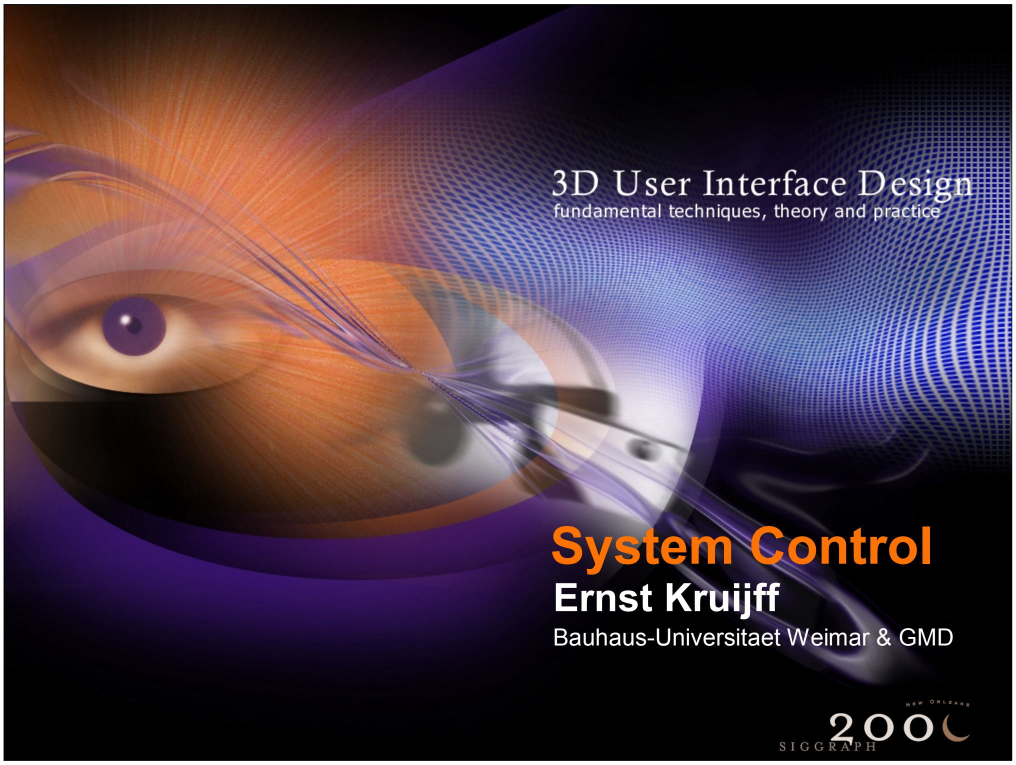


System Control

Ernst Kruijff

Chair for CAAD & Architecture (Prof. Donath), Faculty of Architecture
Bauhaus-Universitaet Weimar, Germany

Virtual Environments group (IMK.VE)

German National Research Center for Information Technology

http:	http://www.uni-weimar.de/~kruijff
email:	ernst.kruijff@archit.uni-weimar.de

3D User Interface Design
fundamental techniques, theory and practice

Lecture outline

- Definition
- Categorisation
- Description of different techniques
- Design guidelines
- Myths
- Conclusion
- Resources

Ernst Kruijff   System Control   2

When we interact with a system, we often require a way of issuing commands to access the available functionality. Within desktop environments, the issuing of commands has received much attention, resulting in a clear understanding of the basic issues of system control. Even with a large body of available commands with a complex structure, users can still manage to interact with desktop systems.

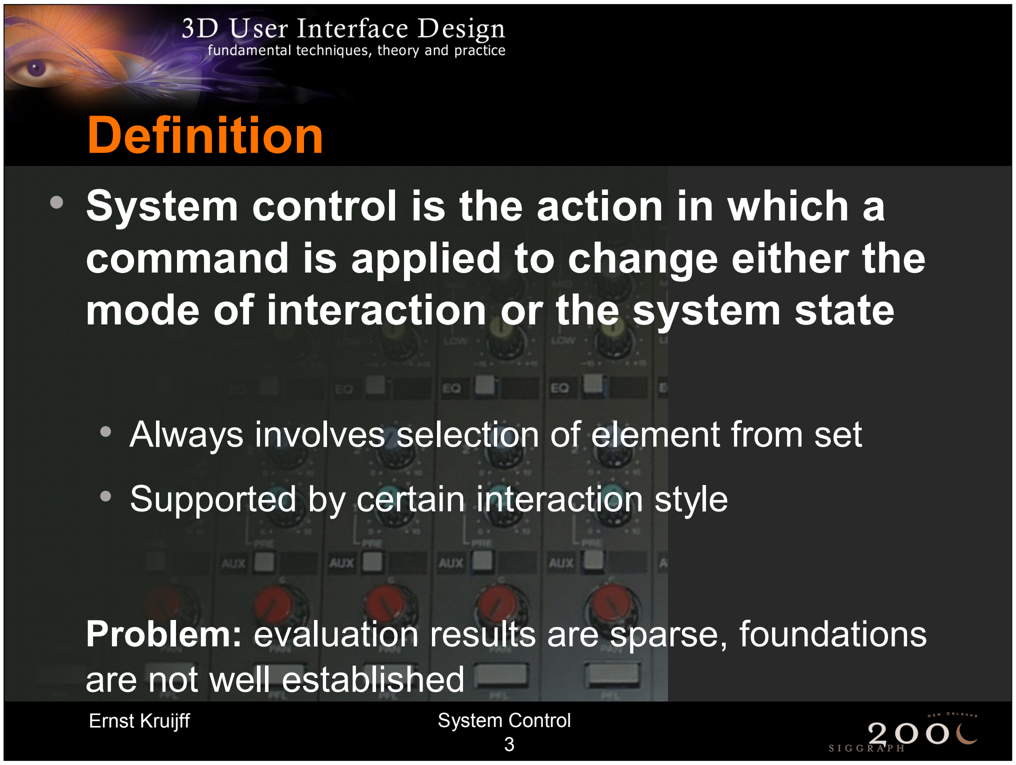
Why a lecture on 3D system control?

Unfortunately, we can not simply transfer our knowledge of system control in desktop environments into three dimensional environments. The action of system control in 3D environments differs too much from its 2D equivalent: users have to deal with many more degrees of freedom for selection, input and output devices differ considerably, and the extra (3rd) dimension poses new difficulties and possibilities on representation of interfaces.

What are the main aims of the lecture, and how is the lecture organized?

