Touch Interface Framework for GIS based C4I Systems

Aleksandar Milosavljević[1], Dejan Rančić[2], Aleksandar Dimitrijević[3] and Vladan Mihajlović[4]

***Abstract* – This paper presents *GinisUI* framework for rapid development of advanced graphical user interfaces with features such as control transparency, automatic arrangement and zooming. *GinisUI* is specially designed and implemented for appliance in GIS based C4I systems with touchscreen use. The framework relies on XML for specifying interface components and their layout. Because C4I systems combine GIS functionality with Virtual Reality (VR) their applications demand both 2- and 3-dimensional visualization. To enable seamless visual transition between these two modes, UI component visualization in *GinisUI* is implemented using both GDI and OpenGL drawing.**

***Keywords* – Touch Interface, Framework, GIS, C4I, XML.**

I. INTRODUCTION

A geographic information system (GIS) is special type of computer-based information system tailored to store, process, and manipulate geospatial data [1]. The ability of GIS to handle and process both location and attribute data distinguishes GIS from other information systems. It also establishes GIS as a technology important for a wide variety of applications [2]. For many years GIS has been considered to be difficult, expensive, and proprietary. The advent of graphical user interface (GUI), powerful and affordable hardware and software, and public digital data has broadened the range of GIS applications and brought GIS to mainstream use in the 1990s [2]. Despite the success of various GIS applications, their current use and usability is still constrained by overly complex user interfaces [3].

The acronym C4I is used to represent the following group of related military functions that enable the coordination of operations: Command, Control, Communication, Computers, and Intelligence [4]. *Command* and *Control* refers to the ability of the military commander to direct his forces. The addition of *Communications* to the grouping reflects the fact that communications is required to enable this coordination. In modern warfare, *computers* are also a key component, while *Intelligence* is the knowledge relevant to the coordination of forces.

One important capability that C4I systems provide commanders is situational awareness, i.e. information about the location and status of enemy and friendly forces [5]. Therefore, building a C4I system must rely on some GIS functionality. Certainly, the most important is mapping, positioning, map navigation, and overlay analysis using different layers of information to create a map.

Users of C4I systems are commanders at different levels of army command chain. Some lower levels of the command require embedment of C4I systems into combat vehicles. Such conditions require use of touchscreen for human interaction with the system.

To enable rapid development of an effective touchscreen graphical user interface for GIS based C4I systems we relied on a design and implementation of a GUI user interface framework.

Framework is a generic term for a powerful object-oriented reuse technique that typically emphasizes the reuse of design patterns and architectures [6]. There are two common definitions of an application framework [6]. The first defines an application framework as a reusable design of the entire, or part of a system represented by a set of abstract classes and the way their instances interact, while the second states that an application framework is the skeleton of an application that can be customized by an application developer. These definitions are complementary, not conflicting, since the former describes framework from the design perspective, whereas the latter describes it from the functional viewpoint.

The paper is organized as follows: Section 2 gives brief overview of user interface paradigms. Section 3 discusses design issues and the architecture of *GinisUI* framework. Section 4 presents GUIDL – XML language for describing *GinisUI* user interfaces. Section 5 considers presentation of a tank C4I system touch interface, and Section 6 summarizes the results achieved.

II. USER INTERFACES

A *user interface* is the aggregate of means by which people (the users) interact with a particular machine, device, computer program or other complex tool (the system) [7]. A *graphical user interface* (GUI) is a method of interacting with a computer through a metaphor of direct manipulation of graphical images and widgets in addition to text. *Touch interfaces* are graphical user interfaces using a touchscreen display as a combined input and output device.

