WINGED: Window Management with In-Air Gestures on Desktop Computers

Simon Walter, Timo Wilhelm
Interaction Engineering Winter Semester 2015/16
Prof. Dr. Michael Kipp
University of Applied Sciences Augsburg
simon.walter | timo.wilhelm1@hs-augsburg.de

***Abstract*—WINGED is a research project which examines a new way of interacting with window based desktop operating systems. In contrast to the common mouse focused interaction concepts, WINGED is offering a motion based system providing intuitive gestures to the users. Due to the higher degree of freedom compared to a mouse, extended interaction possibilities offer a topic for further research. For our proof of concept implementation, a Leap Motion controller is used to capture the hand movement with infrared sensors. Furthermore a desktop environment mockup, created in java-processing 3.0 is used to give visual feedback to the user.**

I. INTRODUCTION

Human-computer interaction has been dictated by the keyboard and mouse combination for over four decades. Especially the mouse has become a favourite for most users, while others criticize the disruption of workflow, switching from keyboard to mouse and back.
With the technological advancements of the last few years, freehand computer interaction is becoming more and more viable. Consumer motion tracking devices like the Leap Motion controller are making it possible to have science-fiction become reality. With the new interaction methods come a range of new questions, like "what are the most intuitive gestures?", "how is the performance compared to a mouse?" and of course "how can this be used for interaction with desktop computers?".
To answer some of those questions, we tried to simulate a window based desktop operating system with common functionalities like resizing and moving of windows, split screen, minimizing and closing.
We used java-processing 3.0 (www.processing.org) to create a prototype and implemented three simple gestures. We decided to use a Leap Motion controller (www.leapmotion.com) to track the hand movement of the user, as it is currently leading consumer-based tracking hardware. It uses two infrared sensors to create a 3D image of the hands and provides positional data of all the bones and joints for the developer. We ended up with a working proof of concept that helps to visualize the idea of operation with motion controls instead of a mouse (See Fig. 1).

Fig. 1: Hardware setup with Leap Motion controller

II. RELATED WORK

Relevant areas for our project include usability studies for both interacting with a desktop environment and with gesture interfaces.

Usability of gesture interfaces:
With the recent developments regarding virtual environments, motion and gesture interfaces have seen a lot more significance. Cabral et al. [2] have conducted studies on the usability of gesture interfaces. While not directly applicable on our project, this research has produced significant results in relation to the performance of motion control. The conclusion of this study shows, that the efficiency of tasks performed by gesture interfaces, is considerably worse than those performed by mouse movement. This result can be explained by the fact, that most participants lack the experience for this interaction method compared to the common mouse controls. Interviews with the users have determined, that motion-based gesture controls are most suitable for short and infrequent tasks, as they are easy to learn.

Usability Analysis of Gesture-Based Control for Common Desktop Tasks:
Extensive research regarding the use of gesture for desktop

control has been conducted by Farhadi-Niaki et al. [1]. The studies have shown that users find arm gestures to be a lot more fatiguing than finger gestures. Small scale finger gestures have also been perceived as a more natural way of interacting with a vision-based system. For the usability study, the users have been asked to think of the most natural way, they would interact with the objects on the screen. This resulted in 85% of participants choosing the pinch gesture to select and interact with the virtual items.

III. Prototype

We decided to use java-processing 3.0 (www.processing.org) to implement our proof of concept, as it provides an easy framework for rapid prototyping and libraries are available to map the raw input data from the Leap Motion controller (https://github.com/nok/leap-motion-processing).
The implementation of our prototype consists of two main components: The gesture recognition that captures the input data as 3D points and maps them to the associated window management functions. While the Leap Motion software provides some rudimentary gesture recognition functions, we decided to implement our own gesture detection code to be able to better fit our requirements. For moving and scaling, the difference in position between two frames is used to determine the speed.