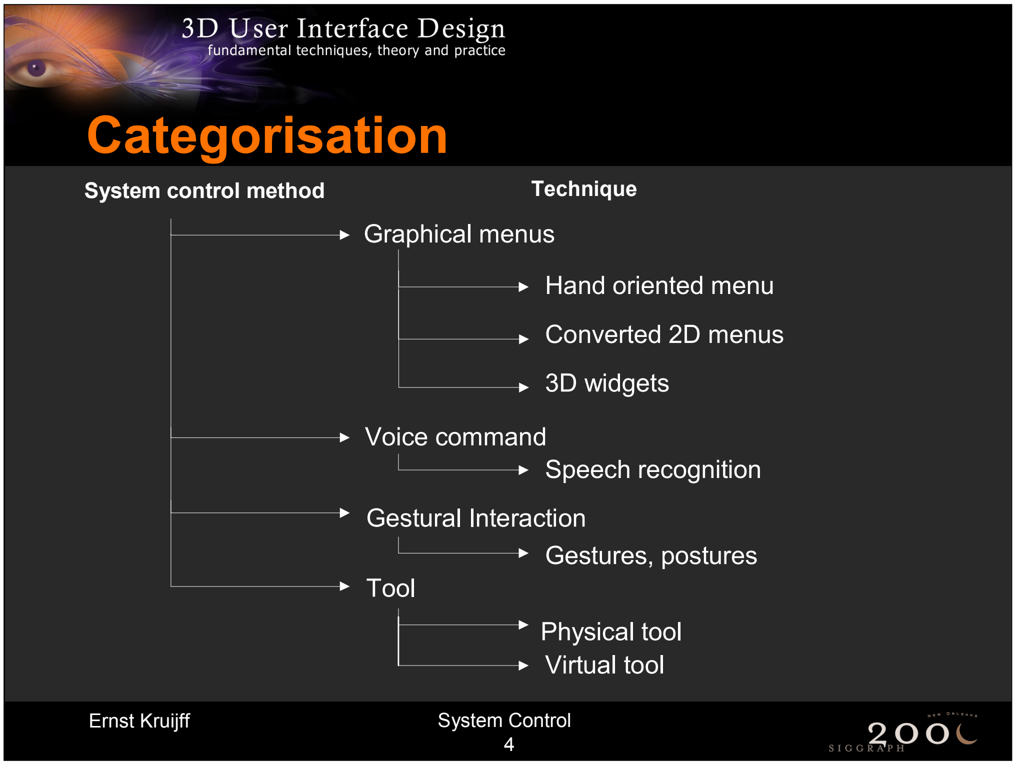
This lecture aims to supply a basic overview of what factors are involved when developing system control interfaces for three dimensional environments. The lecture is organized based on a categorisation, which will be illuminated by four groups of system control techniques which are at the moment commonly applied. From the descriptions of these techniques, several design guidelines can be extracted which will help the developer to implement system control in his/her system.

The lecture will be concluded by exposing several myths, and a conclusion. In addition, to support the interested developer, further reading resources are given that broaden the scope of this lecture.



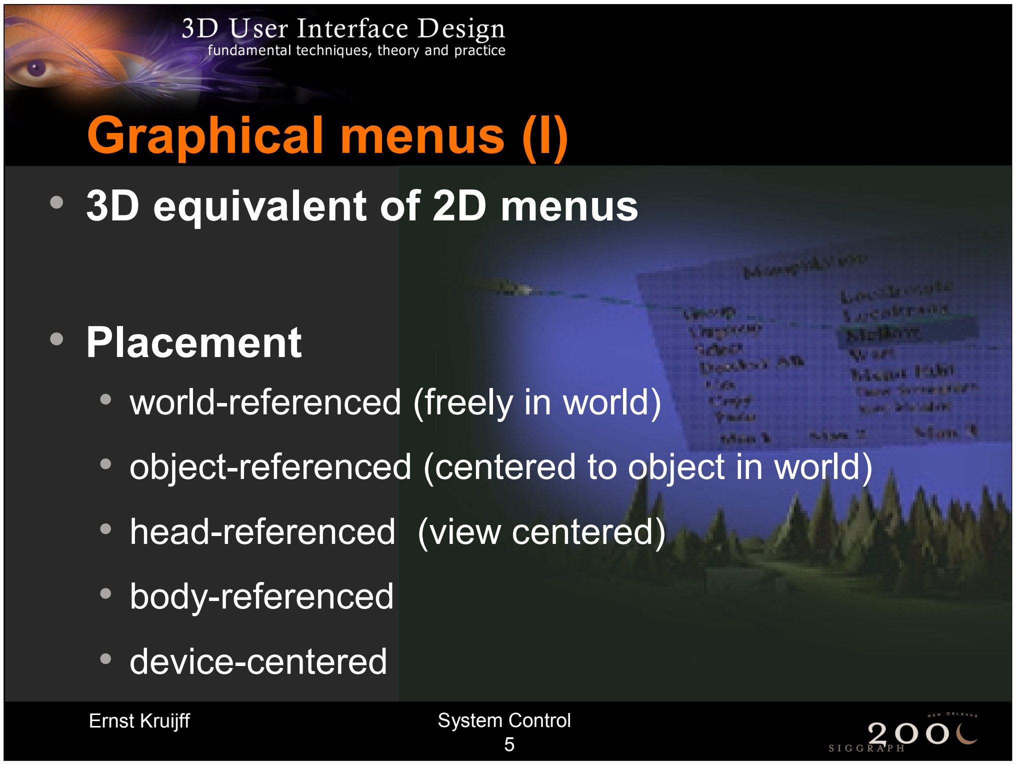
During the process of working with a system we often need to switch between different modes of interaction in order to perform the task we want to execute. In traditional, desktop based environments, we perform system control by accomplishing actions like selecting an item from a pull-down menu, or by clicking an icon in a toolbar. The extensive development of desktop based environments has provided us with a wealth of system control techniques, and a good understanding of how these techniques function. We are often unable to directly transfer WIMP style system control techniques into a virtual environment. 3D user interaction differs too much from desktop based interaction. However, the underlying action characteristics of three dimensional system control are analogous to desktop system control. To define system control, we can say that we use a command to either change the mode of interaction or the system state. In order to issue the command, the user has to select an item from a set. Due to the selection characteristic, a large overlap with selection/manipulation issues (discussed in a previous lecture) can be found.

The basic difference between 3D and desktop based system control is how we perform the action, or, which *interaction style* we apply. The traditional interaction styles are only partially usable, and developers often have to use non-conventional control techniques. In comparison to WIMP (windows, icons, menus, pointers) interaction styles, non-conventional control techniques are still in the early stages of development, and its boundaries are not well defined. Certainly there is a need for quantitative, formal evaluation of non-conventional control techniques in order to understand the effects of applying such techniques. The lack of such evaluation makes it difficult to define the basics of three dimensional system control. Good overviews of 3D system control are sparse, and the lack of basic knowledge results in relatively weak implementations of non-conventional control techniques.

