Design and Analysis of Information Systems (MAS)

Mariusz Trzaska, Ph. D.
mtrzaska@pjwstk.edu.pl

Lecture 12
Implementation of Graphical User Interfaces

http://www.mtrzaska.com/
Outline

• An introduction
• The GUI libraries for
  • the Javy,
  • The C++,
  • The C#.
• A custom GUI library
• A GUI implementation using
  • Manually written code,
  • A visual editor,
  • A declarative way.
• A summary.

The slides make use of the www.wikipedia.org,
GUI Libraries

• Utilization of the libraries
  • Shipped with the language,
  • Third parties,
  • Custom implementations.

• The choice based on
  • Easiness,
  • Customization,
  • Performance,
  • Portability,
  • Price,
  • Aesthetics.
GUI Libraries for the Java

• Possibilities
  • AWT,
  • Swing,
  • SWT,
  • JavaFX

• Evaluation of the possibilities
  • Easiness,
  • Customization,
  • Performance,
  • Portability.
GUI Libraries for the Java (2)

- AWT (Abstract Window Toolkit; http://java.sun.com/products/jdk/awt/),
  - Published: 1995,
  - Uses the native widgets of the platform (different L&F on different OS),
  - Basic set of widgets (buttons, text fields, menus, etc.),
  - Events,
  - An interface between OS and the Java application,
  - Layout managers,
  - Clipboard and Drag&Drop,
GUI Libraries for the Java (3)

• AWT (Abstract Window Toolkit) – cont.,
  • An access to input devices such as a mouse or a keyboard,
  • Native AWT interface making possible rendering directly on the widget’s surface,
  • Access to the system clipboard (not on every OS),
  • Ability to execute some system applications (e.g. mail or an internet browser).
GUI Libraries for the Java (4)

• Swing ([http://www.javaswing.net/](http://www.javaswing.net/)),
  • Published: 1997,
  • Appearance and behaviour determined by the Java (the same on all platforms/OS). Rendered using the Java2D,
  • Extended (comparing to the AWT) set of widgets,
  • Look&Feel of the Swing application is defined by the selected theme (*pluggable look and feel*),
  • Independence from the platform,
  • Customization,
  • Components oriented,
GUI Libraries for the Java (5)

- Swing (http://www.javaswing.net/, ) – cont.,
  - Easy to customize:
    - Utilization of the existing elements during rendering process: *border*, *inset*, *decorations*,
    - Easy modification of properties (e.g. *border*),
    - Renderers,
  - Run-time customization,
  - „Light” *UI*. Own rendering mechanism.
  - Partial utilization of the AWT, e.g. `component.paint()`,
GUI Libraries for the Java (6)

• Swing (http://www.javaswing.net/) – cont.,
  • The MVC (Model-View-Controller) oriented. Most of the widgets has own models (defined by the Java interfaces), which determines the way of working with data.
  • The library is shipped with some basic implemented models.
  • Events:
    • “physical”, e.g. a mouse button click,
    • model oriented, e.g. adding a table row.
GUI Libraries for the Java (7)

- SWT (Standard Widget Toolkit; http://www.eclipse.org/swt/)
  - Created by the IBM, and currently developed by the Eclipse foundation,
  - Uses native OS mechanism for rendering controls,
GUI Libraries for the Java (8)

  - The Eclipse Foundation: SW T is an open source widget toolkit for Java designed to provide efficient, portable access to the user-interface facilities of the operating systems on which it is implemented.
  - Performance: SWT vs Swing:
    - Faster in rendering,
    - Slower in data manipulation (uses JNI - Java Native Interface).
  - The programs which use SWT are portable but they require dedicated version of the libraries (different even for the Windows x86 and x64). Not always available.
  - Very good implementation for the Windows.
GUI Libraries for the Java (9)

• SWT (Standard Widget Toolkit; http://www.eclipse.org/swt/) – cont.
  • In case that widgets under some OS do not provide a functionality, the SWT uses own implementation.
  • Does not use the MVC, but there is a possibility to use 3rd party libraries to support it (e.g. JFace),
  • Because of native widgets the customization could be hard to achieve.
  • Necessity for manual releasing the resources: the .dispose() method. (subclasses of the Resource: Image, Color and Font).
GUI Libraries for the Java (10)

