

# Incorporating Tilt-based Interaction in Multimodal User Interfaces for Mobile Devices

Jani Mäntyjärvi<sup>1</sup>, Fabio Paternò<sup>2</sup>, and Carmen Santoro<sup>2</sup>

<sup>1</sup> VTT Technical Research Centre, Kaitoväylä 1,  
90571 Oulu, Finland

<sup>2</sup> ISTI-CNR, Via G. Moruzzi 1,  
56124 Pisa, Italy

{Jani.Mantjarvi, Fabio.Paterno, Carmen.Santoro}@Isti.Cnr.It

**Abstract.** Emerging ubiquitous environments raise the need to support multiple interaction modalities in diverse types of devices. Designing multimodal interfaces for ubiquitous environments using development tools creates challenges since target platforms support different resources and interfaces. Model-based approaches have been recognized as useful for managing the increasing complexity consequent to the many available interaction platforms. However, they have usually focused on graphical and/or vocal modalities. This paper presents a solution for enabling the development of tilt-based hand gesture and graphical modalities for mobile devices in a multimodal user interface development tool. The challenges related to developing gesture-based applications for various types of devices involving mobile devices are discussed in detail. The possible solution presented is based on a logical description language for hand-gesture user interfaces. Such language allows us to obtain a user interface implementation on the target mobile platform. The solution is illustrated with an example application that can be accessed from both the desktop and mobile device supporting tilt-based gesture interaction.

**Keywords:** Model-based design of user interfaces, gestural interfaces for mobile devices, tilt interfaces.

## 1 Introduction

Pervasive environments are characterized by seamless connectivity and various types of devices, through which a user can interact with applications. This creates challenges in implementing the means for interaction, and several interaction modalities e.g. graphical, vocal, gesture, touch, and tactile can potentially be used. Also, different devices have different operating platforms, and user interfaces must be described in different ways: for example some devices support browsers and the user interface can be implemented with XML-based mark-up languages e.g. X/HTML, VoiceXML, X+V etc., while in other devices access to system level software is required to enable the functionality for supporting specific modalities. Model-based approaches [16] [20] can be useful to address such increasing complexity by providing meaningful abstractions that allow designers to focus on the main

conceptual aspects without being distracted by a plethora of low-level implementation details.

While support for model-based development of multimodal interfaces, limited to graphical and vocal modalities, has already been investigated [1][17][19], so far no proposal to address gesture interaction with mobile devices through model-based design has been put forward. This paper presents an in-depth discussion of problems and challenges related to gesture interfaces for mobile devices and a solution for interface development able to target mobile devices supporting gestures.

The structure of the paper is as follows: after discussing related and background work, we highlight the challenges related to designing gesture interfaces for mobile devices from abstract interface definitions. We then present a solution for enabling hand gesture UI development for target mobile platforms with the Multimodal TERESA environment. The approach is illustrated with an example application. Lastly, discussions and conclusions are provided along with indications for future work.

## 2 Related Work

Obrenovic et al. [15] have investigated the use of conceptual models expressed in UML in order to then derive graphical, form-based interfaces for desktop or mobile devices or vocal devices. UML is a software engineering standard mainly developed for designing the internal software of application functionalities. Thus, it seems unsuitable to capture the specific characteristics of user interfaces and their software. The ICO formalism for user interfaces has shown to be suitable to model and specify multimodal interfaces mainly for analysis in safety-critical application [2], and it has limited automatic support for generation of multi-modal interfaces from such specifications.

One interesting effort to ease multimodal interface development is ICARE [3]: it provides a graphical environment for a component-based user interface exploiting various modalities and modules that allow various compositions of such modalities. In this paper we present a different approach: we show how we can derive multimodal interfaces starting with logical descriptions of tasks and user interfaces obtained through general, platform-independent notations. We still provide the possibility of combining the modalities in various ways, but at different granularity levels (inside a single interaction object and among several interaction objects). While some other work has been carried out to apply transformations to logical descriptions to derive multimodal interfaces [19], our work has been able to provide an authoring environment that is able to suggest solutions for identifying how to combine various modalities and allows designers to easily modify them in order to tailor the interface generation to specific needs. This result has been obtained by extending a previously existing authoring tool [14][17] that was limited to creating graphical and/or vocal interfaces.