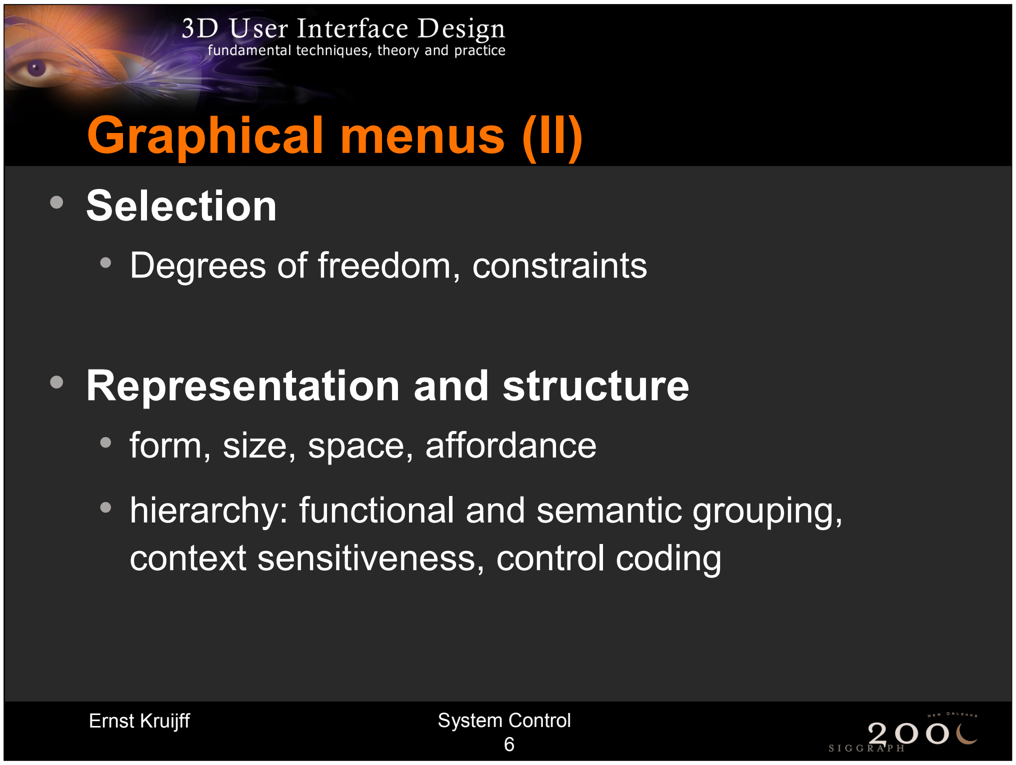


Most of the currently applied system control techniques are derived from a small number of basic metaphors. Influenced by the description of non-conventional control techniques by McMillan et al (in Salvendy, 1997), this categorisation describes four categories under which all commonly used system control techniques should fit. The categorisation is, in many ways, very device oriented. Several metaphors directly depend on a specific input or output device and can not be performed in a similar way with another device. Furthermore, degrees of freedom have also been taken into account, also with respect to input devices. The number of DOFs often coincides nicely with the usage of constraints, as will be shown in the description of several techniques.

The dependence on both devices and DOFs can also be found in the categorisation Lindeman (1999) introduced for manipulation techniques. In this categorisation, many system control techniques can also be found.



The first metaphor which will be described is the group of graphical menus. Graphical menus can be seen as the 3D equivalent of 2D menus. Placement influences the access of the menu (correct placement can give a strong spatial reference for retrieval), and the effects of possible occlusion of the field of attention. The paper by Feiner et al (1993) is an important source for placement issues. The authors divided placement into surround-fixed, world-fixed and display-fixed windows. The subdivision of placement can, however, be made more subtle. World-fixed and surround-fixed windows, the term Feiner et al use to describe menus, can be subdivided into menus which are either freely placed into the world, or connected to an object. Display-fixed windows can be renamed, and made more precise, by referring to their actual reference frame: the body. Body-centered menus, either head-referenced or body-referenced, can supply a strong spatial reference frame. Mine et al (1997) explored body-centered menus and found that the proprioceptive cues which are supplied by the reference frame can significantly enhance menu retrieval and usage. One particularly interesting possible effect of body-centered menus is “eyes-off” usage, in which users can perform system control without having to look at the menu itself. The last reference frame is the group of device-centered menus. These menus should not be mistaken with tools, which will be discussed later. Device-centered placement provides the user with a physical reference frame. A good example is the placement of menus on a responsive workbench, where menus are often placed at the border of the display device. Finally, take note of the effective screen resolution when placing an interface: as Darken (1994) showed, especially the borders of certain displays may cause visibility/readability problems.

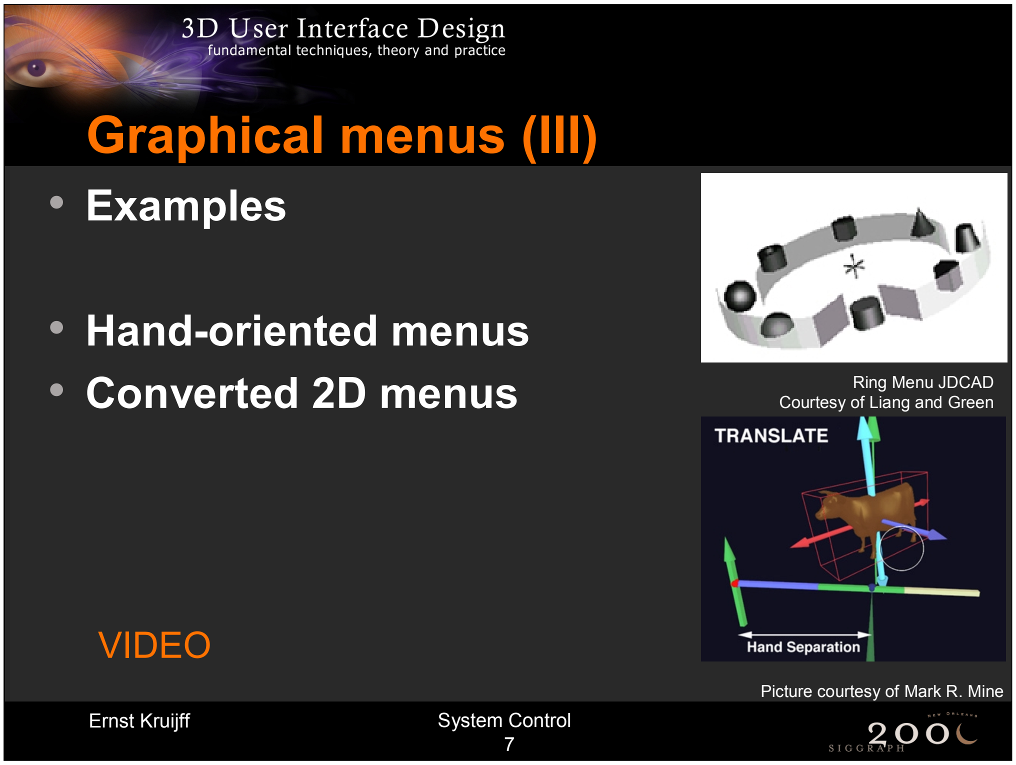


Traditional, desktop menu methods make use of a one dimensional selection technique. When we translate these widgets into a three dimensional environment, we obtain the problem of having to use similar menus with a three dimensional selection method. This makes interaction with the system control interface particularly hard. In order to solve this problem, several alternative selection methods for system control have been developed, by simply constraining the amount of dimensions of the system control interface. As will be described with several examples later, constraining the number of DOFs considerably improves performance.

The next issue in developing a graphical menu is its representation: how do I visualise the functions and, if there are many functions, how do I structure the interface?

