A Prototyping and Evaluation Framework for Interactive Ubiquitous Systems

Rapid prototyping of ubiquitous systems enables researchers and practitioners to quickly test and implement new ideas, but is also necessary to ensure the user's acceptance. In the following we introduce a framework that supports rapid prototyping and evaluation of ubiquitous interactive systems using a modular approach, incorporating different interaction modes.

Ubiquitous Systems often come with innovative design ideas and interaction concepts. The ultimate vision depicts intelligent and interactive environments, where computing devices are pervasive but barely noticeable, as first envisioned by Mark Weiser [1]. A ubiquitous environment optimally supports its users by providing easy information and computing access as well as natural and usable interfaces.

It is our understanding that ubiquitous computing not only comprises all kinds of devices, but also needs innovative, multimodal and natural interaction concepts to achieve usable pervasive user interfaces. It is necessary to test and evaluate those interfaces in early design stages, in order to avoid design errors. In order to develop and evaluate interaction concepts for ubiquitous environments, prototypes are essential.

We therefore developed a *prototyping* and *evaluation framework* for ubiquitous interactive systems designed to support rapid prototyping of interaction concepts in ubiquitous environments. The framework focuses on interaction prototypes and for this purpose provides an abstraction layer for the prototyping engineer that encapsulates different interaction channels. The prototyping engineer therefore doesn't have to implement several interaction techniques but can use the readily implemented interaction components and easily plug them into his backend code.

Our goal is to facilitate the usage of different *interaction channels* without limiting the flexibility of their application. The different interaction channels are therefore made available through a graphical user interface that allows customizing them, as shown in Figure 1. Each interaction channel can be adapted

for the usage in a new prototype. Once adapted, the interaction channels generate events that can be used to trigger responses to the specific interaction that have to be used in backend code. Prototypes that are realized within the framework can be evaluated within the framework itself, using the very same graphical user interface.

The framework is implemented in C# using the Microsoft Kinect sensor for the Xbox 360 but it is not limited to the use of the Kinect sensor. Up to now, we implemented four different interaction channels to support rapid prototyping of interactive ubiquitous systems. These are:

- Postures
- Gestures
- Speech
- Interactive Surfaces

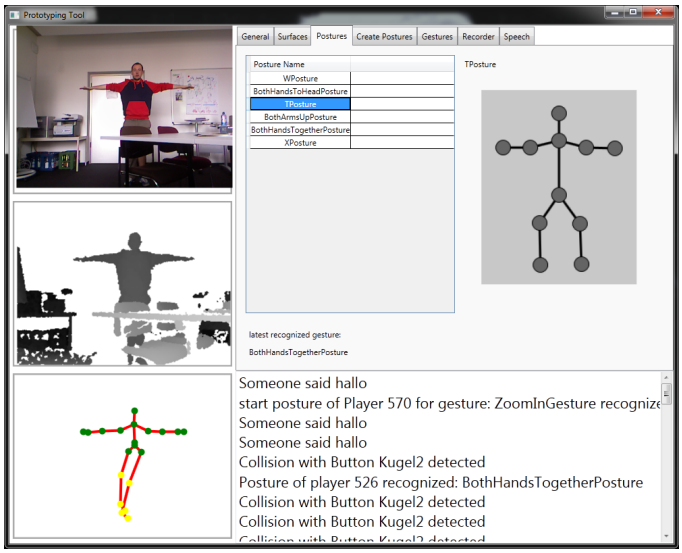

Figure 1: The graphical user interface of our framework.

The Posture Component can recognize designated positions of the skeleton the Microsoft Kinect sensor detects. For Posture development there is a graphical tool provided in the "Posture Creator" tab of the GUI, as shown in Figure 2. The Postures can be saved in an XML file and instantly used in a new prototype.