[1]Aleksandar Milosavljević is with the Faculty of Electronic Engineering, Aleksandra Medvedeva 14, 18000 Niš, Serbia, E-mail: alexm@elfak.ni.ac.yu

[2]Dejan Rančić is with the Faculty of Electronic Engineering, Aleksandra Medvedeva 14, 18000 Niš, Serbia, E-mail: ranca@elfak.ni.ac.yu

[3]Aleksandar Dimitrijević is with the Faculty of Electronic Engineering, Aleksandra Medvedeva 14, 18000 Niš, Serbia, E-mail: aks@elfak.ni.ac.yu

[3]Vladan Mihajlović is with the Faculty of Electronic Engineering, Aleksandra Medvedeva 14, 18000 Niš, Serbia, E-mail: wlada@elfak.ni.ac.yu

Most modern general-purpose GUIs consists of graphical widgets such as windows, menus, radio buttons, check boxes, and icons, and employs a pointing device (such as a mouse, trackball, or touchscreen) in addition to a keyboard. Those aspects of GUIs can be emphasized by using the acronym WIMP (Windows, Icons, Menus, and Pointing device). GUIs that do not follow WIMP paradigm are most notably found in computer games. Another research direction in advancement of GUI is the Zooming User Interfaces (ZUI) [8].

The raising popularity of touch interfaces can be described by a fact that a touchscreen provides the simplest, most direct user interface with a computer [9]. Properly programmed, a touchscreen interface can be intuitive, requiring no learning curve for first time users. Touchscreens are also rugged enough to stand up to harsh environments where keyboards and mice can become damaged. Touchscreens ensure that no space is wasted because the input device is completely integrated into the monitor.

III. GINISUI FRAMEWORK

GinisUI is an application specific GUI framework specially designed and implemented for appliance in GIS based C4I systems with touchscreen use. General concepts underlying the design and the implementation of this application framework are already used in our GIS application framework *Ginis* [10].

GinisUI framework aimed domain is a C4I system embedded within combat vehicle and integrated with existing command, control, and communication subsystems. Harsh environment, and usually limited vehicle inner space require use of relatively small touchscreen interfacing devices.

Because these systems are GIS based, their basic purpose is to display layered map of surrounding area with a variety of geographically located information. So the problem was to develop a touch interface that fits on a relatively small screen already overcrowded with other information. Furthermore, pointing with a finger to a screen in the environment of a shaking combat vehicle requires buttons big enough to avoid mistakes.

Our solution to the problem was to develop a user interface that will enable:

- Transparency,
- Zooming and
- Automatic arrangement of controls.

Transparency is introduced to enable integration of user interface into a full-screen geographic map. This feature is combined with a zooming feature that represents enlargement of buttons when they get input (pointer) focus.

To further preserve an information view area, from the user interface, we included control showing/hiding and automatic arrangement. This feature enables design of a user interface that can automatically reconfigure to a current mode of an application.

GinisUI framework is designed and implemented to enable rapid development of user interfaces that fulfil identified requirements. To enable rapid development, the framework relies on XML for specifying interface components and their layout.

Logical architecture of *GinisUI* framework is based on relatively simple and known concepts (Fig. 1). Class `GUI` is the main class that acts as an interface between an application and the user interface (UI). It is derived from `GWindow` class, and it holds top-level UI controls (instances of `GControl` subclasses `GToolbar`, `GButton`, `GText`, and `GPicture`).

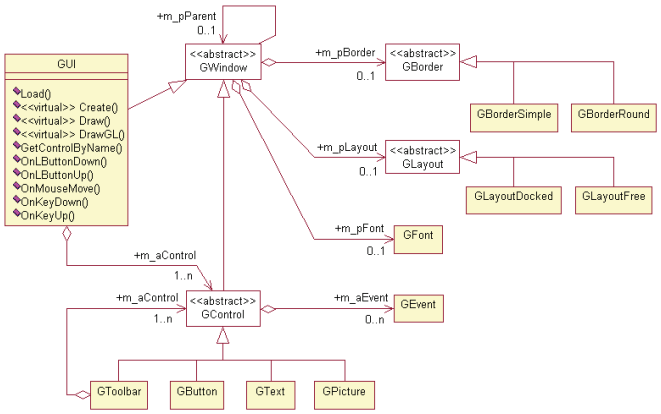

Fig. 1. Logical model of *GinisUI* framework

Characteristics of a UI control window are defined through `GWindow` class. General attributes of a window are background colour or background image and window transparency. Optional attributes are border, layout and font.