In the research area of gesture-based interfaces, Rekimoto [18] carried out pioneering work on tilt-based interaction. Later on, Hinckley presented a study on how tilt actions and other sensors could be used in mobile devices [8]. More recently, other studies have been conducted from various viewpoints e.g., Murray et al. have

applied control theory to tilt interaction [7]. The project XWand has studied a concept of multi-modal interaction for pervasive environments [21]. In this approach a handheld device equipped with inertial sensors is utilized to control various functionalities embedded in a room. Camera-based technologies are also used in mapping hand gestures to interaction commands. In [13], a handheld device with inertial sensing is utilized to control devices and applications in public spaces. The same paper also presents an in-depth examination of approaches for user dependent and independent gesture recognition. Also, the mobile-device, entertainment and consumer electronics industries have presented product concepts utilizing hand gesture interaction [10, 11]. However, all these contributions on gesture interaction have not addressed model-based generation.

### **3 Background**

To provide some background on our approach the specific aspects of gestural interaction in mobile devices are identified and discussed. In addition, we introduce the framework for enabling model-based design of multi-modal user interfaces and situate the support for hand gesture interaction within such a framework.

#### **3.1 Hand Gesture Interaction through Inertial Sensing**

In this subsection we discuss hand gesture interaction and interface design, as well as the challenges they pose for model-based UI design.

Some primary uses of hand gestures in interaction tasks include triggering, selection, and navigation. The design of gesture interfaces in general is a challenging task. The results obtained from pen-gesture and hand-gesture interface design studies are quite similar [9, 10]: gestures must be easily remembered by a user and captured reliably by a recognition system to ensure positive user experience. There are many challenges arising from the nature of gestures: the number of gestures is infinite; gestures can be simple or more complex; lastly, gestures are personal, e.g. the same gesture can be performed with different speed, scale and orientation of the hand.

There are various types of gestures. From the viewpoint of recognition systems they can be divided into three categories [9, 13]:

- Measure and control – measured signals tilt, rotation or amplitude are directly mapped onto interaction events.
- Discrete gestures – more complex gestures e.g. symbols, are captured with machine learning methods, and the beginning and end of a gesture are indicated by pressing and releasing a button.
- Continuous gestures – more complex gestures are captured with machine learning methods, and recognition runs continuously.

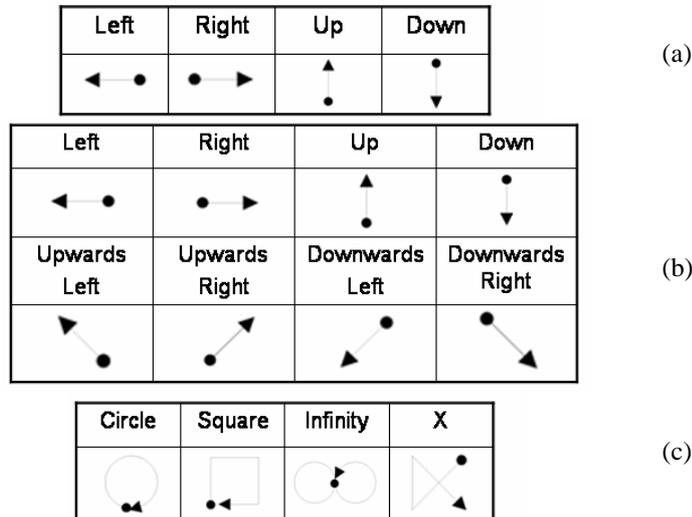
There are also various types of sensors available for various devices that may limit the amount and types of gestures supported by a target platform, e.g. a mobile device. These challenges affect the design of recognition methods and set requirements on how to utilize gestures in end-user devices.

The gestures recognised on a device can be specified or a user can personalize them [9, 11]. In the case of fixed gestures, there usually is a set of trained models with a pre-defined gesture vocabulary in a gesture manager - a gesture recognition service located in a device - producing a set of pre-established consequences. On the other hand, in the case of personalized gestures, the manager may contain a predefined fixed set of trained models. However, the user also has the possibility to edit (add, remove, re-name, etc.) the existing set and, moreover, may train new gestures and link them to specific device actions.

The important role of gestures must be borne in mind in the design of multimodal interfaces. Gestures can potentially be used in navigation and in providing control commands. So a user interface supporting gesture interaction must combine at least the graphical and gesture modalities. In addition, vocal or tactile modalities can potentially be included in mobile devices. For example, let us consider the basic interaction elements: prompt, input and feedback. In interaction with a mobile device supporting gestures, several modalities should be used in a complete interaction: graphical for the prompt, gestures for providing input, and vocal, tactile and/or graphical for feedback.