• JavaFX
  • first release published on 2008-12,
  • desktop applications and RIAs (Rich Internet Applications),
  • replacement for the Swing library,
  • visual editor and FXML (XML format),
  • classes: Stage, Scene, Node (parts of a graph defining a scene),
GUI Libraries for the Java (11)

• JavaFX – cont.
  • events driven, but with bindable properties,
  • theming with CSS,
  • special effects (shadows, blur, mirror),
  • rich animations,
  • support for 3D graphics.
GUI Libraries for the C/C++

• Microsoft
  • Win32 API,
    • Low level,
    • The best performance,
    • Uses: GDI (*Graphics Device Interface*), *Common Dialog Box Library*, *Common Control Library*.
  • MFC (*Microsoft Foundation Class Library*),
    • “wraps” Win32 API,
    • More object-oriented but still lack of pure OO,
  • Borland (legacy?) products: OWL (*Object Windows Library*), VCL (*Visual Component Library*).
GUI Libraries for the C/C++ (2)

• Microsoft – cont.
  • WinForms (Windows Forms; http://windowsclient.net/),
    • Works also with the MS C#,
    • Distributed with the MS .NET,
    • Wraps the Win32 API, but much better then the MFC,
    • Very big number of controls,
    • Extended by the User Interface Process Application Block – Version 2.0 (introduces MVC).
GUI Libraries for the C/C++ (3)

- Microsoft – *cont.*
    - Works also with the MS C#,
    - The best capabilities in terms of visual appearance:
      - Classic controls with themes,
      - Complicated texts,
      - Images,
      - Video,
      - 2D,
      - 3D.
    - Big number of controls.
GUI Libraries for the C/C++ (4)

• Microsoft – c.d.

• WPF (Windows Presentation Foundation; http://windowsclient.net/) – cont.
  • Declarative way of creating GUIs (the XAML language),
  • A new tool for designers: MS Expression Blend.

<table>
<thead>
<tr>
<th></th>
<th>WinForms</th>
<th>PDF</th>
<th>WinForms + GDI</th>
<th>WMP</th>
<th>Direct3D</th>
<th>WPF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forms, Controls</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Complex text</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Images</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Video / Audio</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2D Graphics</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3D Graphics</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
GUI Libraries for the C/C++ (5)

• Qt (http://www.trolltech.com/products/qt/)
  • Started in 1991,
  • Portable (Unix, Linux, MacOS X, Windows, Windows CE, Java),
  • Uses non-standard extensions which are processed by the pre-processor,
  • Support for:
    • i18n,
    • SQL,
    • XML,
    • Threading,
    • Network programming,
    • Files.
GUI Libraries for the C/C++ (6)

- **Qt – cont.**
  - The last releases use OS technologies for visualizing controls,
  - A meta-compiler delivers information, which are not available in „pure“ C++. Such a solution allows e.g. asynchronous method calls.
- **Applications:**
  - KDE,
  - Opera,
  - Google Earth,
  - Skype,
  - Photoshop Elements,
  - Virtual Box.
GUI Libraries for the C/C++ (7)

• wxWidgets (http://wxwidgets.org/),
  • Started in 1992.
  • Portability: Mac OS, Linux/Unix (X11, Motif, and GTK+), OpenVMS, OS/2, AmigaOS.
  • Uses native OS capabilities,
  • Resource management,
  • Additional functionalities, e.g.
    • OpenGL support,
    • ODBC,
    • Network communication.
• GTK+ (The GIMP Toolkit, http://www.gtk.org/),
  • Portability,
  • Created for the GIMP,
  • Different engines for visualizing controls (e.g. emulation of popular L&F),
  • New version (GTK+ 2) contains better control rendering, new theme engine, Unicode support.
• Many other libraries...
GUI Libraries for the C#

- Microsoft WinForms,
- Microsoft WPF (*Windows Presentation Foundation*)
- wxWidgets (binding for the C#),
- GTK+ (binding for the C#),
- Mono (open implementation of the .NET including WinForms),
Your Own GUI Library