An implementation of different window border types is done through `GBorder` abstract class and their subclasses. Class `GBorderSimple` implements rectangular border type, while class `GBorderRound` is obliged for rounded window borders. Further extension of the framework with other border types can be done by implementation of appropriate `GBorder` subclasses.

Abstract class `GLayout` acts as a template for implementation of different window layouts. *GinisUI* framework currently enables docked (class `GLayoutDocked`) and free layouts (class `GLayoutFree`) of UI controls in corresponding parent windows. Controls can be docked on the top, bottom, left or right of the parent window, or they can be freely placed on some absolute position.

A style, size and colour of a text, that is to be displayed within `GText` control, are defined by `GFont` object attached to a control or some of its parent windows. This approach ensures range of possibilities from setting different font each control, to using single font for all controls in the UI.

As we already stated, `GWindow` part of a UI control object deals with display and arrangement of controls. Behaviour of a control (e.g. button) is implemented by event catching mechanism realized in `GControl` part of the object. Different event types (e.g. mouse button down, mouse move, keyboard button down, etc.) are defined by `GEvent` objects that can be attached to a UI control. When event happen, an application should call corresponding `GUI` interface methods (*`OnLButtonDown`*, *`OnMouseMove`*, etc.) that returns a pointer to a control that caught the event. An application that uses this framework, knowing the control name, should then handle that event.

Because C4I systems combine GIS functionality with Virtual Reality (VR) their applications demand both 2- and 3-dimensional visualization. To enable seamless visual transition between these two modes, UI component visualization in *GinisUI* is implemented using both GDI (`GUI` class `Draw` method) and OpenGL drawing (`GUI` class `DrawGL` method).

Described framework defines only basic building blocks needed for a user interface. Building a concrete user interface for some application additionally requires explicit definition what and how these blocks create the interface. For encoding of this definition we use XML. In order to constrain content of these XML documents, we specified appropriate XML language named *Ginis User Interface Definition Language* (GUIDL).

IV. XML LANGUAGE GUIDL

GUIDL represents XML language specified using *XML Schema Definition Language*. A valid GUIDL XML document represents definition of a user interfaces that can be realized using *GinisUI* framework.

The structure of GUIDL and *GinisUI* logical model are closely related. All classes shown in Fig. 1 have their corresponding XML elements in the language specification. To enable setup of instantiated object properties from XML definition, these classes implement virtual method `Create`.

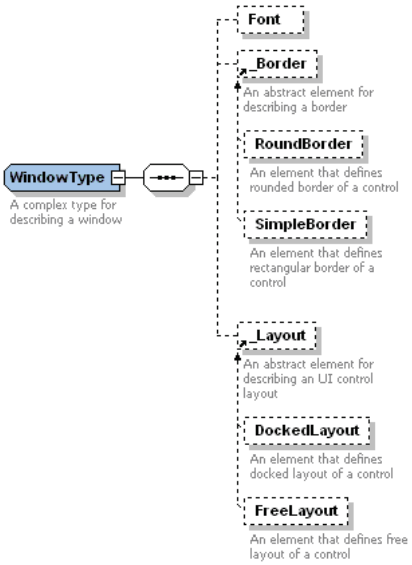

Fig. 2. Structure of an XML definition of a window

Fig. 2 shows structure of a complex XML element type `WindowType` that corresponds to `GWindow` class. Contained elements `Font`, `_Border` and `_Layout` are used to optionally setup font, border and layout properties of a control window. Elements `_Border` and `_Layout` are abstract and serves as placeholder for appropriate substitutes (e.g. `RoundBorder` and `DockedLayout`).

