A Design Method for Next Generation User Interfaces Inspired by the Mixed Reality Continuum

Jörg Stöcklein[1], Christian Geiger[2], Volker Paelke[3], and Patrick Pogscheba[2]

[1] University of Paderborn, Germany
ozone@uni-paderborn.de
[2] Duesseldorf University of Applied Sciences, Germany
{geiger,Patrick.Pogscheba}@fh-duesseldorf.de
[3] Leibniz University of Hannover, Germany
Volker.Paelke@ikg.uni-hannover.de

Abstract. In this paper we present a new approach to the systematic user centric development of next generation user interfaces (NGUI). Central elements of the approach are a conceptual model that extends the well established model view controller paradigm with an environment component, an iterative development methodology that guides development along the mixed reality continuum and tools to support the implementation. The approach is demonstrate with a concrete example of NGUI development.

Keywords: Mixed Reality, Next Generation User Interface Development.

1 Introduction

Next generation user interfaces (NGUI) diverge from the well known WIMP paradigm (window, icon, menu, pointer) and employ novel techniques like virtual, augmented and mixed reality interaction or tangible, embodied and multi-modal interfaces [1]. Achievements in recent years allow researchers to shift their focus from technical questions like IO devices, tracking and rendering to the design of such interfaces. However, they are faced with the challenge that NGUIs are less representational (no icons representing objects) and focus on reality-based interaction styles which leverage the user's built-in abilities by exploiting pre-existing skills and expectations from real-world experiences rather than computer trained skills. The proposed intuitiveness for end users has fundamental consequences for the designer. While design in other disciplines can draw on existing expertise to enable a streamlined, effective design processes without much experimentation, the lack of such expertise and the interdisciplinary nature of NGUI design requires a process based on iterative refinement and evaluation. The lack of a formal specification rules out the use of formal verification techniques and the limited design expertise limits the applicability of techniques like expert reviews for NGUI applications, leaving experimental evaluation through tests with potential end-users as the most promising option. However, a simple implement and test approach is not viable because the implementation of working prototypes is expensive and time consuming, limiting the number of concepts and designs that can be possibly explored. In order to evaluate the concepts

R. Shumaker (Ed.): Virtual and Mixed Reality, LNCS 5622, pp. 244–253, 2009.

under controlled conditions it is therefore necessary to remove technology problems from conceptual tests. To address these problems we developed a design process and corresponding support for NGUI design. The process is aimed at the fast development and evaluation of iteratively refined prototypes along a mixed reality continuum. This paper describes the principal idea of our design approach in section 3, and presents a project which was realized using our approach in section 4. We start with a short review of related work.

2 Related Work

A design approach for NGUI should be defined along three essential components. A conceptual model that allows to describe design entities for all stakeholders. A design process that provides a structured course of action and a technical framework that supports the efficient development of NGUI applications. The most prominent model for graphical user interfaces is the MVC paradigm originally defined for the Small-Talk80 programming language [2]. A model separates the data from its representation (view) and control elements (controller) provide for user interaction. Ishii extended this model to describe tangible user interfaces [3]. He adds a tangible / real representation to the digital view that is basically used as direct control mechanism. Real objects act as physical representations of interface elements and allow to use real and virtual elements in tangible user interfaces. Milgram's mixed reality continuum defines arbitrary combinations of real and virtual elements in more detail and ranges from virtual reality, augmented virtually, augmented reality to real reality [4]. Ourapproach uses this continuum as an additional dimension for the MVC model.

One of the earliest approaches towards a NGUI authoring framework was DART, an extension of Adobe Director by a number of AR features [5]. DART has been used in large projects and allows content experts to build AR worlds using Director's theatre metaphor as its conceptual model. Broll presented a visual authoring framework for 3D interaction techniques at 3DUI 2008 [6]. Their concept of "interactive bits" is a component-based approach to visually specify interactions techniques, object behavior or whole prototypes. The framework combines synchronous control and data flow with asynchronous event and network distribution and provides a XML-based description of the objects, components and the data/control flow. NIMMIT is a graphical notation for multimodal VR interaction techniques that is based on a state-chart model. Task chains between states define a linear control flow and high level variables (labels) allow data flow between tasks. Hierarchical task structures are supported and a tool allows to store the description in an XML file that can be loaded into a VE framework. A model-based design approach is nowadays mainly used for multi-device WIMP user interfaces but the declarative and visual description offers also benefits for NGUIs. Cuppens et al. presented a model-based framework for VR environments. They suggested to start with a task-related model that incrementally evolves towards the final user interface in an iterative design process [7]. Envir3D is a modeling tool, which enables users to visually specify 3D content, while preserving an underlying user interface model that can be used for evaluation and re-engineering. The model is used to generate VRML code that represents the 3D user interface. A number of relevant approaches, including model-based design, visual authoring and