In summary, the main challenges related to designing gesture-enabled interfaces for mobile devices using model-based approach are that gesture descriptions must tolerate uncertainty from gesture recognition. However, this applies mainly for more complex gestures while simple gestures, e.g. tilt gestures are provided usually with 100% accuracy. In addition to fixed naming of gestures, there must be support for defining customized gestures. Since inputting (control, activate, select and navigate) is the primary purpose of gesture interaction and it is impossible to prompt and give feedback through gestures, various types of interaction elements must be carefully combined with other modalities in the user interface to ensure a functional final UI for a target device. Moreover, in the model-based approach the description language defining the gesture UI (gesture vocabulary) must be general and flexible for allowing both stationary and customisable gesture sets.

In this paper, we present a solution able to support multi-device applications involving mobile devices with gesture control. Our development environment supports fixed simple and complex gesture sets. Gesture sets supported by development environment are presented in Figure 1.



**Fig. 1.** Illustration of the gestures sets supported. *a)* Tilt set, *b)* Arrow set, *c)* Symbol set. Dot means the start of the gesture while black triangle (end of arrow) refers to end of gesture.

Particular in gesture interfaces, tilt gesture means that a user may tilt a device, for example to left/right/backward/forward, to initiate some UI action. Tilt events are recognized by a gesture recognition software using signals from accelerometers. Arrow-, symbol- or other free form gestures mean that a device is waved to various directions and shapes of gestures imitate given shapes, arrows or symbols, etc. These types of gestures are more complex from gesture recognition viewpoint and they require machine learning-based recognition algorithms. Due to gesture sensing hardware availability only tilt gestures are supported in our target platform.

### 3.2 Model-Based Approaches

In the research community in model-based design of user interfaces there is a consensus on what the useful logical descriptions are [5, 16, 20]:

- The task and object level, which reflects the user view of the interactive system in terms of logical activities and objects that should be manipulated to accomplish them;
- The abstract user interface, which provides a platform independent description of the user interface;
- The concrete interface, which provides a platform dependent but implementation language independent description of the user interface;
- The final implementation, in an implementation language for user interfaces.

Thus, for example we can consider the task select a work of art, this implies the need for a selection object at the abstract level, which indicates nothing

regarding the modality in which the selection will be performed (it could be through a gesture or a vocal command or a graphical interaction). When we move to the concrete description then we have to assume a specific platform, for example the graphical desktop, and indicate a specific platform-dependent interaction technique to support the interaction in question (for example, selection could be through a radio-button or a list or a drop-down menu), but nothing is indicated in terms of a specific implementation language. By platform we mean a set of interaction resources that share similar capabilities (for example the graphical desktop, the vocal one, the cellphone, the graphical and vocal desktop). When we choose an implementation language we are ready to make the last transformation from the concrete description into the syntax of a specific user interface implementation language. We prefer to use the term platform independent for the abstract level rather than the term modality independent because a multimodal system is a systems that processes two or more combined user input modes (such as speech, pen, touch, gestures, ...) but we can have platforms that use the same input modes but with rather different available interaction resources (for example in terms of screen size) that heavily affect the possible task support and thus requiring different strategies for adapting the user interfaces to their features in order to obtain usable results.

The advantage of this type of approach is that it allows designers to focus on logical aspects and takes into account the user view right from the earliest stages of the design process. In the case of interfaces that can be accessed through different types of devices the approach has additional advantages. First of all, the task and the abstract level can be described through the same language for whatever platform we aim to address. Then, in our approach we have a concrete interface language for each target platform. Thus, a given platform identifies the type of interaction environment available for the user, and this clearly depends on the modalities supported by the platform itself. Actually, in our approach the concrete level is a refinement of the abstract interface depending on the associated platform. This means that all the concrete interface languages share the same structure and add concrete platform-dependent details on the possible attributes for implementing the logical interaction objects and the ways to compose them. All languages in our approach, for any abstraction level, are defined in terms of XML in order to make them more easily manageable and allow their export/import in different tools.

Another advantage of this approach is that maintaining links among the elements in the various abstraction levels allows the possibility of linking semantic information (such as the activity that users intend to do) and implementation, which can be exploited in many ways. A further advantage is that designers of multi-device interfaces do not have to learn the many details of the many possible implementation languages because the environment allows them to have full control over the design through the logical descriptions and leave the implementation to an automatic transformation from the concrete level to the target implementation language. In addition, if a new implementation language needs to be addressed, the entire structure of the environment does not change, but only the transformation from the associated concrete level to the new language has to be

added. This is not difficult because the concrete level is already a detailed description of how the interface should be structured.