• Why do not use existing libraries?
• Selecting a model:
  • ”Physical” events,
  • ”Semantic” events,
  • Mixed,
  • Other?
• Set of widgets,
• Rendering widgets,
• Connecting events with methods (code).
Your Own GUI Library (2)

- Usually, such an implementation requires very low-level work:
  - Catch OS events: pressing/releasing keyboard buttons, cursor movements, pressing/releasing mouse buttons.
  - Drawing widgets using primitive operations (like drawing a line, rectangle, square, bitmap).
- Quite hard work.
The GUI Implementation

- Manually written code
- Utilization of a visual editor,
- A declarative way.
The GUI Implementation – manually written code

• The best possible control over the final result in terms of:
  • Functionality,
  • Performance,
  • Portability,
  • Usability,
  • Aesthetic.

• Usually requires a lot of work and decent knowledge.
• Harder (in some cases) modifications.
• Relatively slow development process.
• Error-prones.
• Usually it is better to use another approach.
• But sometimes this is the only way to achieve the right result.
The GUI Implementation – Visual Editors

• Different qualities of existing tools:
  • Only generation of the code. Manual modifications are lost during regeneration of the project.
  • Bi-directional code generation. The editor reflects manual modifications of the source code in the visual design.
  • Quite fast way of developing the GUI.
  • Less errors.
  • Instant visualization of the project.
It seems that using a good GUI editor is the best way for creating interfaces.

**Recommended solutions:**

- **Microsoft Visual Studio:**
  - Languages: C#, C++,
  - Libraries: WinForms, WPF, MFC
- **NetBeans IDE**
The GUI Implementation – Visual Editors (3)

• Recommended solutions – cont.:
  • Eclipse WindowBuilder Pro
    • SWT, Swing, GWT Designer
    • Currently part of the Eclipse

• IntelliJ IDEA IDE
  • [http://www.jetbrains.com/idea/](http://www.jetbrains.com/idea/)
  • Free edition (Community)
Some magic: maybe a „computer” will create a GUI for us?

Let’s focus on:
  • What to do,
  • Rather then how to do?

Different levels of automation:
  • A **semantic** declarativness: defining which model’s parts should have widgets,
  • A **component** declarativness: defining which widgets should be created.
It seems that this way of working will be more popular in the future.

Especially in case of GUls which are:
  - Quite common (visualization, functionality)
  - Business-oriented.

Finding a balance between:
  - A programmer involvement,
  - Universality of the solution.
• Existing commercial solutions, e.g. Microsoft XAML (Extensible Application Markup Language, Extensible Avalon Markup Language),
  • Heavily utilized in .NET, and especially in the WPF, Windows Phone and Windows Store Apps
  • Defining:
    • GUI items: 2D, 3D,
    • Data binding,
    • Events,
    • Special effects: rotation, animation.
  • Direct translation to the C# code.
  • A programmer has a lot of work defining those elements anyway.
The GUI Implementation – A Declarative Way (4)

• Microsoft XAML – c.d.

• Sample XAML code:

```xml
<Button Content="Click me">
    <Button.Margin>
        <Thickness Left="10" Top="20" Right="10" Bottom="30="/>
    </Button.Margin>
</Button>

(Page
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    x:Class="MyNamespace.MyPageCode">

    <Button Click="ClickHandler" >Click Me!</Button>

</Page>
```
More useful approach could be described in the following way:

- A programmer tells which parts of model (classes) should have GUI:
  - Attributes,
  - Methods.
- A system generates forms (windows with controls/widgets) allowing creating new data instances, visualization, modifications, etc.
- Optional additional descriptions:
  - Tooltips, labels
  - Widgets kinds,
  - ...
The Summary

- Modern programming languages are shipped with GUI libraries.
- It is also possible to utilize products provided by 3rd parties: both free and commercial. Sometimes they are faster, more portable or easier to use.
- GUI could be implemented using the following approaches:
  - Manually written source code,
  - Visual editors,
  - Declarative.
- It seems that currently the best way is to employ a visual editor.
- However, in the future it could change in favour of the declarative way.