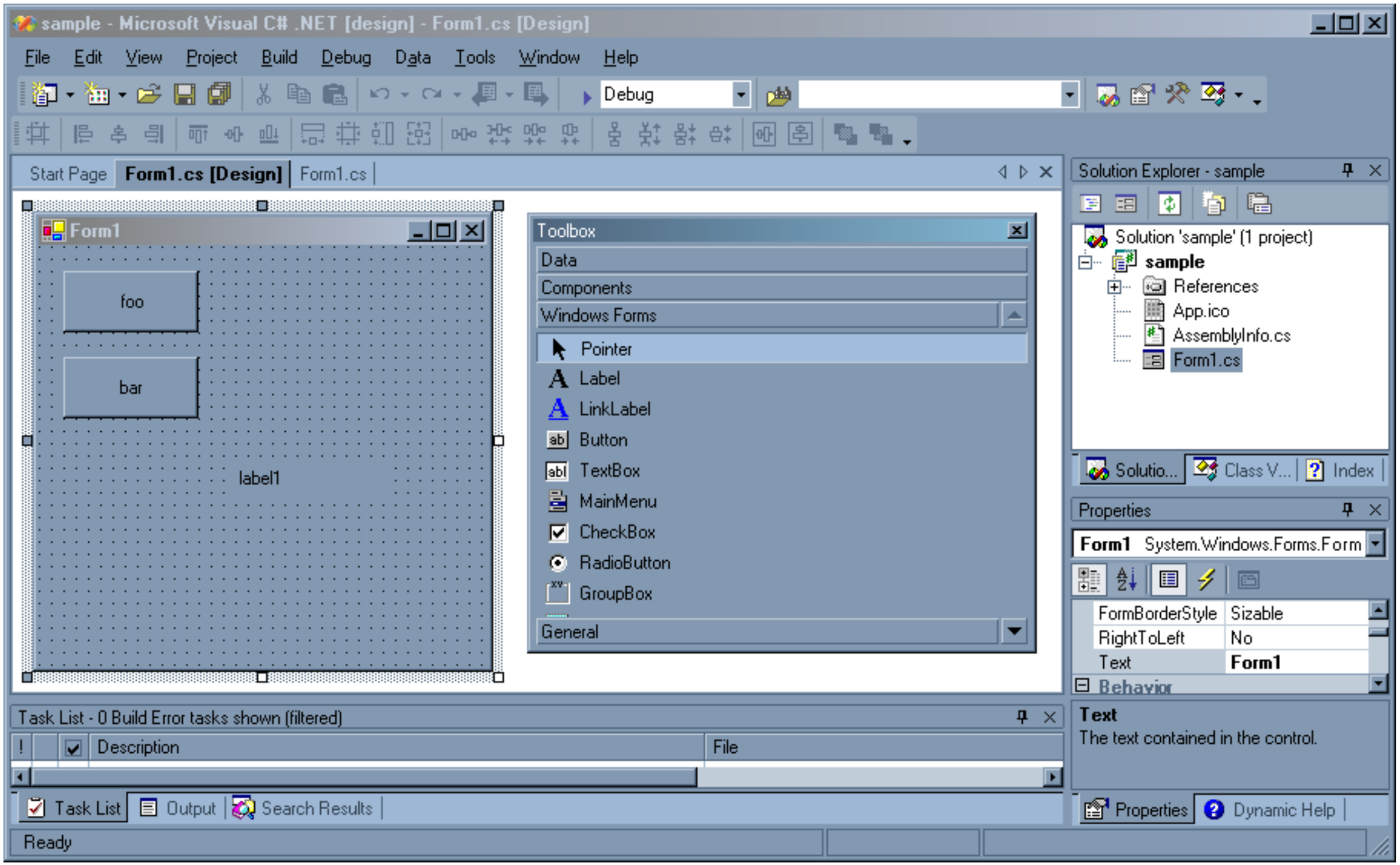
```
<Border Grid.Column="0" Height="35" Padding="5" Background="#4E87D4">
<Label VerticalAlignment="Center" Foreground="White">Names</Label>
</Border>
<ListBox Name="peopleListBox" Grid.Column="0" Grid.Row="1">
    <ListBoxItem>Mike</ListBoxItem>
    <ListBoxItem>Lisa</ListBoxItem>
    <ListBoxItem>John</ListBoxItem>