HCI principles and guidelines for mixed reality user interface design was presented at the MRUI07 workshop at VR 2007 [8]. Due to space limitations we only give a reference to the proceedings that is one of the most relevant sources for our project. The main contribution of this paper is the combination of a extended 3D authoring framework with an iterative design approach that is based on a suitable conceptual model.

3 Design of Next Generation User Interfaces

While next generation user interfaces are often characterized by the employed technologies the ultimate aim is to create better interfaces for the user. It is therefore essential that a user centered design process is employed. At the same time the experimental nature of some base technologies must be adequately handled while the integration into larger systems requires the use of a systematic, controllable and manageable software engineering process. To address the need for experimentation within a structured development approach we structure our systems into loosely coupled components, using a model that extends the model-view-controller pattern with an additional component that captures the influence of the real world.

3.1 The MVCE Model

The model-view-controller pattern (MVC) structures user interfaces into three components and is popular in the development of user interfaces. The key benefit of this decomposition is that the visual and interaction aspects of a user interfaces can beisolated from the underlying application. The model (M) represents the application data and encapsulates the functionality of the application. The view (V) encapsulates the visual elements of the user interface, e.g. button widgets, text or visualizations. The controller (C) handles the interaction details, e.g. mouse events or text input and communicates necessary actions to the model. The MVC pattern enables modulardesigns in which changes in one component aren't coupled to other aspects. It alsoallows to provide multiple views and controllers for the same application/model. This is a desirable property, especially for multimodal interfaces that rely on specialized hardware that may not be available in all situations.

Figure 1a illustrates the communication between the three elements: When information in the model (e.g. application data) changes, the model notifies the view with a change_notification event. If an update of the presentation in the interface is required (e.g. this data is displayed) the view queries the model to retrieve the required

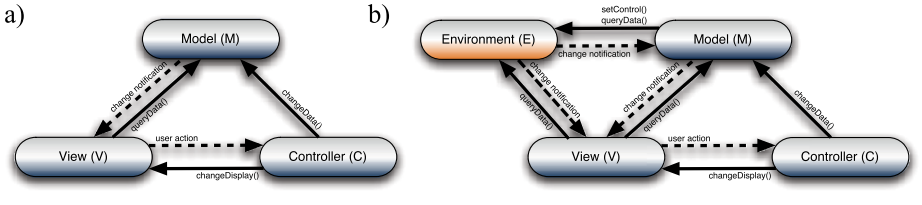

Fig. 1. a) The MVC model, b) The MVCE model

information and updates the information presentation accordingly. If the user interacts with the view (e.g. by clicking elements with the mouse) the controller is notified by user_action events. Depending on the semantics associated with the interaction event (the interaction technique encapsulated in the controller) the controller can change both data in the model (e.g. if the user inputs new values) or the presentation in the view (e.g. if the user changes the presentation).

One central feature of mixed-reality user interfaces is the integration with a real environment. The application requires information about objects and spaces, whose geometry and behavior is not under the control of the designer but must be acquired from the real environment. Real objects can be subject to real-world manipulation (e.g. in a maintenance task) or external forces. Therefore, it must be possible to track state changes in the environment. In practice the "real world" model of a mixed reality application often consists of a combination of static information (e.g. geometry of the environment that is assumed to be fixed) and dynamic information (e.g. position and orientation information for the user and central objects) that is acquired by sensors at runtime. While sensor information could be handled as controller events in the MVC model this can lead to complex and obscure models. We have therefore introduced an additional environment (E) component that captures the "real world" model of the application (see Figure 1b). A perfect real world model would contain all information about the real environment at the time of query. In practice both the amount of information required by the application and the amount actually accessible through sensors is limited.