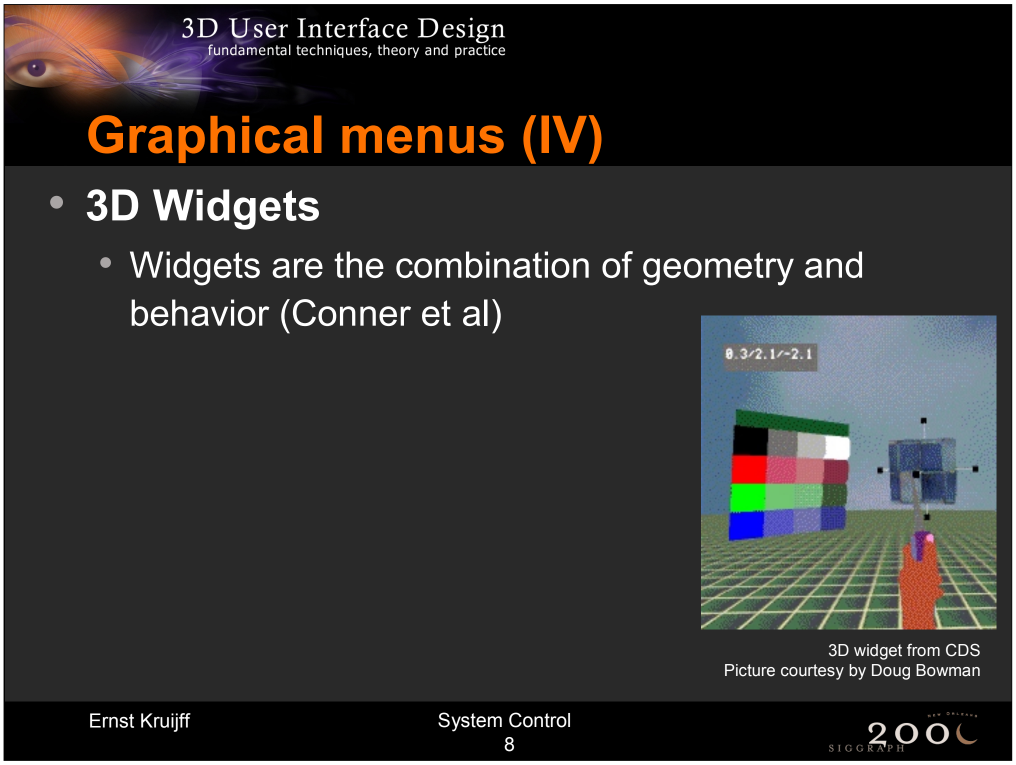
With respect to form and affordance, desktop icon-like forms funtion well, since many users are used to this kind of graphical communication due to their experience with desktop environments. Of course, several tricks like animating the icons can make affordance stronger, but one should take care not to overdo such communication. Partly due to technical limitations, the size of items which can be selected, and the space between those items is very important - don't make items and inter-item distances too small, or the user might well have problems selecting the item.

The more complex the application gets, the more functions will be available. Make sure to structure the interfaces, by either using functional or semantic grouping (refer to Salvendy 1997), or by subdividing functions in several context-sensitive menus. Finally, control coding can give an extra cue about the relations between different items in a system control interface, and therefore make the structure and the hierarchy of the items clearer to understand. Control coding methods, as described by Bullinger et al, can be applied by utilizing color, shape, surface, texture, dimension, position, text and symbols.



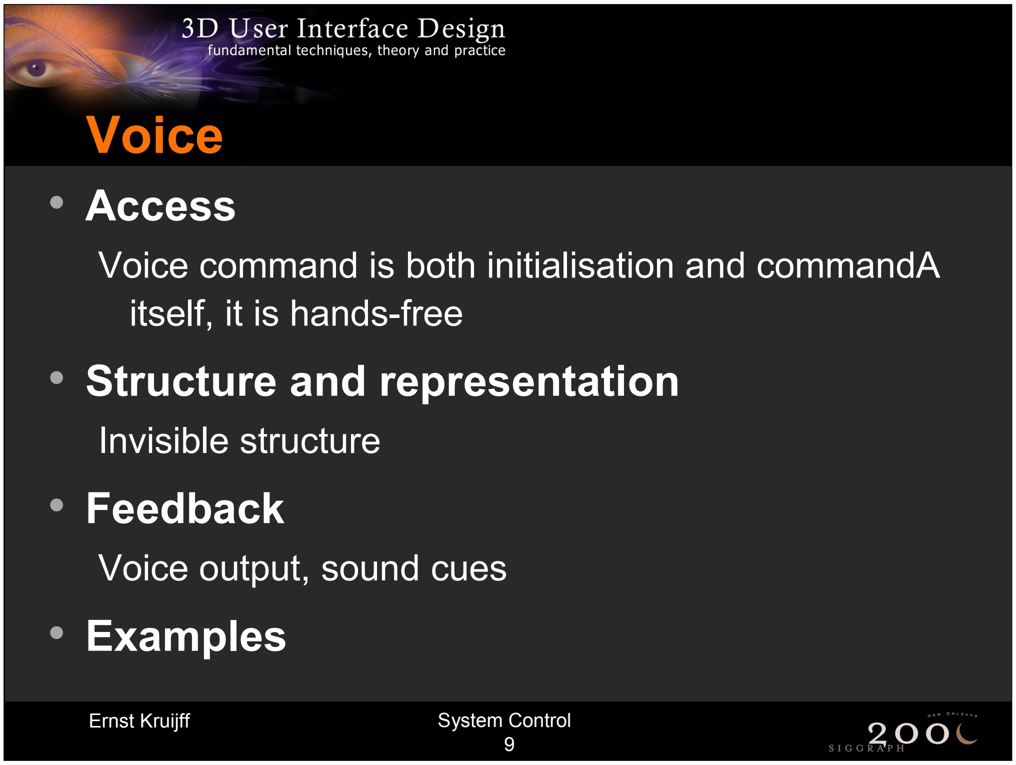
As one can see in the categorisation given earlier, we can subdivide graphical menus into hand-oriented menus, converted 2D menus and 3D widgets. 3D widgets can be seen as a graphical menu system, though they different in certain aspects and will therefore be handled separately. One can identify two major groups of hand-oriented menus. 1DOF menus are menus which use a circular object on which several items are placed. After initialisation, the user can rotate his/her hand along one axis until the desired item on the circular object falls within a selection basket. The performance is highly depending on hand and wrist physical movement and the primary rotation axis should be carefully chosen. 1DOF menus have been made in several forms, including the ring menu (see picture), sundials, spiral menus (a spiral formed ring menu), and a rotary tool chooser. The second group of hand-oriented menus are hand-held-widgets, in which menus are stored at a body-relative position. Hand-held-widgets function use relative hand positions for selection of items. Please refer to (Liang and Green 1992), (Shaw and Green 1995), and (Mine 1997) for more details.

The second group is the most often applied group of system control interfaces: converted 2D widgets. These widgets basically function the same as in desktop environments, although one often has to deal with more DOFs when selecting an item in a 2D widget. Popular examples are pull-down menus, pop-up menus, flying widgets, toolbars and sliders.



The final group of graphical menus is the group of 3D widgets. In a 3D world widgets often mean moving system control functionality into the world or onto objects. This matches closely with the definition of widgets given by Conner et al (Conner et al 1992): “widgets are the combination of geometry and behavior”. This can also be thought of as “moving the functionality of a menu onto an object.”