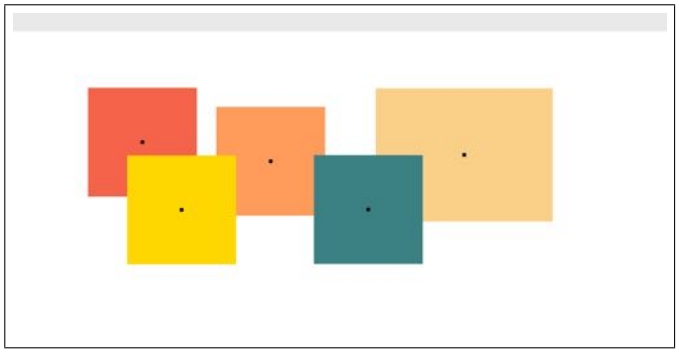

Fig. 2: Mockup of desktop operating system

Instead of having the software directly interact with the underlying operating system, we implemented a desktop mockup with colored rectangles representing the windows. In early stages of development we had a cursor representing the current position of the mapped hand position. After some experimenting we decided to change it in favor of a relation based system. The current iteration has the closest window to the users mapped position selected, even if the user is not directly hovering over a window. The top of the desktop representation has a indication bar showing minimized windows and the currently detected gesture to give the user more feedback on what the system is recognizing.
The currently selected window is indicated by a red border and put in the foreground. Minimizing a window is indicated by the window flattening horizontally and moving towards the top bar while closing is represented by a large red cross on the window and the window rapidly closing vertically before disappearing.

IV. Interaction mechanisms

The prototype maps gestures to different use cases of managing desktop windows. We focused entirely on basic managing functionalities which are necessary and useful to the user. The prototype can be easily extended with functionalities and gestures in the future.

A. Hand gestures

With the combination of different gestures the prototype provides a large number of interaction mechanisms based on just a few simple gestures. The project uses three different gestures for its first prototype:

1) Pinch Gesture:
The most basic gesture of the project is the *Pinch Gesture* (see Fig. 3). The prototype will recognize the pinch strength of the users index and middle finger with the help of the library leap-motion-processing. We used the strength and developed our own gesture event by refining the values. The gesture is mainly used for a drag-and-drop like behaviour of a window and is enabled for both the left and the right hand.

Fig. 3: Pinch Gesture

2) Swipe Gesture:
The second gesture is the *Swipe Gesture* (See Fig. 4). It consists of the two variants “swipe up” and “swipe down”. It is based on a method of the leap-motion-processing library which returns the angle of the hand. In the prototype the gesture is implemented for the left hand only.

3) Expand Gesture:
The third gesture is the *Expand Gesture* (See Fig. 5). Like the previous two gestures is it based on a functionality of the leap-motion-processing library. This library provides the grab strength of a hand, or rather how much a fist is clenched. With this information we implemented the expand gesture, which recognizes a fast unclenching of the fist. In the prototype only the right hand is supported but an extension for the left hand is possible.

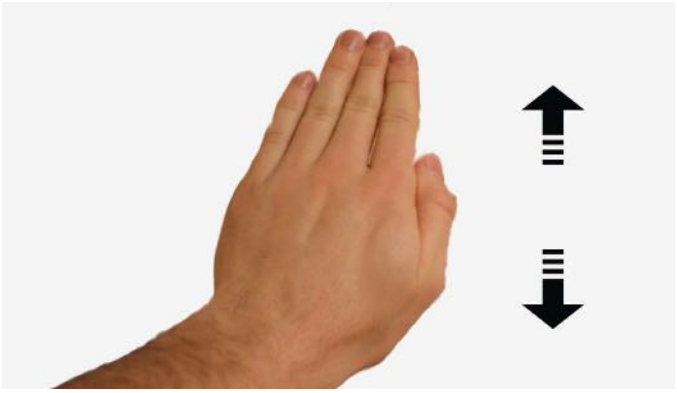

Fig. 4: Swipe Gesture

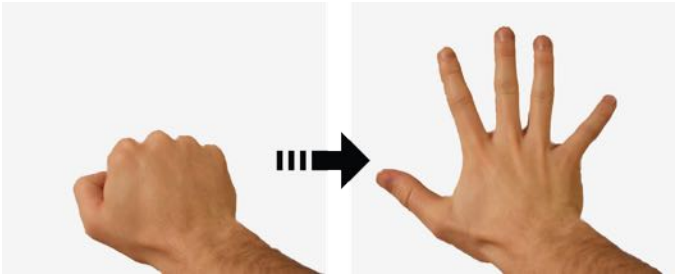

Fig. 5: Expand Gesture

B. Use cases

Our concept aims at seven use cases, which are invoked by using defined gestures in specific situations:

1) Move Window:
The first use case is *Move Window*. The user can select a single window with a pinch gesture of his right hand and then drag-and-drop the window to another position of the desktop.

2) Resize Window:
With a second pinch gesture of the left hand and the movement of both hands from and to another, the user can increase and decrease the size of a selected window.

3) Minimize Window:
The use case *Minimize Window* works similarly to the previously explained *Close Window* action. The functionality can be triggered by performing an upwards oriented swipe motion with the left hand, while selecting a window with the pinch gesture on the right hand. After minimizing a window, a small circle in the top status bar indicates the condition the window (See Fig. 6). All windows can be brought back to the foreground by the *Show all Windows* movement.

