Embedding Interface Sketches in Code

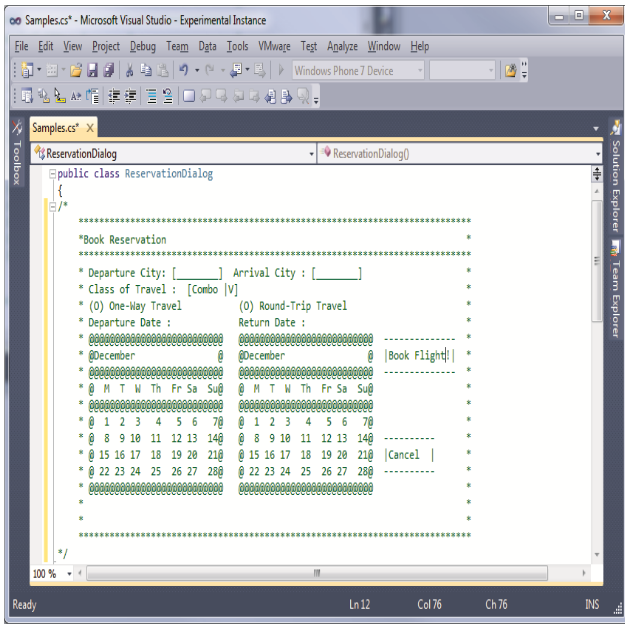
James Simpson, Michael Terry
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario N2L 3G1, Canada
{j2simpso, mterry}@uwaterloo.ca

ABSTRACT
This paper presents a user interface (UI) design tool, GUIIO, which uses ASCII text as its medium for rendering interface components. Like other UI design tools, GUIIO allows individuals to create and manipulate UI components as first-class objects. However, GUIIO has the advantage that its UI designs can be embedded directly within the program code itself. We implemented GUIIO as an extension to an existing development environment. As a result, developers can fluidly transition from editing code to editing the UI mock-up, with the text editor automatically switching its mode from code editing to UI editing as a function of the location of the cursor. By rendering UIs as ASCII art, GUIIO fills an important gap in the design, implementation, and revision of UIs by providing a highly portable and immediately accessible visual representation of the UI that embeds with the code itself.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

General terms: Design, Human Factors, Documentation

Keywords: design, development, user interface, documentation, ASCII art, plug-in


Samples.cs* - Microsoft Visual Studio - Experimental Instance
File Edit View Project Debug Team Data Tools VMware Test Analyze Window Help
Windows Phone 7 Device
Samples.cs*
Toolbox
Solution Explorer
Team Explorer
ReservationDialog
ReservationDialog()
100 %
Ready
Ln 12
Col 76
Ch 76
INS

```
public class ReservationDialog
{
/*
   **************************************************************************
   *Book Reservation                                                        *
   **************************************************************************
   * Departure City: [_______]  Arrival City : [_______]                    *
   * Class of Travel :  [Combo |V]                                          *
   * (O) One-Way Travel          (O) Round-Trip Travel                      *
   * Departure Date :            Return Date :                              *
   * @@@@@@@@@@@@@@@@@@@@@@@@@@  @@@@@@@@@@@@@@@@@@@@@@@@@@  -------------  *
   * @December                @  @December                @  |Book Flight!|  *
   * @@@@@@@@@@@@@@@@@@@@@@@@@@  @@@@@@@@@@@@@@@@@@@@@@@@@@  -------------  *
   * @  M  T  W  Th  Fr Sa  Su@  @  M  T  W  Th  Fr Sa  Su@                 *
   * @@@@@@@@@@@@@@@@@@@@@@@@@@  @@@@@@@@@@@@@@@@@@@@@@@@@@                 *
   * @  1  2  3   4   5  6   7@  @  1  2  3   4   5  6   7@                 *
   * @  8  9 10  11  12 13  14@  @  8  9 10  11  12 13  14@  ----------     *
   * @ 15 16 17  18  19 20  21@  @ 15 16 17  18  19 20  21@  |Cancel  |     *
   * @ 22 23 24  25  26 27  28@  @ 22 23 24  25  26 27  28@  ----------     *
   * @@@@@@@@@@@@@@@@@@@@@@@@@@  @@@@@@@@@@@@@@@@@@@@@@@@@@                 *
   *                                                                        *
   *                                                                        *
   **************************************************************************
*/
```

Figure 1: A GUIIO sketch of a flight reservation system, complete with date picker calendar widgets.

INTRODUCTION

A wide range of tools and techniques exist to design and document user interface (UI) designs. Informal, low-fidelity sketches can be created using traditional media or sketch-based tools and software [2, 3], while high fidelity representations can be produced using commonly available user interface builders. Collectively, this spectrum of tools enables one to focus exclusively on the visual design of the interface.