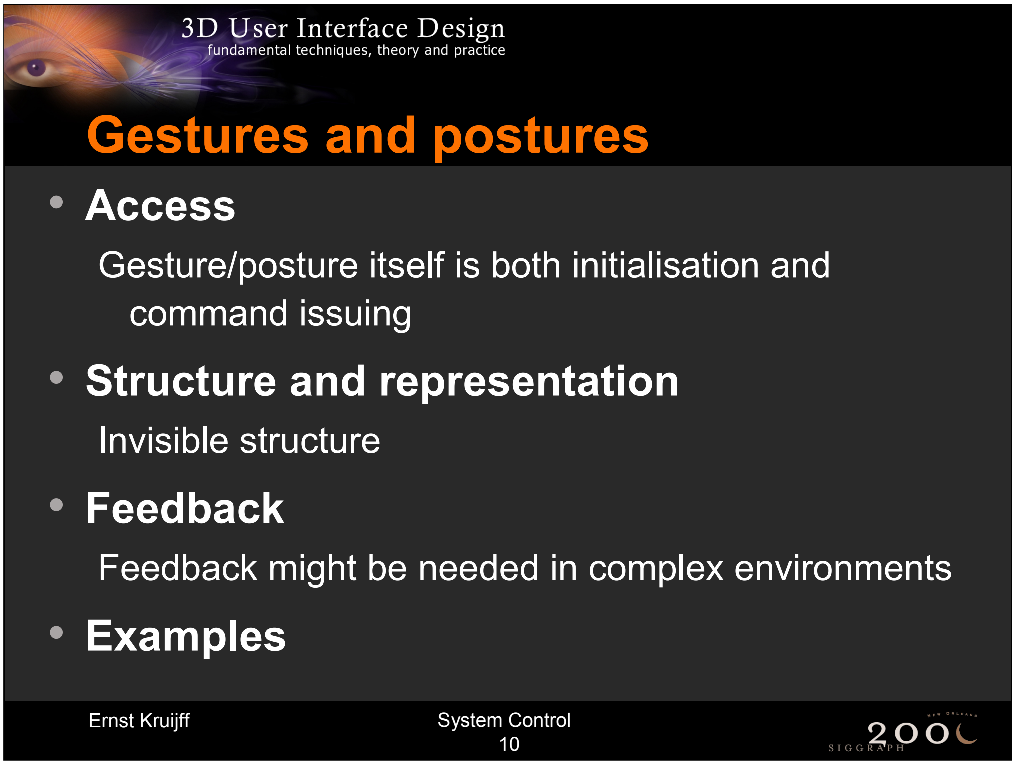
A very important issue when using widgets is placement. 3D widgets differ from the previously discussed menu techniques (1DOF and converted 2D menus) in the way the available functions are mapped: most often, the functions are co-located near an object, thereby forming a highly context-sensitive “menu”.



A voice input is both the initialisation, selection and the issuing of a command. Sometimes, another input stream (like a button press) or a specific voice command is used to allow the actual activation of voice input for system control. The usage of voice input as a system control technique can be very powerful: it is hands-free and natural. Still, continuous voice input is tiring, and can not be used in every environment. Furthermore, the voice recognition engine often has a limited vocabulary. In addition, the user first needs to learn the voice commands before they can be applied.

Problems often occur when applications are more complex, and the complete set of voice commands can not be remembered. As we will see later with gestural interaction, the structural organisation of voice commands is invisible to the user: often no visual representation is coupled to the voice command in order to see the available commands. In order to prevent mode errors, it is often very important to supply the user with some kind of feedback after he/she has issued a command. This can be achieved by voice output, or by the generation of certain sounds.

A very interesting way of supporting the user when interacting with voice and invisible menu structures can be found in telecommunication: using a telephone to access information often poses the same problems to the user as using voice commands in a virtual environment. Please refer to Brewster (Brewster 1998) for a further discussion of issues involved in such communication/interaction streams. Popular applications of voice input include the activation of widgets (“computer, give color palette”), or direct actions like exiting the system.

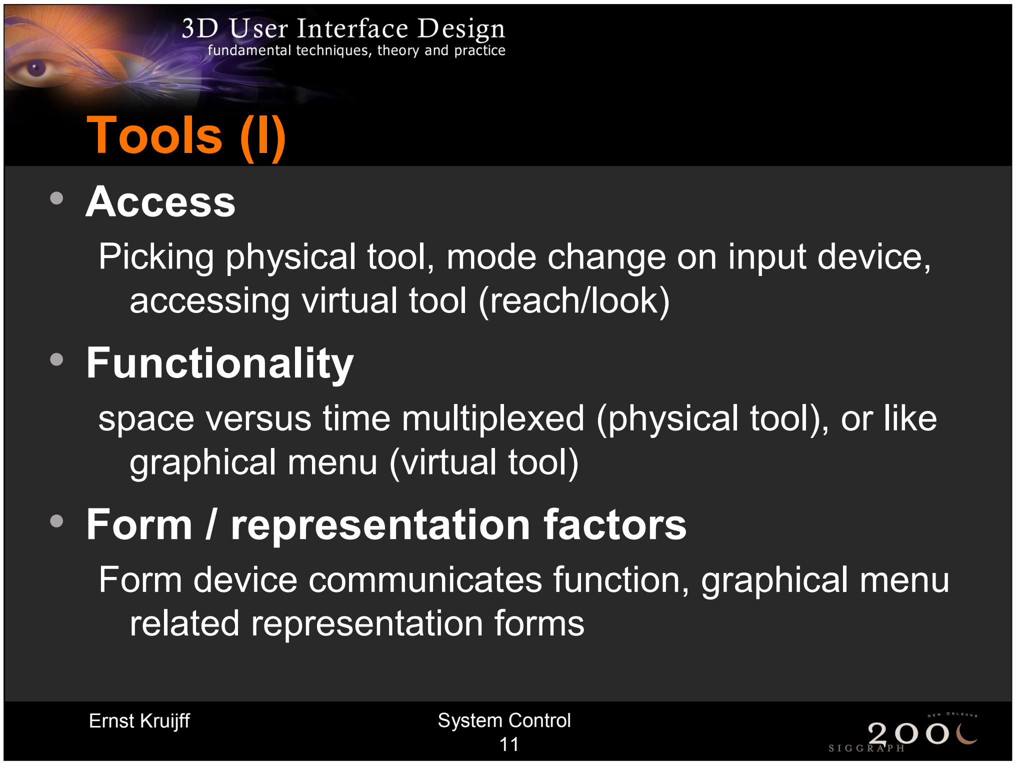


When using gestural interaction, we apply a “hand-as-tool” metaphor: the hand literally becomes a tool. When applying gestural interaction, the gesture is both the initialisation and the issuing of a command, just as in voice input. When talking about gestural interaction, we refer in this case to gestures and postures, not to gestural input with pen-and-tablet or similar metaphors. There is a significant difference between gestures and postures: postures are static movements (like pinching), whereas gestures include a change of position and/or orientation of the hand. A good example of gestures is the usage of sign language. From now on, we will talk about gestures and gestural interaction as the group of both gestures and postures.