The environment (E) is used similar to the model (M), with the main difference that the software has only a limited influence on the environment through dedicated controls, while the model is in theory completely controllable by software. Both the model (M) and the view (V) can query the environment (E). This allows to capture spatial association (e.g. the common augmented reality scenario in which a view object is fixed to a physical location or object) as well as control relations (e.g. objects in the real environment influenced by the application). Sensors in the environment can issue change_notification events to inform other components about detected changes, which can be used to implement tangible interaction techniques in which physical interaction with physical objects is interpreted as an action on software objects.

Using the MVCE structure, components can be refined independently. The current development state of a prototype can be characterized by the amount of complexity/realism for each component, as visualized in the component refinement diagram in figure 2, where the center indicates the most abstract representation and movement along the MVCE axes represents in creasing refinement/realism of the corresponding components. Each axis indicates the independent refinement state of the corresponding component. One key benefit is the possibility to develop a user interface "along the mixed-reality" continuum, starting with a virtual environment in which the environment (E) is represented by a model. Testing mixed reality interfaces in a virtual environment allows to focus on

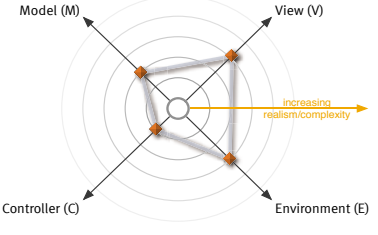

Fig. 2. Component refinement state diagram

interaction mechanisms and can provide controlled conditions for tests, while avoiding limitations of mixed reality technologies (e.g. tracking systems) that are often present in early development stages. Refinement of the E component ranges from more refined models to real-time data acquisition in the real environment. Arbitrary combinations of components are possible, e.g. it is sometimes useful to combine refined MVC components with a simple E model, for tests or demonstrations in later development stages.

The design approach needs adequate tool support that reflects the iterative nature of the process. Tools that support the efficient exchange of modules during iterative prototyping should fulfill a number of requirements: (1) a component-or building block-oriented system architecture, (2) a large component repository providing a variety of functions for NGUI design and (3) a visual authoring framework supporting quick prototyping. We provide such tool support with our HYUI system. More details about the underlying framework were presented in [9].

4 Example

We applied the iterative design approach as described in section 3 in an ongoing project where next generation user interfaces were developed along the mixed reality continuum. We used an indoor airship and are currently developing a training system that uses the complete MR continuum for different design variants. The idea is to train users of a radio-controlled indoor airship and develop new interaction techniques and training scenarios.

Technically the indoor airship is controlled by three propeller, as shown in figure 3a. The left and right propeller (figure 3b) can be rotated around the pitch axis (z). Both are connected to each other, thus the rotation speed and the pitch angle are the same. With these two propeller the zeppelin can navigate back and forth as well as up and down and diagonal. The third propeller is mounted at the back of the zeppelin (figure 3c) and is used for rotation around the yaw axis (y). Therefore the zeppelin has 3 degrees of freedom, translation on the x-and y-axis and rotation around the y-axis. The zeppelin can‘t directly move along the z-axis or roll and pitch.

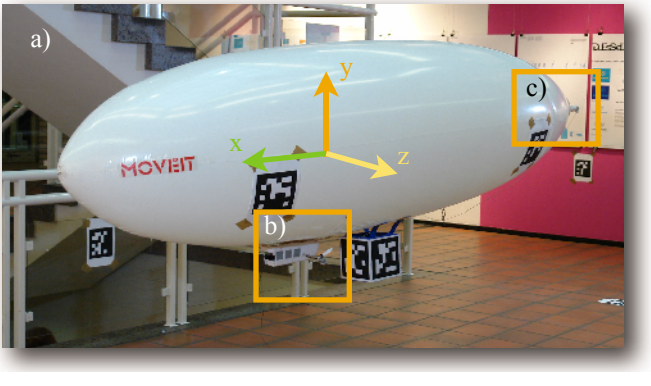

Fig. 3. The real Zeppelin

For controlling the airship's propeller a radio-powered remote control is used. Steering the zeppelin with it is tricky and for simple movement users have to practice many hours. Difficult maneuvers like flying to a specific point with a certain orientation need even more practice. The development of such a training application was used to demonstrate our design process.