While it is desirable to focus on the visual representation of an interface separate from the code that powers it, the two are intricately linked. However, despite this tight coupling, the two representational forms typically require different toolsets for creating and editing content. This separation of an interface design from its underlying code can be problematic in the case of distributed software development, where some team members may not have access to the same tools. For example, software developers may not have access to design tools. In situations such as these, designers may need to create static screenshots to represent their designs. However, these screenshots cannot be edited and manipulated as easily as in the design tool that created them.

This problem of heterogeneous toolsets has been observed in the open source community, where communication consequently defaults to text-based media [4]. In the context of designing user interfaces, this can lead to the use of ASCII art to represent UI designs [4].

The use of ASCII art for interface designs yields a number of benefits: designs can be easily embedded in existing communication media (such as email); they can coexist with the code itself (typically inside comments near the code implementing the UI); and the designs can be edited by anyone with a text editor. However, editing ASCII art is tedious and time consuming since text editors operate at the level of *characters*, rather than the higher-level objects represented by the UI design (i.e., components). For example, moving a UI widget in ASCII art requires manually shifting and deleting text across rows to maintain the layout of the other components. Additionally, users must manually create the UI components' look using the text editor, which may require experimentation with various characters to achieve an acceptable visual representation. To address these issues, we developed GUIIO, a UI designer that uses ASCII text as its means of rendering user interface designs.

THE GUIIO PLUG-IN

GUIIO is a plug-in developed for Visual Studio 2010 (Figure 1) that integrates with the source-code editor, effectively embedding a UI design tool in the text editor itself.

UI components can be added as comments to source code by invoking a keyboard shortcut that presents the developer with a context menu listing possible components. When a component is selected from the menu, GUIIO inserts an ASCII art rendering of the component into the source code. GUIIO supports a variety of common UI components, including windows, buttons, text fields, list boxes, and progress bars.

Like other graphical interface builders, GUIIO allows properties of UI components to be set. For example, the window component contains properties such as height, width, and whether close, minimize, and maximize buttons should be rendered with the component. These properties are set through direct manipulation of the ASCII art representation of the component. For instance, pressing the space key inside a window causes GUIIO to increase the width of the window.

GUIIO supports positioning of components through a similar interaction paradigm. Components can be shifted by one or more units in any direction through the use of the return, space, and backspace keys. As an example, a component can be moved to the right by positioning the caret immediately before the rendered component and pressing space. GUIIO automatically understands this interaction as a request to shift the component by a unit and automatically renders the UI again to reflect the change. Compared to manual editing of ASCII art text, the developer no longer has the tedious task of realigning shifted text when they modify aspects of their design.

Benefits

ASCII art as a medium for the design and implementation of UIs has several distinct advantages that make it a worthwhile complement to existing representational forms. One of the most obvious benefits is that it bridges the graphical world of design and the textual world of development, allowing interface designs to be incorporated as comments in source code. This capability allows developers to create documentation that directly expresses the graphical nature of their UIs. Since the designs are implemented in source code, it also means that interface designs can always accompany the underlying implementation as represented by source code. This feature reduces the dependence on design tools (which every developer may not have), and increases the likelihood that documentation and designs are updated in response to changes in the code.

This design medium is also a highly compatible medium, as designs can be easily integrated into e-mails, newsgroups, and other online communication channels, without the need for special tools to view, modify, or annotate the design. This feature ensures that developers and designers alike can easily discuss interface designs through existing communication channels, without the need for specialized tools.

Finally, the rough nature of renderings created in ASCII art gives interface designs a very coarse appearance. It has been shown in Buxton, that giving designs a rough rendering encourages critique [1]. This critique could lead to better interface designs, particularly if this medium is used in online discussions and debates of proposed designs.

CONCLUSION

ASCII art as a medium for interface design has received little attention, yet it confers a number of advantages compared to existing mediums. Its text-based nature enables designs to be directly embedded with the working copy of code. Similarly, designs can be included in-line with existing communication mediums. Recognizing these benefits, GUIIO provides tools for first-class creation and editing of ASCII-based user interface designs.

REFERENCES

1. Buxton, W. 2007. *Sketching User Experiences: getting the design right and the right design*. Elsevier Inc., San Francisco, CA

2. Mauro Cherubini, Gina Venolia, Rob DeLine, and Andrew J. Ko. 2007. Let's go to the whiteboard: how and why software developers use drawings. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (CHI '07). ACM, New York, NY, USA, 557-566.

3. James A. Landay and Brad A. Myers. 1995. Interactive sketching for the early stages of user interface design. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (CHI '95), Irvin R. Katz, Robert Mack, Linn Marks, Mary Beth Rosson, and Jakob Nielsen (Eds.). ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 43-50.

4. Twidale, M. B. and Nichols, D. M 2005. Exploring Usability Discussions in Open Source Development. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences – Volume 07 (*January 03- 06, 2005). HICSS. IEEE Computer Society, Washington, DC, 198.3.