This means that the main challenge we had to address for supporting gestural interactions with mobile device through a model-based approach was the design of a concrete interface language for this platform. This language on the one hand should be the refinement of the general abstract language and on the other hand should contain all the relevant information necessary to generate then an implementation for this target platform in any language able to support it. In particular, as first implementation environment we have used the Microsoft Visual Studio 2005 with C# language enriched with the libraries for controlling the 2D accelerometer from Ecertech (<http://ecertech.com>).

## 4 A Logical Language for Gesture Interface

The XML concrete gesture interface language is a refinement of the abstract TERESA language [3]. It provides support for various types of gestures, which can be associated with the interaction types supported. It covers some gesture sets, e.g. basic tilt-set, free form arrow and symbol sets. Defining more complex custom sets also enables support for user-defined gestures. The concrete gesture interface language defines an interface using default settings and a number of presentations, each of which consists of definitions of the relevant interactors (interaction or only output) as well as the composition operators.

In the following the CUI declarations are presented. For all presentations, the gesture settings define which gesture sets are used:

```
<!ELEMENT default_settings (graphic_settings, gesture_settings, operators_
settings, interactors_settings)>
<!ELEMENT gesture_settings (change_focus*, gesture_sets*)>
<!ELEMENT gesture_sets (tilt, arrow, symbol, custom*)>
```

The important issue in interacting with a multi-device application using gestures is simply to navigate between elements on a UI. For gesture UI it is crucially important to highlight the current focus of interaction. Changing the focus is defined as default settings within the gesture modality. In our example changing the focus is done by using basic tilt commands:

```
<!ATTLIST change_focus
    next (%tilt_act;) #REQUIRED
    previous (%tilt_act;) #REQUIRED>
```

Showing the focus graphically on a target platform can be done by highlighting graphical elements, such as, font-change, box, colour, etc.

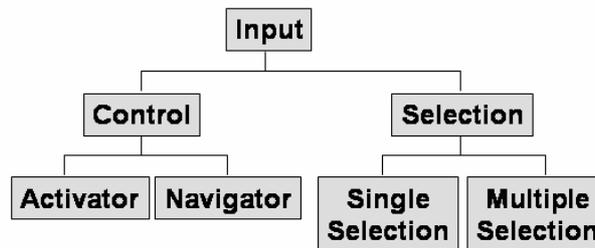
In the tool, the gesture actions can be associated with interactors. The gesture actions are defined by default with the following sets:

```

<!ENTITY % tilt_act "null | tilt_left | tilt_right | tilt_forward | tilt_backward">
<!ENTITY % arrow_act "null | arrow_left | arrow_right | arrow_up | arrow_down |
    arrow_downwd_left | arrow_downwd_right | arrow_upwd_left |
    arrow_upwd_right ">
<!ENTITY % symbol_act "null | circle | square | infinity | x">

```

In multimodal UI, gestures can be potentially used only for input. In particular, they are suitable for control and selection. The relevant input-interactors to be used with gestures are indicated in Figure 2. The control interactor includes navigator and activator elements, while the selection interactor includes single and multiple selection elements. Such types of interactions also need to operate in parallel with graphical elements such as buttons or various types of lists. Graphically, interactors can be implemented using many objects, such as links, various types of buttons and menus and boxes.