    <ListBoxItem>Mary</ListBoxItem>
</ListBox>
<!-- View report button -->
<Button Grid.Column="0" Grid.Row="2" Margin="0,10,0,0" Width="125"
        Height="25" HorizontalAlignment="Right"
        Click="Button_Click">View</Button>
```

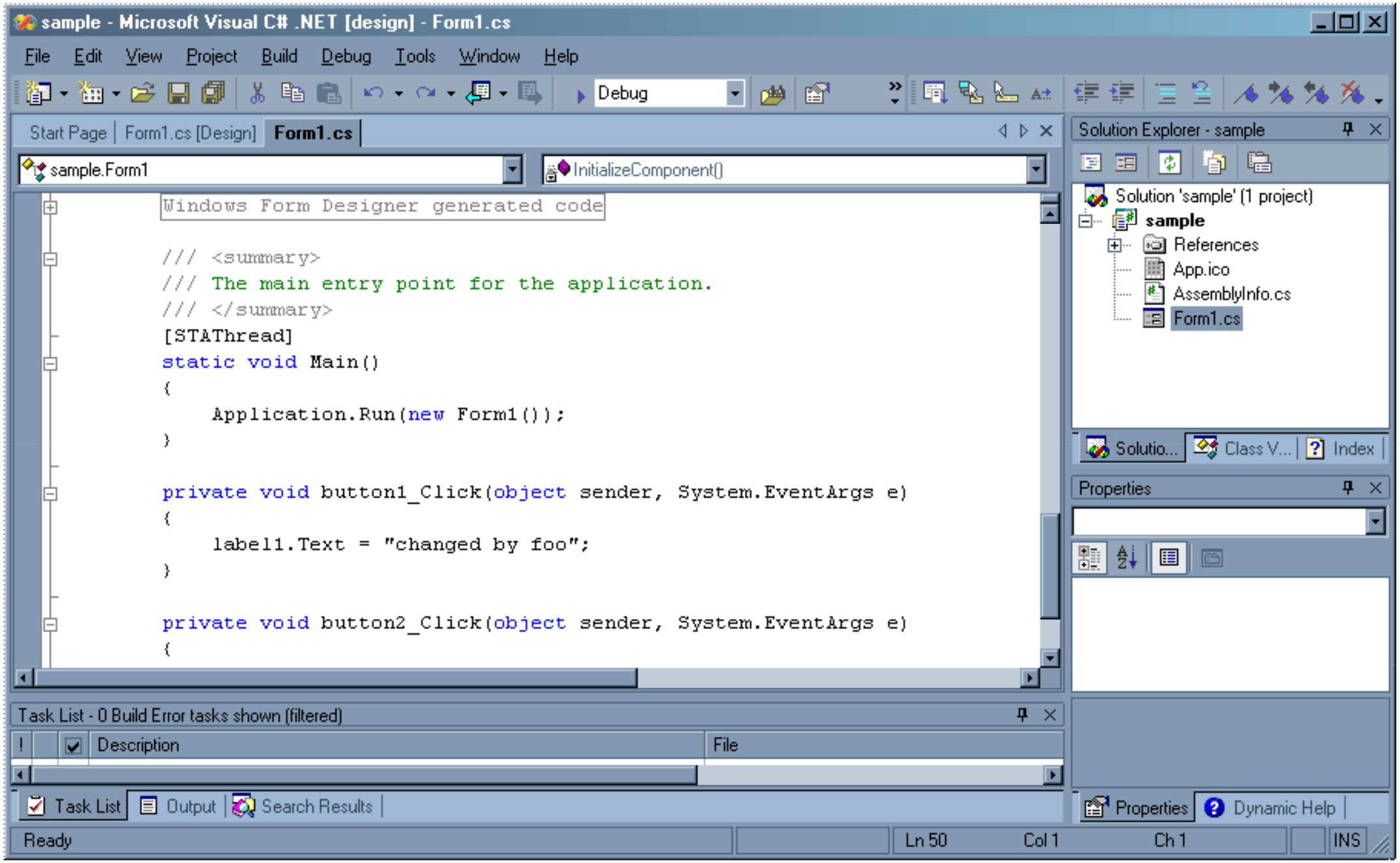
XAML: Example

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    // View Expense Report
    ....
