Alternative GUI for Interaction in Mobile Environment

Juraj Švec*
Department of Computer Science and Engineering
Czech Technical University in Prague
Prague / Czech Republic

Abstract

Standard personal digital assistant (PDA) graphical user interfaces (GUI) become unusable if used in complex applications, like graphical editors containing tens or hundreds of functions and manipulating with complex data structures. It is necessary to investigate new user interface design patterns and simplify the development process of complex mobile applications. This paper researches this area, uncovers the reasons why the common user interface is unusable, improves and designs new user interface components and finally implements the designed features and tests them.

Keywords: Graphical user interface, Personal digital assistant, Pocket PC, Pie menu, Mobile device, Vector editor, Usability testing

1 Introduction

The PDA has grown into a tool of every day usage. Today the PDA devices are used in many applications. Unfortunately, there are many problems with usability of the standard graphical user interface (GUI) on PDA devices.

With miniaturization of PDA dimensions the space available for GUI components becomes critically small. However, only few manufacturers react appropriately and create usable user interfaces. Standard PDA software is frequently only a version of the same program coming from desktop systems converted a PDA platform (e.g., smaller buttons, fewer functions). Especially applications with more complex user interface like the graphical editor converted in this way to a PDA platform become unusable.

We have inspired ourselves mainly by a set of standard user interface problems revealed on the basis of usability test proceeded in the Mobile knowledge management research project (MUMMY) [1]. We have got inspiration also from another research projects in the human computer interaction and ergonomics:

- Fitts' law [2], which describes the information capacity of the human motor system.
- Accot-Zhai steering law [3], which describes scale effects in steering law tasks.
- Pie menu theory [4], which describes the advantages of pie menu, by using various pointing devices.

We have designed proposals for new user interface improvements for the Pocket PC PDA platform based on the research projects described above.

2 User Interface Problems

According to the study of the MUMMY project [1], the standard graphical user interface of the Pocket PC device has many problems, which violate good user interface design practices. These problems originate not only from the PDA device restrictions (e.g., small screen, different pointing device – stylus) but also from inappropriate usage of the graphical components and features applied from the desktop computers in the unchanged form.

These problems resulted into the usability issues identified by usability test performed in the framework of MUMMY project. The most problematic widgets were menu and toolbar.

The standard menu does not support icons and custom graphic symbols. It often contains many disabled gray menu items. The farther is the menu item, the bigger problem has the user to properly select it.

It is possible to display only one standard toolbar on the PDA display. The standard toolbar has a static position – it is placed on the bottom screen edge next to the main menu. The standard toolbar buttons and graphic are strictly determined by native programming framework.

2.1 Small Display

One of the biggest problems in the area of complex GUIs is the small display, see Figure 1. It directly influences all other aspects of the user interface and forces us to invent new ways how to save space for visualization of important information. Currently, there is no satisfactory solution to this problem. It is necessary to define new efficient UI patterns based on the usability tests.

A common mistake is hard-coded user interface layout without the possibility to adapt it to the user needs, see standard Pocket PC menus on Figure 1, which have fixed position and size. As the users have different preferences they are frequently forced to change their behavior according to the hard-coded design.

* j.svec@fee.ctup.cz

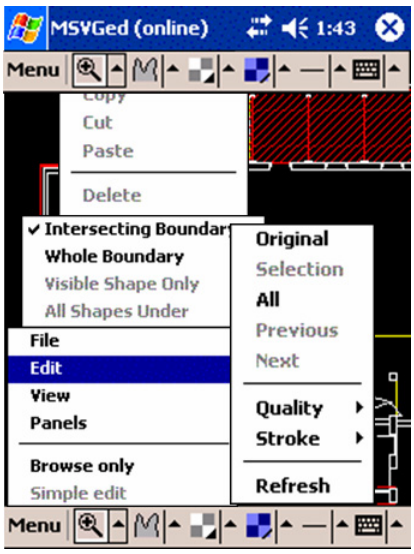

Figure 1: Problem of small screen of the graphical editor. The displayed plan is entirely covered by the UI widgets.