Gestural interaction can be very powerful, though gloves can be uncomfortable, and calibration is not always very accurate. Another problem with gestural interaction is that the user needs to learn all the gestures. Since the user can normally not remember more than about 7 gestures (due to the limited capacity of the working memory), inexperienced users can have significant problems with gestural interaction, especially when the application is more complex and requires a larger amount of gestures. Users can often not refer to a graphical menu when using gestural interaction - the structure underneath the available gestures is completely invisible. In order to make gestural interaction easier to use for a less advanced user, strong feedback, like visual cues after initiation of a command, might be needed.

Powerful examples of gestural interaction can be found in Multigen’s SmartScene, and other examples talked about in the manipulation lecture: gestural interaction is often integrated into a manipulation action, so a large overlap can be found there.

For further reading on gestural interaction, please refer to (Bordegoni 1993), (Mapes and Moshell 1995), (LaViola 1999) and (Sturman 1992).



Principally, we can identify two different kinds of tools, namely physical tools and virtual tools. Physical tools are context-sensitive input devices, which are often referred to as props. A prop is a real-world object which is duplicated in the virtual world. A physical tool might be space multiplexed (the tool only performs one function) or time multiplexed, when the tool performs multiple functions over time (like a normal desktop mouse). One accesses a physical tool by simply reaching for it, or by changing the mode on the input device itself. Virtual tools are tools which can be best exemplified with a toolbelt. Users wear a virtual toolbelt around the waist, from which the user can access specific functions by grabbing at particular places at the toolbelt, as in the real world. Sometimes, functions on a toolbelt are accessed via the same principles as used with graphical menus, where one should look at the menu itself. The structure of tools is often not complex: as stated before, physical tools are either dedicated devices for one function, or one can access several (but not many) functions with one tool. Sometimes, as we will see in the examples, a physical tool as the display medium for a graphical menu. In this case, it has to be developed according to the same structural issues as graphical menus. Virtual tools often use proprioceptive cues (body references) for structuring.

A uniform answer to the question of how we design tools can not be given. Still, some general design issues can be stated. In the case of physical tools, the form of the tool often strongly communicates the function one can perform with the device, so take care with the form when developing new props. The form of a tool can highly influence the directness and familiarity with the device too. With respect to virtual tools which can not be used without looking at them, the representation is very similar to graphical menus. For further reading on tools, please refer to (Hinckley et al 1994), Mine (1997) and (Butterworth et al 1992).

3D User Interface Design
fundamental techniques, theory and practice

Tools (II)

- **Examples**
  - **Tablet interface**
  - **VRAM toolchest**

VIDEO

Gorilla Exhibit menu
Courtesy of Bowman and Hodges

Toolchest by Toralf Sontag and Hartmut Seichter

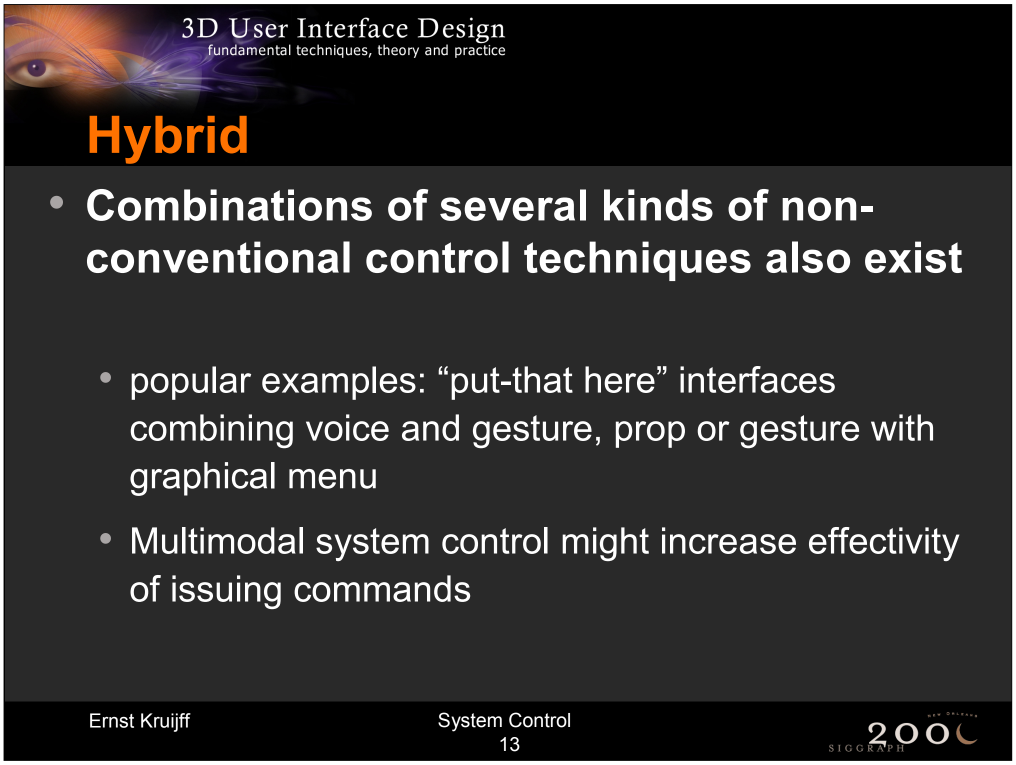
Ernst Kruijff
System Control
12

Here are two examples of tools. The first is a “pen & tablet” interface: in the Virtual Habitat (see the later lecture on VEs for Design Education), users hold a large plastic tablet on which a (traditional) 2D interface is displayed in the virtual world. Users are able to use graphical menu techniques and can move objects with a stylus within a window on the tablet. The tablet supplies the user with strong physical cues with respect to the placement of the menu, and allows increased performance due to faster selection of menu items.