**Fig. 2.** Gesture-supported interaction interactors

Input interactors for gestures are defined as extensions to existing graphical ones; while defining graphical description definitions state gestures for interactors from the given sets.

```

<!ELEMENT activator (button, gesture*)>,
<!ELEMENT navigator (button, gesture*)>,
<!ELEMENT single ((radio_button | list_box | drop_down_list), gesture*)>,
<!ELEMENT multiple ((check_box | list_box), gesture*)>,

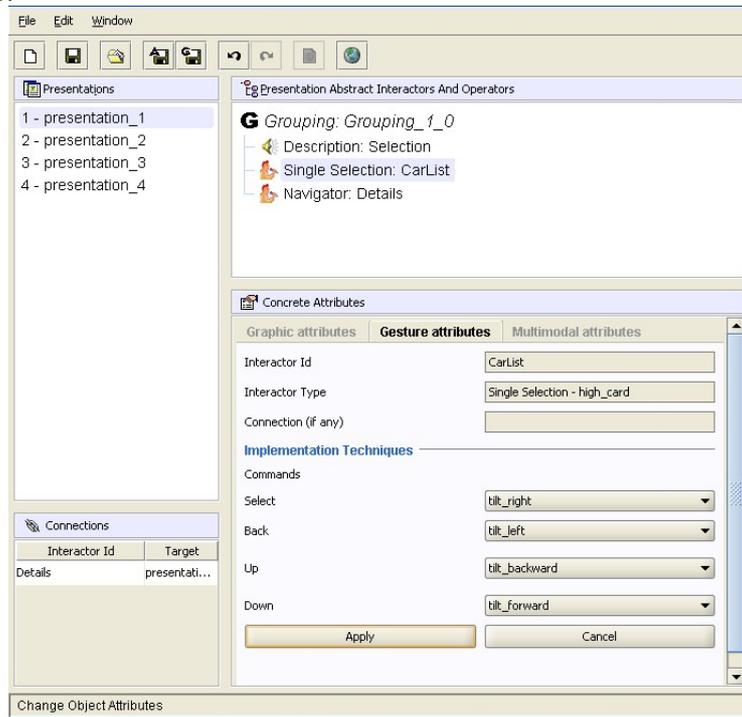
```

where gesture element is defined with given gesture set(s) with possible gesture entities.

## 5 Gesture Interface Development

The Multimodal TERESA tool [14] already supports development of graphical and vocal modalities and enables interfaces implemented in XHTML, XHTML Mobile Profile, VoiceXML and X+V. In order to extend the interoperability of the tool for various types of platforms we have chosen the Pocket PC OS for our target platform and the C# implementation language for the target platform. The Microsoft environment provides robust support for PDAs. Our authoring tool has been extended

in order to define gesture actions for interacting with a mobile device supporting gesture interaction. A screenshot of the user interface editor of the tool when defining the application user interface supporting graphical+gesture modalities is shown in Figure 3.



**Fig. 3.** Screen view of a UI development tool when designing graphical+gesture interface for a mobile device for our example

The structure of the control panel of the tool is described in [14]. For supporting gesture modality there is a new Gesture Attributes- area in the Concrete Properties subpanel. Moreover, the Multimodal Properties panel supports the specification of the relevant CARE (Complementary, Assignment, Equivalence and Redundancy) properties [6], which can be used to set how the supported modalities are utilized by a user to reach a goal. In addition, how to compose the modalities can be better analyzed by considering the different phases of an interaction (prompt, input, feedback). In the case of gesture-based interactions it is reasonable to output information only graphically. In this case outputting information refers to providing prompt and feedback, so the property for outputting data is graphical assignment by default.

A user should be able to interact using either the graphical or gestural modalities. Thus, the property for inputting information is equivalence, while graphical assignment can be another option. In the case of composition operators (grouping, hierarchy, ordering and relation), the only relevant property is graphical assignment

for the same reason as in the output-only interactor (the gestural modality cannot be used to output information).

Graphic attributes related to fonts, background, and graphical objects such as text and image descriptions are set on the tool for CUI. In the final UI generation phase, they are converted to corresponding definitions on target language by using corresponding classes, methods and properties.

On a target implementation, the user interface is defined in the following way. Firstly, interaction objects are associated to presentations and graphics and gesture attributes are set in the tool. Then, during the final UI generation the selected interactors are used to define corresponding controls by using the corresponding classes on the target language. Activators and navigators are implemented as buttons while single and multiple selectors are implemented using radio-button, check-box, list-box or drop-down-list. Regarding composition operators, grouping of UI object on a final UI is supported by using boxes including the relevant elements.

The gesture events supported are defined in a tool. In the code of the final UI, the commands defined are set as alternative choices for activating defined controllers either graphically or gesturally (when modalities are equivalent). For a target platform, equivalent gesture controls are implemented in the following manner. Elements of the generated graphical UI layout with event handlers are defined. The gesture events are associated with state transitions between interactor elements. When generating this part of code the input is taken from connections between presentations from the CUI. In order to incorporate interoperability a designer should make sure that naming of gesture actions in the tool matches to gesture events produced by GestureManager on a target platform. On a target platform, showing and moving focus graphically according to tilt-gestures is carried out by adding focus-related methods and properties to the state transitions of the user interface. The focus of the active state is shown through graphical highlighting.

The type of application that we have considered is form-based, meaning that the interactive part of the UI on various platforms (desktop and mobile device) consists mainly of interactive forms.