2.2 Confusing Interface

Mobile user interfaces often contain confusing GUI components leading to the user confusion. This is mostly caused by small space, smaller color depth and different interaction methods. Designers often provide us with strange-looking overcrowded and confusing interfaces.

On the basis of the usability tests performed in the framework of MUMMY project [1] we will demonstrate the usability issue. As an example we have chosen the standard toolbar from the Pocket PC platform, see Figure 2. If we look at the buttons we can see that some of them have the ability to show another submenu and this ability is visualized by the small black arrow on the button side. The problem is that it is not visible which button the arrow belongs to and according to the separator position we can think that it would be the button on the right. Unfortunately, it is the button on the left side behind the separator. This is an example of a new user interface pattern that was derived from desktop UI pattern without taking into account PDA platform specifics.

Figure 2: Confusing toolbar from Pocket PC platform

This kind of mistakes would be efficiently eliminated by executing the usability test. The user test can even help to develop new UI patterns.

3 Proposals for Improvement

All our new user interface ideas originate from the application of Fitts' law [2]. Every interaction component that can be customized in its size and position has a Fitts' law advantage against its static predecessors.

In the following sections we will introduce two UI concepts – floating toolbar and context pie menu. Both concepts solve the hard-coded layout, small display and even confusing GUI issues.

3.1 Floating Toolbar

Our floating toolbar concept is based upon the standard toolbar model – user has a panel with some icons and they can be pressed and execute an action. We have improved the toolbar concept for the PDA device. It is drawn clearly, without any confusing separators that mislead users – the button always looks like a button. It is possible to have many toolbars on the screen and the user is able to move them (Figure 3-d), hide them (Figure 3-b) and resize them (Figure 3-a). If it is partially moved off the screen (Figure 3-d) it gently slides to the side leaving a handle available for the user next to the screen border (Figure 3-b). After clicking on the toolbar it slides back onto the screen and if it is not used for a certain time period, it slides back out of the screen again.

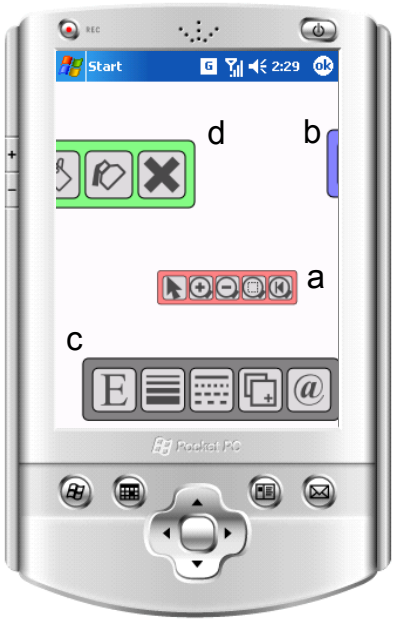

Figure 3: Floating toolbar examples

It is possible to manage the overall interface layout and appropriately react on the changing scene complexity just by hiding, showing or moving toolbars. The move use-case is described in Table 1. The toolbar can be zoomed by dragging its left or right side (Table 2). Another feature is the ability to change the background color of the toolbar, so the user can easily determine the toolbar on the screen even if it is hidden behind the screen border. This feature helps the user to remember in which toolbar which function is located.

1	click and hold down the stylus anywhere except resize handles
2	move the stylus (also moves toolbar)
3	release stylus (releases toolbar)
4	if is the toolbar beyond slide limit, slide it out

Table 1: Floating toolbar move use-case

1	click and hold down the stylus on resize handle
2	move the stylus (also resizes toolbar within limits)
3	release stylus (releases toolbar)

Table 2: Floating toolbar resize use-case

The ability to customize the user interface (as described above) in such a way improves its clarity and makes the learning curve steeper.