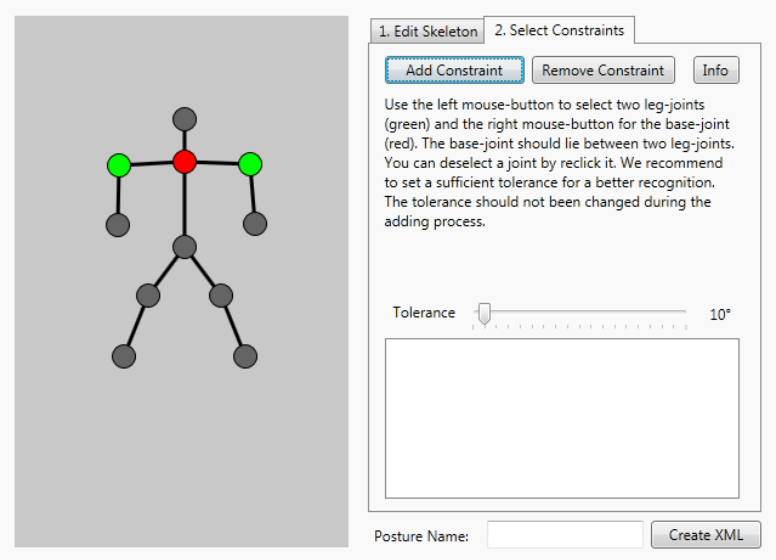

Figure 2: The Posture Creator.

The gesture recognition is realized in the **Gesture Component**. The framework comes with predefined gestures for easy use in a gesture library. Using the gesture tab of the GUI, all gestures of the gesture library can be selected and a picture of the postures that form the start and end of one gesture is displayed as well as a textual description of the gesture, as shown in Figure 3.

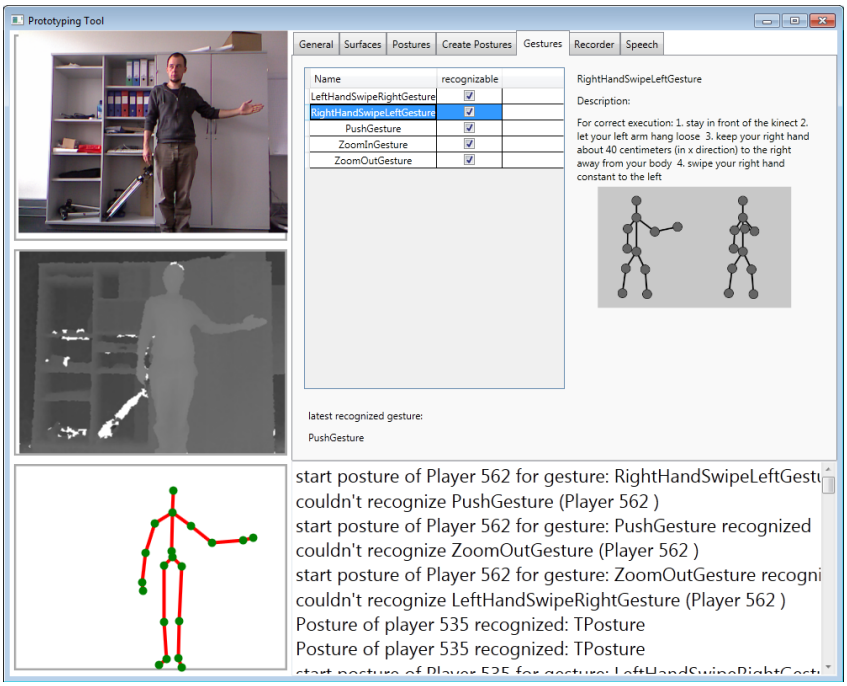

Figure 3: Using gestures from the gesture library.

We also realized a **Speech Component** using the Windows Desktop Speech API from Microsoft. In order to use speech recognition in a prototype, the prototype engineer can add new words and phrases for the usage as commands by typing them into the speech tab of our framework GUI.

The Surface Component is used to create interactive plane surfaces (any polygons) or volumes (e.g. spheres) defined by their coordinates in space. Interactions are triggered by collisions of a body-joint with these defined surfaces. Interactive surfaces can be created by directly drawing them into the depth frame of the Kinect sensor that is displayed in the GUI. Interactive surfaces can be exported and saved to a file for reuse and import in other prototypes and of course, they can also be deleted.

The graphical user interface also offers a view in which it is possible to record a user test. This is the Evaluation Component. The recordings of a user test consist of a log file which lists all recorded interactions that occurred during the test and a video of the user interacting with the prototype.

The framework was tested in some first case studies. As test cases, a ubiquitous music player was implemented, which only took four hours of work. An 3D Twister game was also implemented in half a day. Both case studies showed the practicability and acceptance of the approach.

[1] Mark Weiser. The computer for the 21st century. Scientific American, 265 p.94–104, 1991.