4.1 Prototype 1: Simple Virtual Reality

We started the first prototype with a small 3D scenario using a simple script that modifies the transformation matrix of the model. With the keyboard the user can move the airship around in a virtual environment. We specify the four different parts of our MCVE paradigm as followed: The model is the transformation matrix (position and orientation) which moves and rotates our virtual zeppelin. The environment consists of a virtual floor only and constrains the flight level. The view is the simplified 3D model of the airship, and the controller is the direct manipulation of the transformation matrix of our model. This is done using the computer keyboard.

Users can easily practice maneuvers and learn how to handle the zeppelin. A disadvantage is the use of a keyboard as input device, because it is very different to the remote control used for the real zeppelin. Figure 4 shows our simple application and the assignment in our component refinement state diagram. The model, the environment, the view and the controller are very basic, therefore all the points in the component refinement state diagram are near the center.

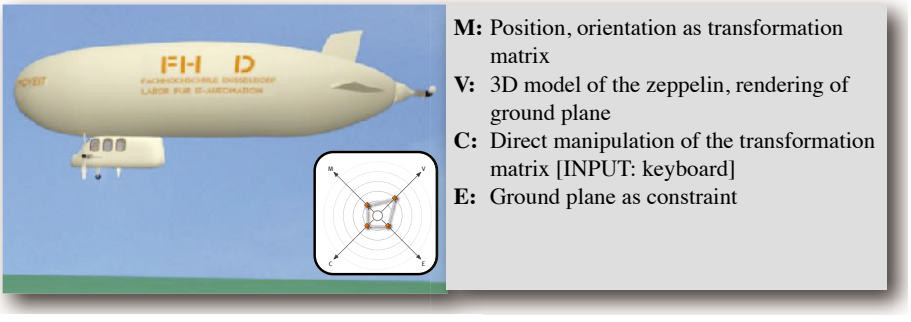

Fig. 4. Simple virtual reality application

4.2 Prototype 2: Game Physics

In the second prototype we tweaked the application towards more realism and complexity. As model we uses the Havoc game physic engine [10], which calculates the position and orientation of the airship (figure 5). We changes the environment to a static 3D CAD model of the real environment. The 3D model of the zeppelin is improved and the 3D model of building (the environment) is rendered. Forces of the propeller are visualized as vectors. For the controller we manipulate the 3 DOF of the zeppelin directly (rotor orientation, rotor speed, tail-rotor speed) and uses a remote control connected to the PC via USB for steering. In our component refinement state

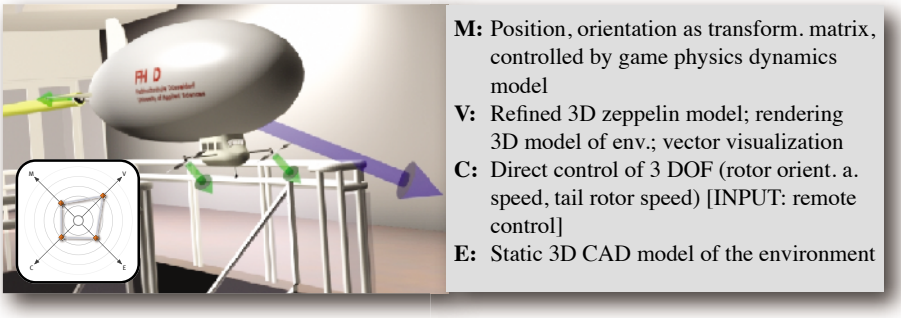

Fig. 5. Prototype using game physics

diagram all points placed one cycle away from the middle towards more realism and complexity. Now the user application can steer the zeppelin by using a standard remote control, which is more realistic than using the keyboard and thus better suited for training. Also the game physic engine create more realism in the motion-behavior.