3.2 Context Pie Menu

The idea of our context pie menu is to replace the standard menu component with something easier to remember and follow. The pie menus allow the user to follow continuously the use-case, without any disturbance by searching for hidden tools and menu functions. It allows the user to focus entirely on the workflow and thus increase productivity. Standard menus suffer from a lot of usability issues. Not only the standard menu bar takes place on the screen, but if the user tries to follow the menu and find the function it often covers the whole screen and the user looses the context in which the user has been working.

A pie menu, see Figure 4, invented by Don Hopkins, is a circular popup menu where selection depends on direction. A pie menu is made of several "pie slices" around an inactive center and works best with stylus input, and well with a mouse [4].

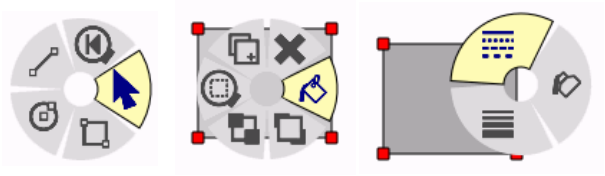

Figure 4: Context pie menu

For the beginner, pie menu is easy to use because it is a self-revealing gesture interface (see use-case Table 3). It shows what we can do and directs us how to do it. By clicking and popping up a pie menu, looking at the labels, moving the stylus in the desired direction, then clicking to make selection, we learn the menu and practice the gesture to "mark ahead". With a little practice, it becomes quite easy to mark ahead.

Beginner	
1	click and hold down the stylus
2	wait for a while causing the pie menu to display
3	move the stylus into the desired slice
4	release stylus – execute desired action
Expert (rely on muscle memory)	
1	click and hold down the stylus
2	move stylus onto the desired action (direction)
3	release stylus

Table 3: Pie menu use-case

Although the stylus was identified as the best pointing device for interaction with pie menu, it is rarely used on PDA.

3.3 Traditional and New User Interface Comparison

It is possible to display only one standard toolbar on the PDA display, the floating toolbar is not limited in count. The standard toolbar has a static position – it is placed on the bottom screen edge next to the main menu. The floating toolbar is not limited by either position or size. It can change the size and can be also hidden behind the screen edge to save some valuable space. The standard toolbar buttons and graphic are strictly determined by a native programming framework. The floating toolbar is not limited in any way – its design is fully in the hands of the programmer.

The standard menu does not support icons and custom graphic symbols. It often contains many disabled gray menu items. The farther is the menu item, the bigger problem has the user to properly select it. The context pie menu is fully configurable. It allows custom graphics and icons. The content is intelligently designed not to contain any grayed menu items. Every pie menu item has the same visual priority. Pie menu item selection improves over time because of muscle memory.

The new components are certainly better than the native components. The user confusion rises from the routine developed with the native user interface and the desktop user interface.

4 Test Plan

The goal of the testing was to check the usability of our new GUI components. Usability testing is generally focused on qualitative analysis. Therefore measurement of the user performance (e.g., time to task completion) serves solely for indicative purposes. These measures cannot be used for comparison of different usability tests. The usability tests are mostly focused on new designs and are also intended to identify bad designs.

The usability test was performed to evaluate our designed components – floating toolbar and context pie menu. We intended to test also the affordance of our new GUI components. Therefore the test participants were provided with no help or explanation of components functionality before and during the test. The usability test was not intended to test the logical layout of widgets or understandability of the icon symbols.

For testing purposes we had to develop a powerful user interface framework in which we could design freely our new GUI components. Standard displaying methods and rendering engine supported by Pocket PC and Windows Mobile 2003 do not fulfill our requirements. The rendering engine allows us to draw aliased graphics only. It does not support transparency and matrix transformations. We are limited by the speed and non object-oriented programming interface.

Because of these limitations we had to build our own rendering library based upon modular concept, which allows us to change the individual parts of the library in the future (e.g., better widgets library, faster graphical engine).

4.1 Participants

The user interface components will be used by people with experience in graphical editors (bitmap editor, vector editor, CAD/CAM, 3D modeling) on the desktop systems. For testing these features, the most important target audience consists of two typical groups. People using PDA and people without PDA experience. All these people should have at least some experience with graphical software or another complex interface, because the new components are mostly for them.