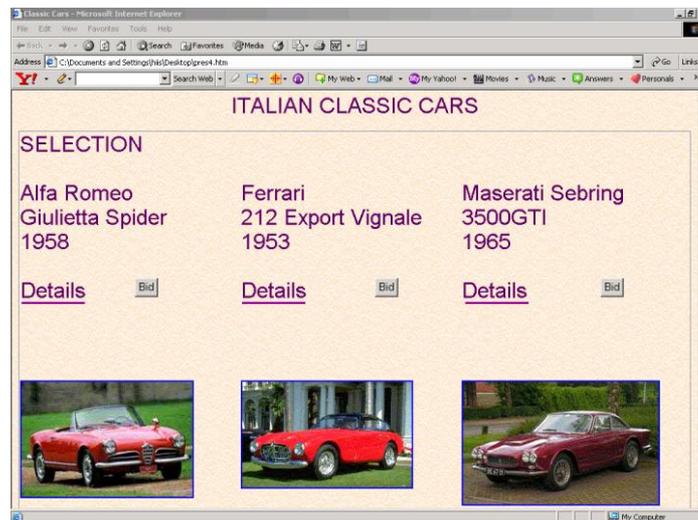
In our approach for producing UI for the selected target platform in Microsoft C# (supported by Visual Studio 2005), we assume that there is a project for containing libraries or an interface to a software producing gesture events (GestureManager - recognition software). In Microsoft Visual Studio 2005 a user interface is (mainly) defined through the Windows Forms, which, in C#, are defined as `<name>.cs` and `<name>.designer.cs` -files. The first file type contains the source for defining functionality of the form - together with other sources files while the latter file type defines the graphical layout of a window (Form). The `<name>.designer.cs` -files can be created traditionally by typing or automatically when designing layout by drag and drop in the Designer tool of the Interface Development Environment. The goal in our transformation is to generate content to the `<name>.designer.cs` -files and to the `<name>.cs` -files for defining graphical and gesture actions.

As discussed above, due to gesture sensing hardware availability only tilt gestures are supported in our target platform. In our tilt interaction model, vertical tilt actions (`tilt_forward` - next, `tilt_backward` - previous) cause a change of focus among interactors, i.e. vertical tilt allows navigation between interactors. Horizontal tilts, right or left, respectively define the actions selection and step back. After an interactor

(e.g. a drop-down menu) is activated with `tilt_right`, vertical tilt can be used to navigate amongst elements on the menu. Once again, `tilt_right` defines activation of the selected item and `tilt_left` exits the current selection mode (e.g. returning to the upper level). The complete specification of the user interface through the tool defines the interaction with the application on a target language. After the final UI generation, the parts of code generated along with image and text resources are included into a project in the IDE, which must be then compiled and installed to a device. On mobile devices this can be done through over-the-air (OTA) transfer.

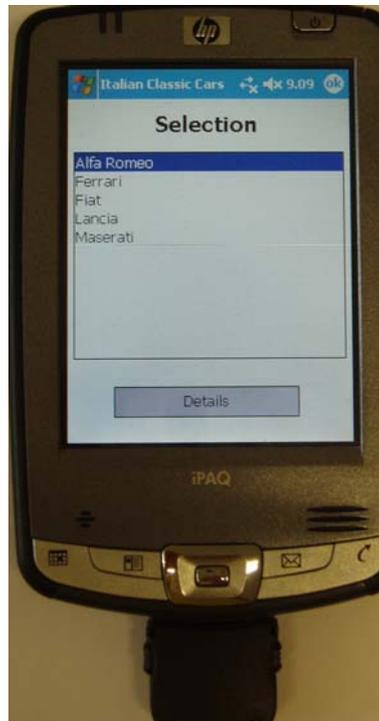
## 6 Example

This section illustrates the development of gesture enhanced mobile application with a scenario and an example application.



**Fig. 4** The user interface for the desktop application

Our usage scenario considers Giovanni, who searches information on classic Italian cars. He accesses a desktop application, which provides him with information on such cars. The application allows him also to place offers on cars. Later, Giovanni decides to go and have a look at the car selection. The car sales point also supports mobile access for describing cars and for making offers. When Giovanni enters the sales point place, his mobile receives notification on the possibility of installing an Italian Classic Cars application.



**Fig. 5** Application on a mobile device. *Upper image:* Main selection view; *Bottom-left image:* Overview of selected car; *Bottom-right image:* Description of the selected car.