4.3 Prototype 3: MATLAB/Simulink Simulation

Because the game physic engine is not as realistic as it should be, we build a MATLAB/Simulink [11] model for calculating the transformation and orientation in the third prototype. We also added an automatic hight control to our model, whichensure that the zeppelin retains a preset height. For the environment we uses a live image of the real environment (figure 6). The 3D CAD model used in prototype 2 now is needed as collision object and for height measurement. For the view we have the same airship-model as in prototype 2. The virtual representation of the environment is replaced by the live video. The parameter of the automatic height control is added to the controller. The input device is the same remote control as in prototype 2, additionally a slider and a button on the remote control is mapped to the height control for setting the hight or enable / disable the height control.

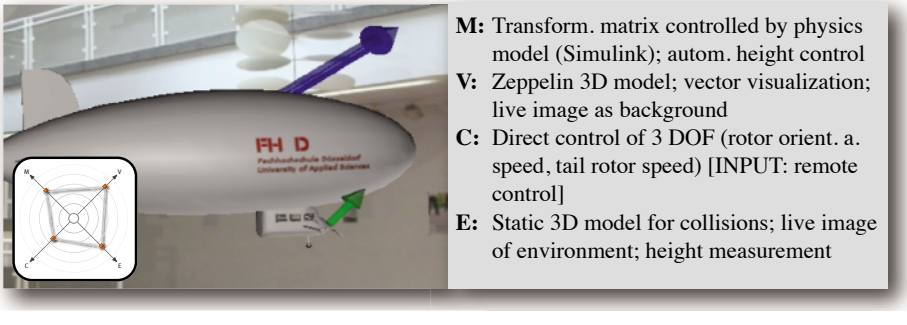

Fig. 6. Application using the MATLAB/Simulink model and a video background

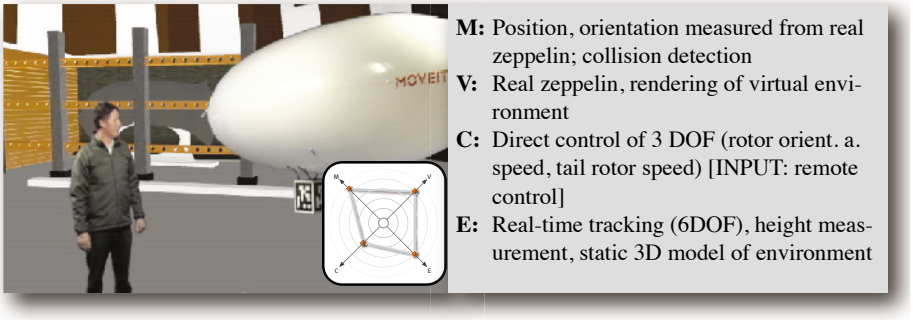

Fig. 7. AV scenario

4.4 Prototype 4: AV Scenario

In our next prototype we exchange the virtual model with the real zeppelin (figure 7). To minimize possible damage the airship fly in a large blue room with no obstacles. We added virtual obstacles to the scenario to build a parkour to fly through.

The MATLAB/Simulink model is replaced by the real zeppelin. It is tracked to get the actual position and orientation as transformation matrix. This transformation matrix is used by the collision detection, which controls the zeppelin when it collide with a virtual obstacle. This is done by calculating the collision vector and mapping it on the propeller of the zeppelin. At this point the user input is ignored and the collision detection takes over the control of the airship until the collision is resolved. For the environment we uses the real-time tracked zeppelin, the height measurement and the static 3D CAD model of the real environment. The real zeppelin itself is now handled as a part of the environment and only it‘s transformation matrix has to be acquired by appropriate tracking sensors. For the view we have the real zeppelin flying around in a virtual environment. The controller and input device is the same as in prototype 3.

4.5 Prototype 4: AR Scenario

In our last prototype (figure 8) the user control the real zeppelin in the real environment. We visualize the force vectors of all three propeller and the resulting force for the zeppelin as we did in prototype 3.

Our model consists of the position and orientation, which is tracked from the real zeppelin. The model itself is very simple compared to the model in MATLAB/Simulink, but as we control the real zeppelin, it is the most realistic one. The environment contains the real-time tracked zeppelin and the real environment. On the view-axis of our component refinement state diagram the environment and the zeppelin are real and in addition we have virtual visualization for the forces.