It is important that the user does not have any serious disability, because we focus on the end-user working in mobile environment with high pixel density displays and stylus. The user should be able to work with stylus (hold it and use it) and should have no serious visual impairment (can use the small screen and recognize drawings on it).

Optimal number of participants for usability testing ranges from five to ten. We have recruited five participants (four men and one woman) who fulfilled the requirements specified above.

4.2 Task List

The test was divided into two tasks categories, interaction with the new UI components and interaction with complex graphical editor. The test of graphical editor was focused on the scene creation and modification of existing scene.

Each user had to solve following seven tasks:

- Task 1: Browse the user interface
- Task 2, 3: Reading the text hidden behind toolbars
- Task 4: Finding and executing function
- Task 5: Drawing simple objects
- Task 6: Drawing complex scene
- Task 7: Modifying complex scene

The time needed to accomplish the tasks was about 30 minutes. The first four tasks were intended to investigate the affordance and learning curve of the tested components. The following three tasks (Task 5, 6, 7) tested the usability of components in context of a real situation (drawing and modifying vector graphics scene).

4.3 UI Framework Implementation

Our user interface framework is placed between standard graphical components and native rendering libraries (GAPI, DirectX), see Figure 5. As mentioned in the beginning of this chapter, the framework position is determined by our needs to improve future applications GUI.

Our user interface framework consists of three separate layers. The first layer is a graphic library. It is the low level implementation of custom graphical algorithms in the C programming language. This layer operates with memory abstraction buffers to ensure platform independence. Above this layer the object-oriented wrapper library is placed. This wrapper is implemented in the .NET Compact Framework C# language. Finally, the third layer is an object-oriented component framework inspired by the Java Swing.

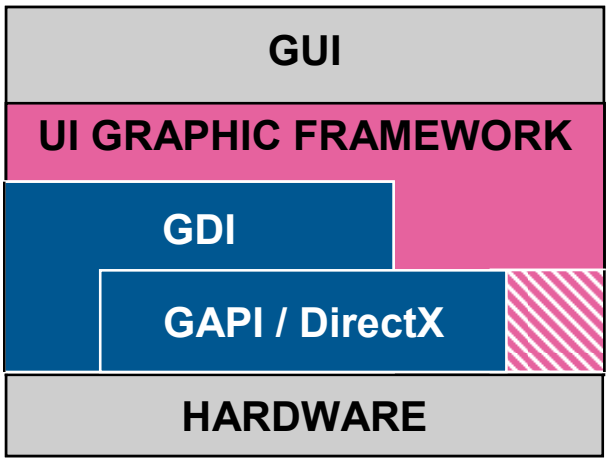

Figure 5: Layers of PDA Graphics Architecture

The component library has been designed with design patterns in mind to be extensible and easy to maintain. The graphical output of the custom library is bound to the memory window allocated by the standard image component. This allows also interaction with native Pocket PC components like onscreen keyboard or transcriber.

The project was implemented in MS Visual Studio 2005. The user interface was tested in the MS Pocket PC Emulator 2003 and on the HP iPAQ H2200 Pocket PC device.

5 Test Results

The first four tasks focused on investigation of components affordance and learning curve were finished

within 10 minutes. Users rated these tasks as easy and fast to accomplish.

Users understood quickly the floating toolbar functionality without any help. The floating toolbar component control and behavior was found much more affordable than the standard toolbar of Pocket PC platform (see section 2.2). As regards the tasks 5, 6, 7 the users have shown efficient usage of floating toolbar by exploiting all its functionalities. The floating toolbar was accepted by all users.

The context pie menu was not used as intended. The users did not recognize the situations where it could be used. They were unable to identify pie menu role in the workflow defined by the tasks 5, 6, 7. The behavior of the context pie menu was problematic, see the findings list bellow. These issues led the users to prefer the usage of floating toolbars.