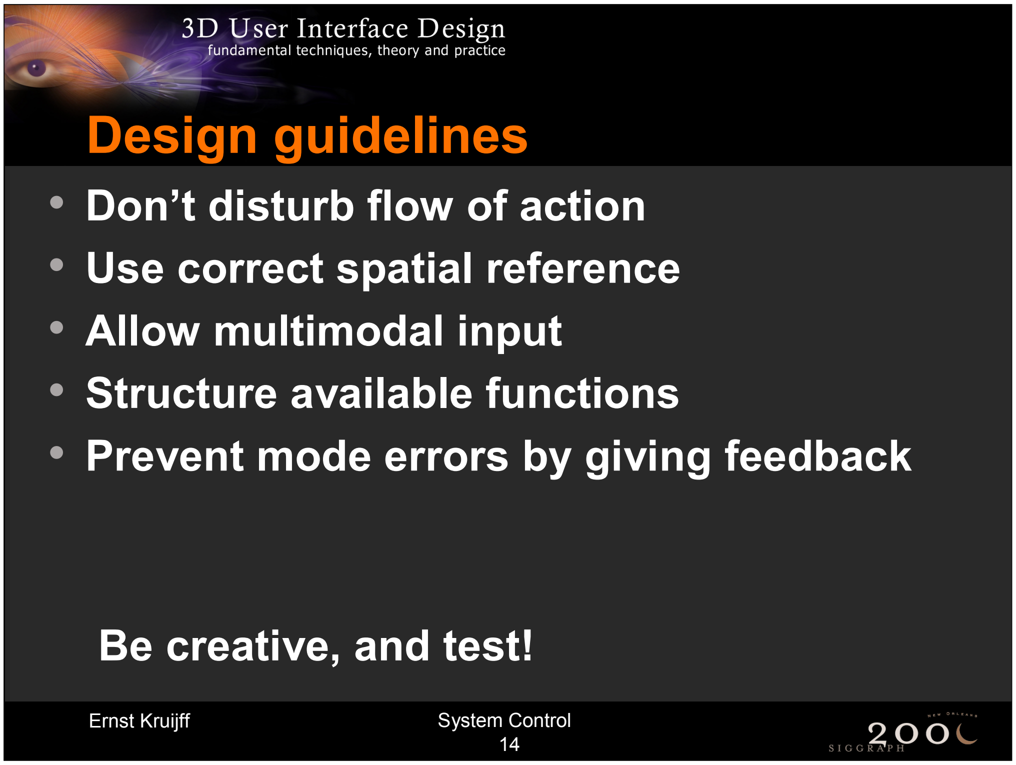
Reference: (Bowman, Wineman, Hodges, and Allison, 1999)

The second example shows a virtual toolchest in VRAM, a software environment developed at Bauhaus-Universitaet Weimar. With respect to placement, we should note that toolchests are easy to access and use, even if the user’s attention is focused away from the actual task. The size of widgets turn out to be another important factor: both for selection and readability, large widgets (elements) need to be created to overcome to limited resolution of our HMD and noise in the tracking, even though users are selecting the items within close reach.

For more information on VRAM, please refer to http://www.uni-weimar.de/iar and http://www.igroup.org .



In some cases, it might make sense to combine several kinds of system control, or non-conventional control, techniques. For example, think about cases in which both hands are in use, and a command has to be issued. In this case, one does not necessarily want to use one hand for issuing the command - voice input might help in this case. Other examples include the usage of gestures for clicking through items in a menu, and the display of menus on props. When appropriate, the usage of multimodal system control techniques can certainly increase the effectiveness and efficiency of issuing a command.

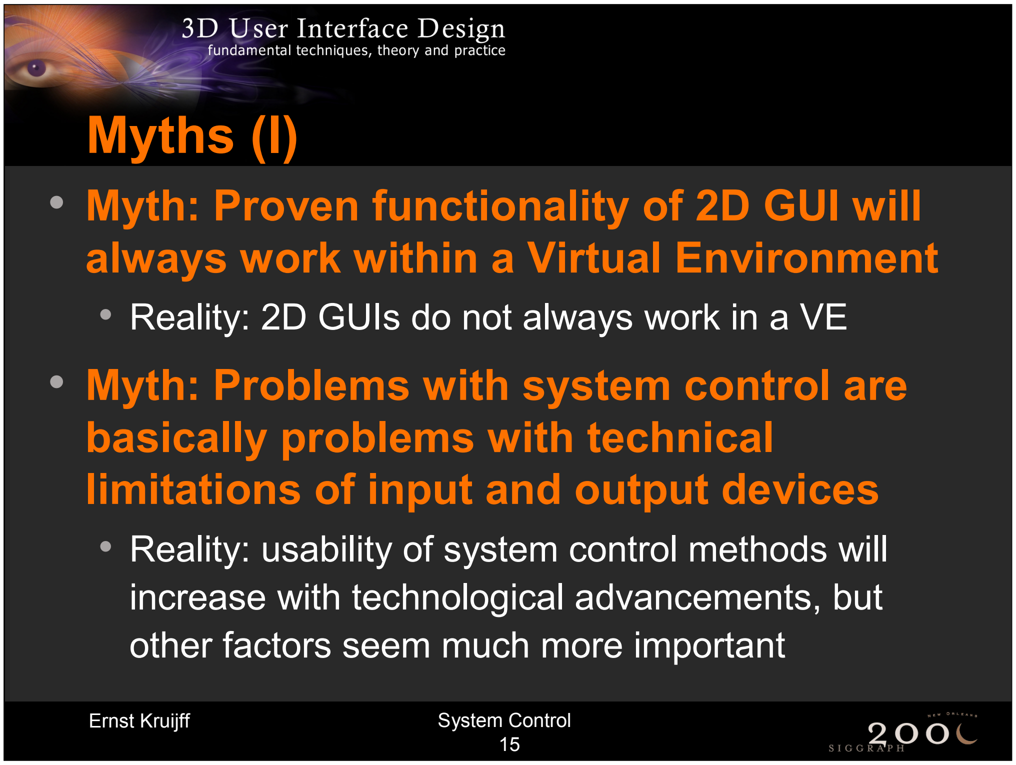


Extracted from the descriptions of system control techniques, several important design guidelines can be stated. Due to the relative lack of formal evaluations, these guidelines are primarily based on tendencies described by researchers and personal experience.

System control is often integrated within another universal interaction task. Due to this integration, we should avoid disturbing the flow of action of an interaction task. The user should stay focused on the task. “Modeless” interaction (where the mode changes are very natural) is ideal. One way of supporting the user to easily access a system control interface is by using a correct spatial reference. This guideline is of course mostly applicable to graphical menus, but tools also benefit from a strong spatial reference. Another method to allow a more seamless integration of system control into a flow of action is to use a multimodal, or hybrid, system control interface. Multimodal interfaces can increase the performance of issuing a command, and may allow multiple channels to access the system control interface. However, keep in mind that multimodal system control is not always suitable or applicable.

After the user has accessed a system control interface, he/she has to select an item from a set: when this set is large, i.e. when a large number of functions are available, one needs to structure the items. As stated in the guidelines on graphical menus, this might be achieved by methods like using context-sensitivity, or by clearly communicating the hierarchy of items and (sub)menus.