}
```

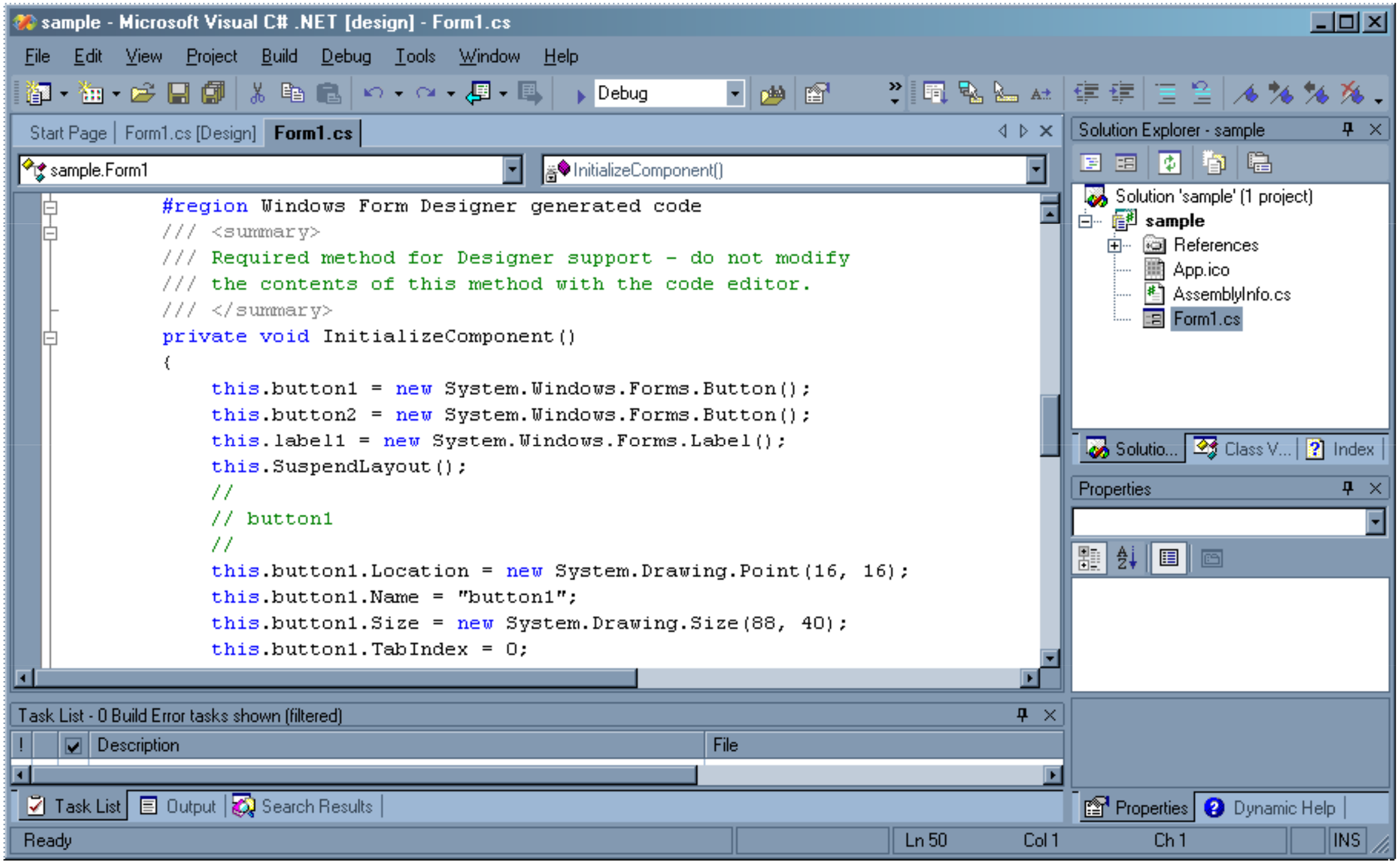
WinForms Designer



WinForms Designer



WinForms Designer



Windows: Evaluation

- Availability: 90% of all PC's!
  - but only for Windows Mobile/NT/XP/Vista/7
- Productivity: high (Microsoft Visual Studio)
  - high learning curve
  - Visual Basic has lower learning curve but has limited functionality
- Parallelism: yes for both external and internal

Windows: Evaluation

- Performance: good, but passing data between DLLs is a big overhead
- Graphics model: mostly vector based (since Vista)
- Appearance: fixed
  - Windows XP introduces themes ("look"), but you still can't change the "feel"

Windows: Evaluation

- Extensibility: fairly high
  - closed source but you can write your own extensions (DLLs)
- Adaptability: resource files
- Resource sharing: yes
- Distribution: no
- API structure: MFC is an extended C++; WinForms uses Managed C++, C#, Visual Basic ...

Windows: Evaluation

- API comfort: complicated, but extensible
- Independence:
  - MFC: high - document-view architecture (similar to MVC)
  - WPF: medium
- Inter-App Communication: everything from a clipboard to OLE

Further Reading

- Lots of books on Windows programming
- http://msdn.microsoft.com/
- http://www.winhistory.de/

Mic & Mac