He accepts the installation and starts to use the application. Our scenario is one possible usage scenario regarding mobile and pervasive services and applications. Services have various implementations available to support constantly widening selection of mobile and pervasive target platforms. For this reason, it is necessary to be able to provide development environment from which various versions of

interfaces and applications can be fluently generated. The desktop UI generated by using the tool supports only graphical modality while the mobile device version also generated by the tool supports graphical and gesture modalities, and appearance and functionality of the application are adapted according to properties of a mobile terminal, as set in the tool.

User interfaces for desktop and for mobile device developed using our authoring tool are shown in Figures 4 and 5, correspondingly. The accelerometer able to sense gestures is the small box at the bottom of the PDA (Figure 5. Upper image).

The desktop application is implemented through a Web site showing overview of the selection with links to details and trading. The application on a mobile device is implemented as executable application. It utilizes tab-type-UI, since it provides users with an idea on several views, which can be changed by tilting horizontally. The main selection view on mobile device shows names of cars implemented as a list box, which can be browsed up and down with vertical tilting. Details link and Bid-button are changed into tabs on a mobile device. By tilting right the overview presentation of the selected car (Figure 5. Bottom-left image) can be accessed as illustrated in our tilt interaction model. By tilting again right another presentation, description, can be accessed (Figure 5. Bottom-right image), etc. By tilting left previous presentations are shown until main view is accessed. The interactors can be used in parallel with graphical modality. Overview-tab on a mobile device shows the basic description while the description-tab shows extended description. On the desktop version, the overview is shortly shown on the main page and details behind details-link. On mobile application objects in each presentation are grouped together to a compact presentation suitable for small display.

## 7 Discussion

From the gesture interaction viewpoint, the solution proposed for developing gesture user interfaces using a model-based approach is based on experiences with hand gestures interfaces developed by using miniaturized inertial sensors (accelerometers, magnetometers, gyroscopes), which can be attached to small hand-held devices. We have a device using an Ecertech 2D accelerometer device (it can be seen in the bottom part of the upper photo in Figure 5). The solution is demonstrated with a simple tilt gesture set and the target application can be extended to support to more complex shape-based or customised gestures enabled by 3D inertial sensing HW. The solution can also be generalized to be applicable to other types of hand gesture interfaces, in which gesture events are sensed by other means, e.g. cameras etc.

In order to increase the interoperability of the TERESA development environment for supporting pervasive embedded devices the chosen target implementation language in our concrete example is C# running on a Pocket PC operating system, instead of a Web-based application. Although generating the executable application and downloading it to a target device is more complex than dealing with document descriptions supported by browsers, we see that in the near

future pervasive computing environments will involve various types of embedded wireless devices, which necessary do not support browsers.

## 8 Conclusions

We have discussed issues related to design of gesture interfaces. A solution based on using a model-based approach for developing multimodal interfaces supporting gesture interfaces is presented. It is obtained extending the existing TERESA tool. The solution is illustrated with a concrete example in which both desktop and mobile device applications are generated. The mobile application supports gestural interaction. The desktop platform provides graphical interface for prompt, feedback and input while the interface in the multi-modal mobile device uses the limited graphical resources for prompt and feedback and allows gesture interface for user input. Actually, the mobile device uses both modalities in equivalent manner by providing a user with additional options for performing commands through gestures. The approach is illustrated with real application example using a quite simple set of gestures. We would like to point out that the approach may support any set of gestures supported by the target platform.

The approach also shows the interoperability of the framework, since the target platform is a PDA with Pocket PC operating system. The form-based application is compiled after applying code generated from the tool and then the application must be downloaded to the mobile terminal.

Future work will be dedicated to integrating this functionality to migratory interface environments and to extend it to support multi-modal gesture interaction with mobile devices. Also, we plan to develop applications with a more complex and free form sets of gestures supported by the target platform.

**Acknowledgments.** Our thanks to ERCIM for support.

## References

1. Abrams, M., Phanouriou, C., Batongbacal, A., Williams, S., Shuster, J., 1994. UIML: An Appliance-Independent XML User Interface Language, Proceedings of the 8th WWW conference. M., Chang, C.-C.K., Gravano, L., Paepcke, A.: The Stanford Digital Library Metadata Architecture. *Int. J. Digit. Libr.* 1 (1997) 108–121.
2. Bastide, R., Navarre, D., Palanque, P., Schyn A. & Dragicevic, P., A Model-Based Approach for Real-Time Embedded Multimodal Systems in Military Aircrafts, Proceedings ICMI 2004, pages 243-250, ACM Press.
3. Berti, S., Correani, F., Paternò, F., Santoro, C., The TERESA XML Language for the Description of Interactive Systems at Multiple Abstraction Levels, In Workshop Organized at Advanced Visual Interfaces 2004, pp. 103-110.