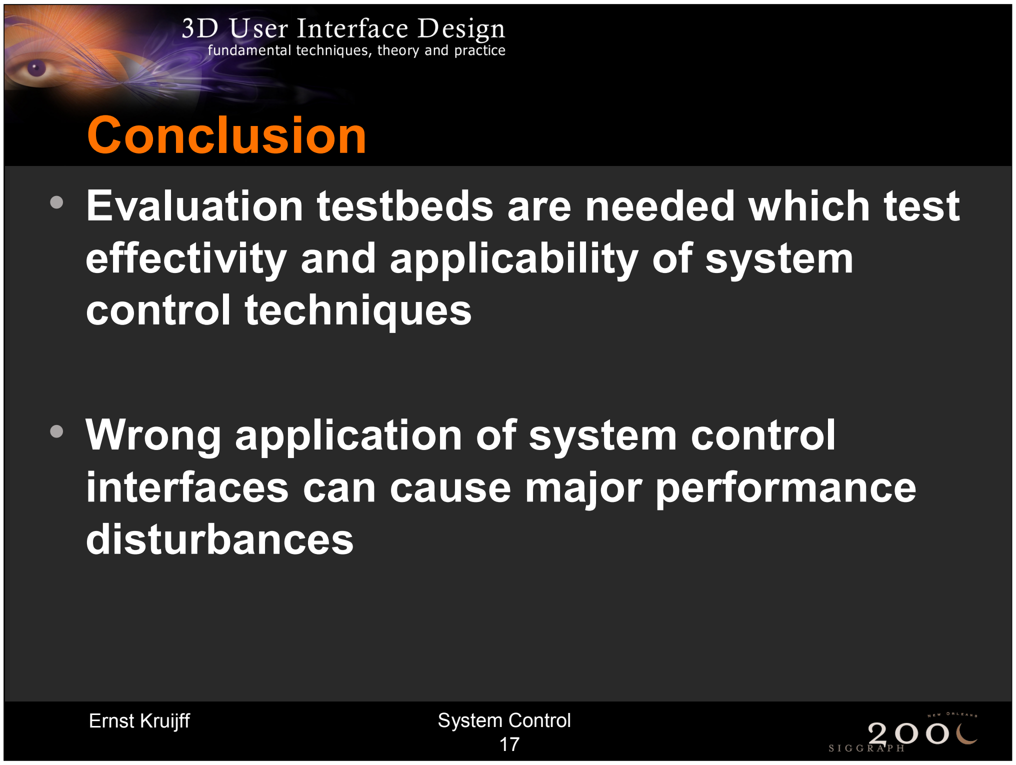
Finally, always try to prevent mode errors by providing the user with appropriate feedback during and after selection of a command. Mode errors can be highly disturbing and they interrupt the flow of action in an application.



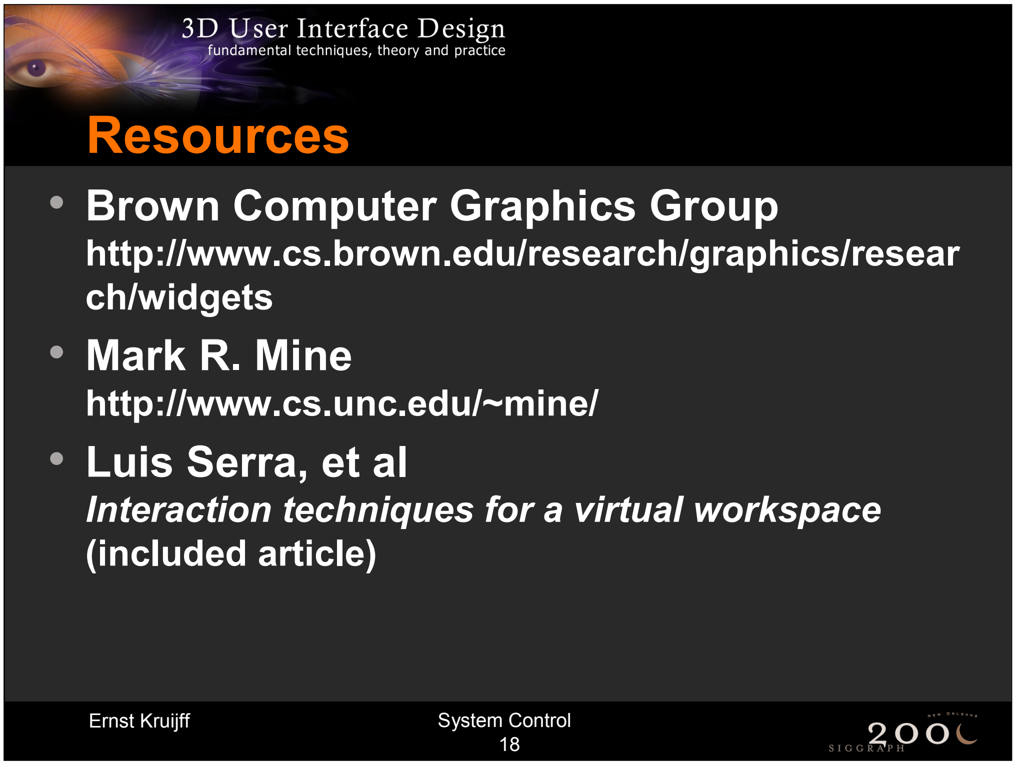
The lack of substantial formal evaluations and the sparse availability of basic knowledge in this area certainly have created myths relating to the application of system control interfaces. Here, we have listed three examples of myths which can easily cause misunderstandings by developers, and which might easily cause a performance decrease in a application.

Myths (II)

- **Myth: Different input modes are capable of transmitting comparable content** (Oviatt 1999)
  - Reality: one can not always exchange system control techniques without loosing some content



At the current moment, there is a strong need for formal evaluations of system control techniques. The techniques should be tested in widely varying conditions in order to create a basic understanding of when and in which situations a technique performs well. The creation of more and better guidelines should also help alleviate the problems with badly implemented system control interfaces, and stimulate developments of new techniques which might suit our current and future systems better.



Bibliography

For most references, refer to the “20th century 3DUI Bib”. Below are those references that are not in the main bibliography.

Bullinger, H., Kern, P., Braun, M. Controls. In Handbook of Human Factors and Ergonomics, edited by G. Salvendy, John Wiley & Sons, 1997.

Balakrishnan, R., MacKenzie, I.S. Performance Differences in the Fingers, Wrist, and Forearm in Computer Input Control. In Proceedings of CHI’97, pp. 303-310, 1997.

Billinghurst, M., Savage, J. Adding Intelligence to the Interface. In Proceedings of VRAIS’96, 1996.

Bowman, D. Principles for the Design of Performance-oriented Interaction Techniques. To appear in Stanney, K. (ed.), Handbook of Virtual Environment Technology, Lawrence Erlbaum Associates, 2000.

Brewster, S.A. Using Nonspeech Sounds to Provide Navigation Cues. In ACM Transactions on Computer-Human Interaction, 5(3), pp. 224-259, 1998.

•Kurtenbach, G., Fitzmaurice, G., Baudel, T. & Buxton, W. *The design and evaluation of a GUI paradigm based on tablets, two-hands, and transparency.* Proceedings of CHI '97, pp. 35-42, 1997.

McDonald, J.E., Molander, M.E., Noel, R.W. *Color-coding categorization in menus.* In Proceedings of ACM CHI’88, pp. 101-106, 1988.

•Mine, M., Brooks, F., Sequin, C. Moving objects in space: exploiting proprioception in virtual-environment interaction, in Proceedings of SIGGRAPH'97, ACM Press, pp. 19-26, 1997.

Continuation of resources

•Norman, D.A. *The design of everyday* things. Basic Books, New York, 1988.

•Oviatt, S. *Ten Myths of Multimodal Interaction.* Communications of the ACM, 42(11), pp. 74-81, 1999.

•Tang, S.H., Linton, M.A. *Blending Structured Graphics and Layout.* In Proceedings of UIST'94, pp. 167-174, 1994.

Acknowledgements

The writer would like to thank Dirk Donath, Mark Mine, Holger Regenbrecht, Luis Serra, Andreas Simon, Gerold Wesche, and the tutorial colleagues for their help and comments on structure and contents of the course notes.