Complex type `ControlType`, shown in Fig. 3, is defined as an extension of `WindowType` and corresponds to `GControl` class. `ControlType` additionally defines events that control shall catch and different control titles for multilingual support.

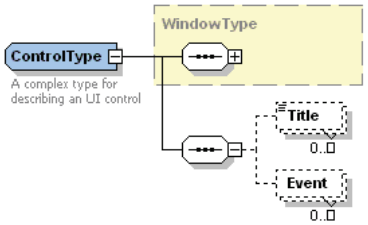

Fig. 3. Structure of an XML definition of a control

Finally, knowing structures of window and control element types, we can discuss structure of a root element `UIDef` (Fig. 4) that corresponds to `GUI` class. The root element inherits structure of `WindowType` that is used for specification of an UI background window. Additionally, it can contain several control elements defined through substitution group `_Control`.

Elements `Toolbar`, `Button`, `Text` and `Picture` (Fig. 4) are used to specify properties and structure of actual user interface controls. All these elements extend previously described `ControlType` (Fig. 3) with additional properties defined in corresponding classes `GToolbar`, `GButton`, `GText` and `GPicture` (Fig. 1). For example, `Toolbar` element can contain several control elements; `Picture` element has attribute defining filename of an image that will be shown and `Button` element can optionally contain images for normal and focused display.

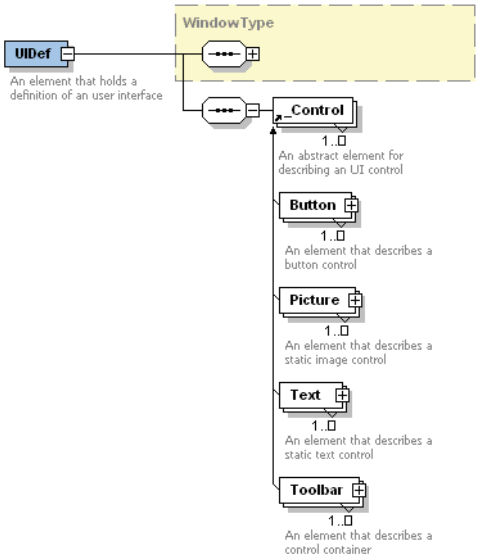

Fig. 4. Structure of an XML definition of a user interface

V. TOUCH INTERFACE FOR A TANK C4I SYSTEM

To illustrate features implemented in *GinisUI* framework, we present touch interface of a tank battlefield management system (*BMS*). *BMS* is in essence geographical information system extended with a command, control, communication, and virtual reality functions. *BMS* is equipped with a touchscreen for interfacing with a user.

BMS application can run in two modes: *combat* and *planning*; with two different views: *2D* and *3D*. Each mode with each view has different functions that user should be able to access through an interface.