The controller is the same as in the AV prototype, which outputs the 3 DOF for controlling the zeppelin directly. As input device we first used the standard remote control, but then exchanged it by more complex input devices, including a gesture recognition using the Wiimote and a tracked TUI zeppelin model which is used to specify complete maneuvers (green dotted line in the diagram of figure 8).

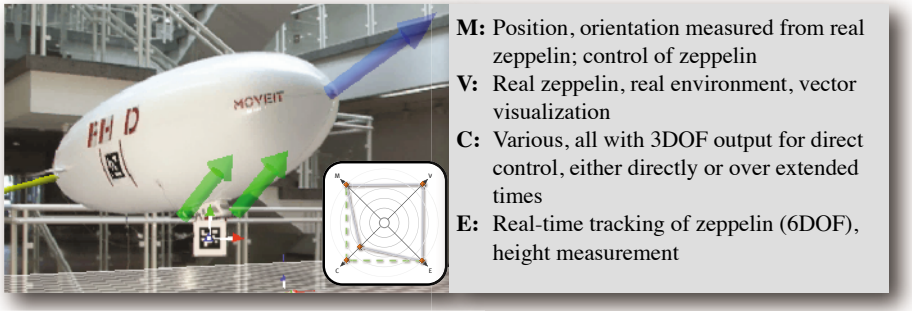

Fig. 8. Image of the AR application

5 Conclusion and Outlook

A key challenge in the development of mixed-reality (MR) applications is the use of ad-hoc development approaches and the lack of standardization and reuse. Most existing MR projects are research projects that focus on a single technology specific aspect and do not consider issues of tool development, content reuse and integration with other development processes. One area where this shortcoming is of central relevance is the reuse of software components in a flexible fashion. To exploit the full potential of MR interfaces a systematic development approach is required. In this paper we have presented an iterative development approach that is based on structuring MR applications into model, view, controller and environment components that can refined individually. We have shown how this model was used successfully to iteratively develop a variety of refinements of a NGUI to control a zeppelin and illustrated the benefits of component-wise refinement. The MVCE model helps to provide concrete support for software development and reuse in MR applications. In the future we aim to refine our process and experiment with extended tool support for MVCE applications. An area of development support that is still left largely unaddressed is thereuse of MR augmentation content between applications. A way to describe such models in a standardized and interchangeable format is still lacking and clearly requires more research in the future.

References

1. Shaer, O., et al.: User Interface Description Languages for Next Generation User Interfaces. In: CHI 2008 extended abstracts on Human factors in computing systems, ACM Press, Florence, Italy (2008)
2. Burbeck, S.: Applications Programming in Smalltalk-80: How to use Model-View-Controller, MVC (1992)
3. Ishii, H.: Tangible User Interfaces. In: Sears, A., Jacko, J. (eds.) Handbook of HCI, 2nd edn., Lawrence Erlbaum Association, Mahwah (2008)
4. Milgram, P., Kishino, F.: A taxonomy of mixed reality visual displays. In: IEICE Transactions on Information and Systems (1994)

5. MacIntyre, B., et al.: DART: A Toolkit for Rapid Design Exploration of Augmented Reality Experiences. In: User Interface Software and Technology, UIST 2004 (2004)
6. Broll, W., Herling, J., Blum, L.: Interactive Bits: Prototyping of Mixed Reality Applications and Interaction Techniques through Visual Programming. In: International Symposium On 3D User Interfaces, Reno, USA (2008)
7. Cuppens, E., Raymaekers, C., Coninx, K.: A model-based design process for interactive virtual environments. In: 12th International Workshop on DSVIS (2005)
8. Mixed Reality User Interfaces: Specification, Authoring, Adaptation. MRUI 2007. Charlotte, North Carolina, USA (2007)
9. Geiger, C., et al.: HYUI: a visual framework for prototyping hybrid user interfaces. In: TEI 2008: Proceedings of the 2nd international conference on Tangible and embedded interaction (2008)
10. Havok: Havok Physics (2009), http://www.havok.com
11. MathWorks, T.: MATLAB/Simulink (2009), http://www.mathworks.com