4) Close Window:
By doing a pinch gesture with the right hand for selecting a window and a swipe down gesture with the other hand, the user can close a window. The closing is visualized with a red cross and a shrinking animation. Unlike a minimized window, a closed window cannot be opened again afterwards.

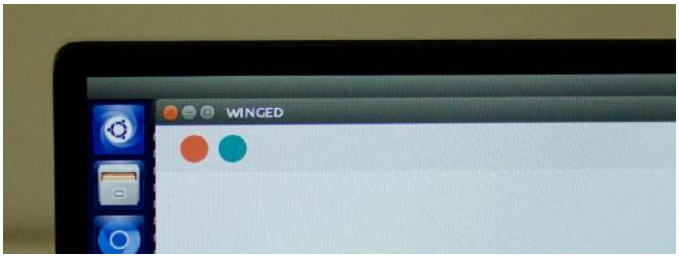

Fig. 6: Circle indicator for minimized windows

5) Split Screen Mode:
Split Screen Mode is an extension of the first use case. By dragging a window within 10 pixels to either the right or left side of the display, the window will be maximized to the respective half of the screen (See Fig. 7). This way two windows can be arranged in a split screen mode. This mode can be exited by performing the *Show all Windows* action.

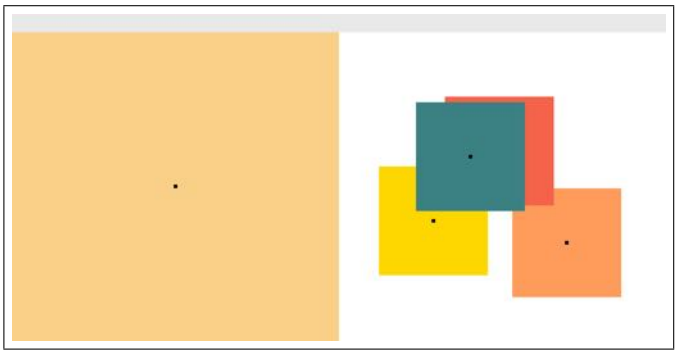

Fig. 7: Split screen Mode

6) Change Proportions of Split Screen:
The user can alter the proportion of the windows by pinching anywhere on the screen and moving the middle separation line right or left.

7) Show all Windows:
With the expand gesture, all windows can be shown. This can be used to exit split screen mode, or to bring minimized windows into the foreground. For the prototype, the windows are arranged randomly.

V. Outlook

A subsequent research in terms of a user study may determine which gestures feel the most natural and intuitive to the users. Thereby it is useful to test multiple different gestures for each functionality and to compare the results. The convenience

and practicability of the gestures can be identified by a follow-up user survey.
Furthermore the prototype can be transferred to a real operating system in a future work. At the moment the prototype of WINGED is a simple user interface built with Processing 3 representing a common operating system of a computer. With the knowledge of this project the functionalities of WINGED could be ported to Windows, Mac OS X, Linux and others.
In further projects the Leap Motion Keyboard by Hewlett-Packard may be supportive. This keyboard has a built in Leap Motion sensor which makes it the ideal device for future versions of WINGED.

VI. CONCLUSION

In this project we captured in air gestures with a leap motion controller to manage application windows on a mockup desktop operating system. With this we wanted to provide a solution for the following problem:
The functionality of managing windows on common operating system is often hidden behind abstract buttons in a menu bar on top of an application. These buttons are neither easily understood nor intuitive to handle. Additionally the user needs a mouse to use these operations. This leads to a permanent switch from the keyboard to the mouse and back which interrupts the users workflow constantly.
With WINGED there have been implemented seven use cases, which try to solve these problems. Thereby different combinations of simple gestures enable an easy usage. While the prototype worked quite well in most situations, the quality of tracking provided by the Leap Motion controller proves to be a challenge in some situation, especially when a lot of ambient infrared light is present (sunlight). As technology advances, those problems will be overcome.
With this project we developed a first usable version of window management functionalities with gestures. This can be used as a foundation for subsequent improvements and exciting advancements.

REFERENCES

[1] F. Farhadi-Niaki and S. A. Etemad and A. Ary, *Design and Usability Analysis of Gesture-Based Control for Common Desktop Tasks* University of Ottawa, Carleton University, Ottawa: Springer-Verlag Berlin Heidelberg, 2013.

[2] M. C. Cabrali and C. H. Morimoto and M. K. Zuffo, *On the usability of gesture interfaces in virtual reality environments* University of Sao Paulo: CLIHC '05 Proceedings of the 2005 Latin American conference on Human-computer interaction p. 100-108, New York, 2005.