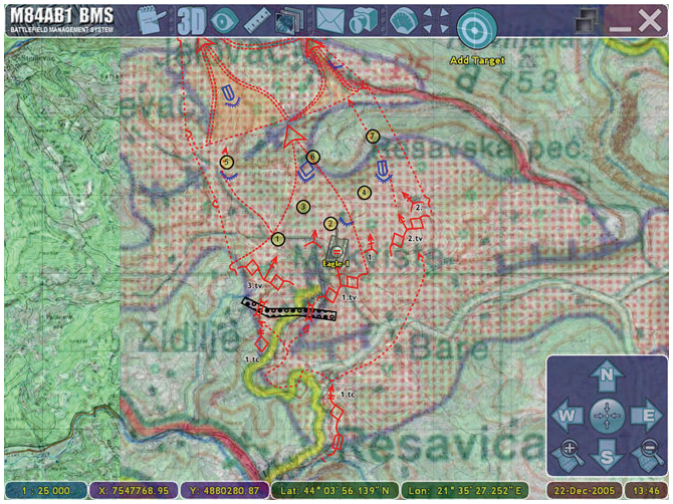

Fig. 5. BMS user interface in the 2D View

Fig. 6. BMS user interface in the 3D View

The user interface of *BMS* can be divided in always visible and visible on demand parts. Always visible parts are the main toolbar on top of the screen, status line at the bottom and navigation panel placed in the bottom-right corner. These interface elements are noticeable in Fig. 5 and 6. Depending on a mode or a view different controls can appear within these elements. The other, visible on demand part of the *BMS* user interface consists of various panels (Fig. 6), dropdown toolbars, message boxes, etc.

Fig. 6 shows *BMS* in the 3D view. *BMS* 3D view represents VR extension of GIS. It is used to display virtual terrain with some additional 3D objects (combat vehicles, trees, houses, buildings, etc.). This figure also shows 3D view specific compass panel and general system status panel.

User interface transition between 2D view (Fig. 5) which is drawn using GDI functions, and 3D view (Fig. 6) visualized using OpenGL, is seamless.

VI. Conclusion

We have discussed problems and identified requirements for a user interface of a GIS based C4I systems for combat vehicle embedment. Problems that we have faced were:

- Limited vehicle inner space – use of small touchscreen interfacing devices.
- Harsh environment – shaking of combat vehicles during use.
- Large amount of information to display – a geographic map or a virtual 3D terrain.

The solution that helps to reduce these problems relies on three key concepts:

- Transparency – enables integration of user interface with a background map or a virtual 3D terrain.
- Zooming – enables enlargement and additional information display of focused buttons.
- Automatic arrangement – combined with show/hide mechanism enables reconfiguration of a user interface to a current mode of an application.

Further, in the paper, we have introduced *GinisUI*, an application framework designed and implemented to enable rapid development of user interfaces based on these three concepts.

Described framework defines only basic building blocks needed for a user interface, while building a concrete user interface additionally requires explicit XML description what, and how these blocks create an interface. In addition to, and based on the framework, we specified XML language GUIDL to cope with these UI descriptions.

Finally, as an evaluation of our solution, we presented a case study concerning successful user interface realization of a C4I Battlefield Management System.

References

[1] Worboys, M., and Duckham, M., *GIS: A Computing Perspective, Second Edition*, CRC Press, Boca Raton, FL, 2004.
[2] Chang, K., *Introduction to Geographic Information Systems, Third Edition*, McGraw-Hill, New York, NY, 2005.
[3] Rauschert, I., Sharma, R., Fuhrmann, S., Brewer, I., and MacEachren, A., "Approaching a New Multimodal GIS-Interface", 2002, http://www.geovista.psu.edu.
[4] C4ISTAR, Wikipedia, http://en.wikipedia.org/wiki/C4I.
[5] What is C4I?, C4I.org, http://www.c4i.org/whatisc4i.html.
[6] Fayad, M., Johnson, R., and Schmidt, D., *Building Application Frameworks: Object-Oriented Foundation of Framework Design*, John Wiley & Sons, 1999.
[7] User Interface, Wikipedia, http://en.wikipedia.org.
[8] Bederson, B., and Meyer, J., "Implementing a Zooming User Interface: Experience Building Pad++", Software: Practice and Experience, Vol. 28, No. 10, pp. 1101-1135, 1998.
[9] Small, C. H., "Touchscreens Provide a Robust and Intuitive User Interface", TechOnLine, http://www.techonline.com.
[10] Milosavljević, A., Đorđević-Kajan, S., and Stoimenov, L., An Architecture for Open and Scalable WebGIS, 8th AGILE Conference on GIScience, pp. 629-634, Estoril, Portugal, 2005.
[11] Milosavljević, A., and Stoimenov, L., "Implementation Model of Ginis framework for Web-based GIS", ETRAN 2005, Conference Proceedings, pp. 35-38, Budva, Montenegro, 2005. (in Serbian)