5.1 Findings List

The main recommendations resulting from the usability testing are:

- Adjust pie menu behavior – every user expected that the pie menu stays on the screen after it appeared, see new use-case Table 4.

Beginner	
1	press and release the stylus, causing the pie menu to display
2	move stylus into the desired slice
3	click the desired action
X	user can exit by clicking the centre
Expert (rely on muscle memory)	
1	click and hold down the stylus
2	move stylus onto the desired action (direction)
3	release stylus

Table 4: New context pie menu use-case

- Adjust toolbar behavior – add a special “hide me” button to the sides of the toolbar – small arrows could be there to indicate the function (see Figure 6). Change the toolbar behavior after sliding back on the screen. Let it stay there until a function is executed or until the “hide me” handle is clicked. Introduce the self organization function of toolbars that are slid out behind the screen edge. Organize the toolbars to not overleap. Introduce the ability to slide the toolbars behind the top and bottom edge of the screen.

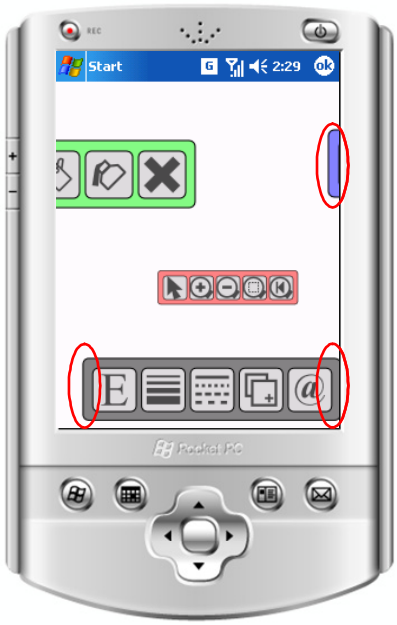

Figure 6: Floating toolbar enhancements. Red ellipses mark places for the “hide me” handles, which replace the automatic toolbar hiding feature.

- Disable toolbar while moving – every user was afraid that s/he will click the icon (execute icon function), during the movement of the toolbar.
- Show resize handles on the toolbar – one user found the resize function intentionally, three users by accident and one user did not even find this feature.
- Make user interface configurable – each user recommended the ability to somehow configure toolbars timeout or context pie menu timeout.

6 Conclusions and Future Work

The task to find problems in GUI for complex graphical applications in PDA environment was met. We have found several problem groups of different importance: inadaptable UI layout, confusing UI components, weak rendering capabilities of Pocket PC platform and missing PDA UI patterns.

We analyzed possibilities of user interface improvement and designed successfully new interface components. We also implemented a prototype of specialized vector editor, to test the overall widgets interaction in application context. The designed user interface solved the major problems and passed the usability testing. Results of the usability testing confirmed that the newly designed floating toolbar is usable and accepted by the users. The pie menu had worse results, which were caused mainly by the unexpected behavior and impossibility to identify its role in the workflow. They also discovered several minor problems that can be solved easily in the future. The main task to design and implement alternative user interface was fulfilled successfully.

As future work, primarily the implementation of the interface components should be changed according to the recommendations of the usability test results. After the user interface is functional and usable it will be integrated into a real vector editor application.

References

[1] MUMMY project - Mobile knowledge management (funded by European Commission, IST-2001-37365), http://www.mummy-project.org/

[2] Paul M. Fitts, *The information capacity of the human motor system in controlling the amplitude of movement*. Journal of Experimental Psychology, volume 47, number 6, June 1954. (Reprinted in Journal of Experimental Psychology: General, 121(3):262--269, 1992)

[3] Johnny Accot and Shumin Zhai, *Scale effects in steering law tasks*. In Proceedings of ACM CHI 2001 Conference on Human Factors in Computing Systems

[4] Jack Callahan, Don Hopkins, Mark Weiser, Ben Shneiderman, *An empirical comparison of pie vs. linear menus*. Proceedings of ACM CHI Conference on Human Factors in Computing Systems, 1988