4. Bouchet, J., Nigay, L. Ganille T., ICARE software components for rapidly developing multimodal interfaces. Proceedings ICMI 2004, pages 251-258, ACM Press.
5. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J. A Unifying Reference Framework for Multi-Target User Interfaces. *Interacting with Computers*. Vol. 15, No. 3, June 2003, pp. 289-308.
6. Coutaz J., Nigay L., Salber D., Blandford A, May J., Young R., 1995. Four Easy Pieces for Assessing the Usability of Multimodal Interaction: the CARE properties. Proceedings INTERACT 1995, pp.115-120.
7. Eslambolchilar, P., Murray-Smith, R., Tilt-based Automatic Zooming and Scaling in Mobile Devices - a state-space implementation, *Mobile Human-Computer Interaction MobileHCI 2004: 6th International Symposium, Glasgow, UK, September 13-16, 2004*. Proceedings. Stephen Brewster, Mark Dunlop (Eds), LNCS 3160, Springer-Verlag, p120-131, 2004.
8. Hinckley, K., Pierce, J., Sinclair, M., Horvitz, E., Sensing Techniques for Mobile Interaction, ACM UIST 2000 Symposium on User Interface Software & Technology, CHI Letters 2 (2), pp. 91-100.
9. Kela, J., Korpipää, P., Mäntyjärvi, J., Kallio, S., Savino, G., Jozzo, L., Di Marca, S., Accelerometer-based gesture control for a design environment, In *Personal and Ubiquitous Computing*, Springer, DOI 10.1007/s00779-005-0033-8,(online), 2005.
10. Kim, S-H, Ok, J., Kang, H.J., Kim, M-C., Kim, M., An interaction and product design of gesture based TV remote control. In *CHI '04 extended abstracts*, Vienna, Austria, 2004, pp. 1548 – 1548.
11. Linjama J., Kaaresoja T., Novel, minimalist haptic gesture interaction for mobile devices, In *Proc.Nordic-CHI, 2004, Tampere, Finland*, pp. 457 – 458.
12. Long AC, Landay JA, Rowe LA. Implications for a gesture design tool. In: *Human Factors in Computing Systems (SIGCHI Proc.)*. ACM Press, 1999. 40-47.
13. Mäntyjärvi, J., Kallio, S., Korpipää, P., Kela, J., Plomp, J., Gesture interaction for small handheld devices to support multimedia applications, In *Journal of Mobile Multimedia*, Rinton Press, Vol.1(2), pp. 92 – 112, 2005.
14. Mori, G. Paternò, F. Santoro C., Design and Development of Multi-Device User Interfaces through Multiple Logical Descriptions, *IEEE Transactions on Software Engineering*, August 2004, Vol.30, N.8, pp.507-520, IEEE Press.
15. Obrenovic, Z., Starcevic D., Selic B., A Model-Driven Approach to Content Repurposing, *IEEE Multimedia*, January March 2004, pp.62-71.
16. Paternò, F., “Model-Based Design and Evaluation of Interactive Application”. Springer Verlag, ISBN 1-85233-155-0, 1999.
17. Paternò, F., Giammarino, F., Authoring Interfaces with Combined Use of Graphics and Voice for both Stationary and Mobile Devices, *Proceedings AVI'06*, ACM Press, May 2006.
18. Rekimoto J., Tilting operations for small screen interfaces, *ACM UIST 1996, 1996.*, pp. 167-168.
19. Stanciulescu A., Limbourg Q., Vanderdonckt J., Michotte B., Montero F., A Transformational Approach for Multimodal Web User Interfaces based on USIXML. *Proceedings ICMI 2005*, pages 259-266, ACM Press.
20. Szekely, P., 1996. Retrospective and Challenges for Model-Based Interface Development. *2nd International Workshop on Computer-Aided Design of User Interfaces*, Namur, Namur University Press.
21. Wilson A, Shafer S. Between u and i: XWand: UI for intelligent spaces. *Proceedings of the conference on Human factors in computing systems, CHI 2003, April 2003*. pp 545-552.