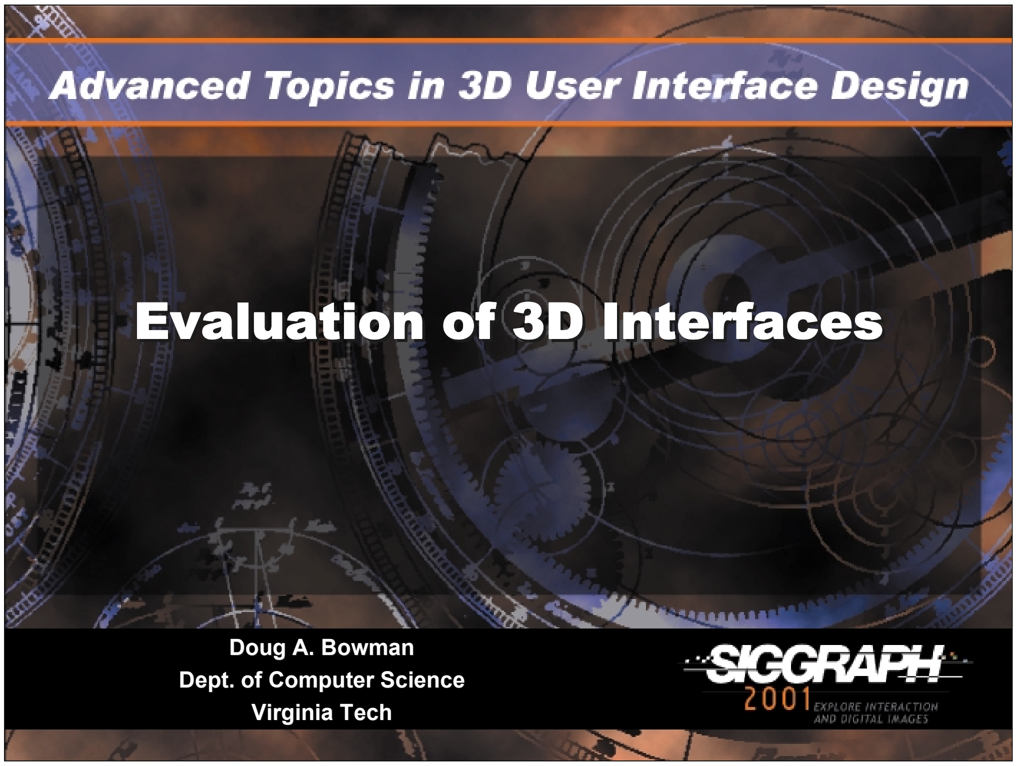


Advanced Topics in 3D User Interface Design

Evaluation of 3D Interfaces

Doug A. Bowman
Dept. of Computer Science
Virginia Tech

SIGGRAPH 2001 EXPLORE INTERACTION AND DIGITAL IMAGES

Evaluation of 3D interfaces

Doug A. Bowman
Dept. of Computer Science (0106)
660 McBryde Hall
Virginia Tech
Blacksburg, VA 24061 USA

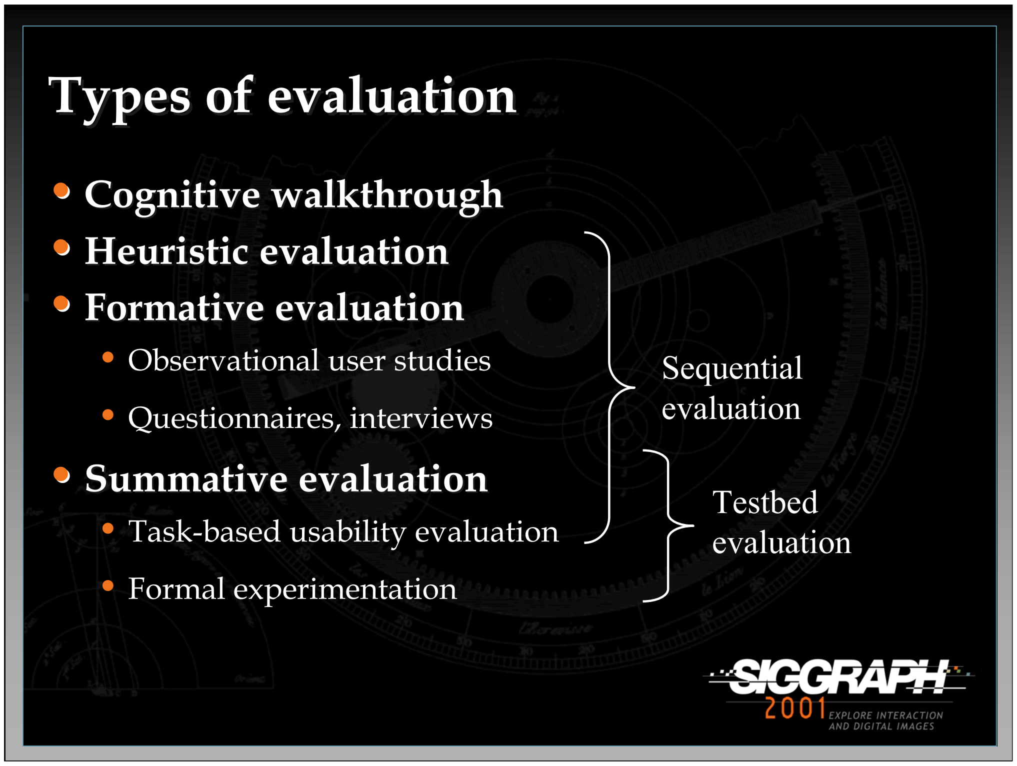
Email: bowman@vt.edu
Web: http://www.cs.vt.edu/~bowman/

In this section, we'll discuss the evaluation of 3D interfaces and interaction techniques. Topics include:

- •evaluation types
- •evaluation issues
- •how 3D UI evaluation differs from the evaluation of traditional interfaces
- •evaluation approaches (testbed and sequential evaluation)
- •metrics for 3D UI evaluation
- •guidelines for 3D UI evaluation

-continued on the next page

We should note that systematic evaluation approaches are fairly new to 3D interface design. Until a few years ago, most researchers performed either cursory user studies or none at all, judging by published research papers. Thus, this is still an ongoing area of research. However, it draws heavily from traditional human-computer interaction (HCI) research. There are likely some readers who are not convinced of the necessity or usefulness of usability evaluation, but it’s clear from experience and from the literature that a designer is not likely to produce a completely usable interface the first time – this is especially true for 3D interfaces, where there are fewer guidelines and examples from which to draw. Thus, assessment is necessary to catch the inevitable usability problems.



Types of evaluation

- Cognitive walkthrough
- Heuristic evaluation
- Formative evaluation
  - Observational user studies
  - Questionnaires, interviews
- Summative evaluation
  - Task-based usability evaluation
  - Formal experimentation

Sequential evaluation
Testbed evaluation
SIGGRAPH 2001 EXPLORE INTERACTION AND DIGITAL IMAGES

Here are some general categories of user interface evaluation that are applicable to 3D UIs.

A cognitive walkthrough is an evaluation done by experts, who step through each of the tasks in a system, asking detailed questions about each step in the task. For example, "Is it clear to the user what can be done here?", or "Can the user translate his intention into an action?" The answers to these questions reveal potential usability problems.

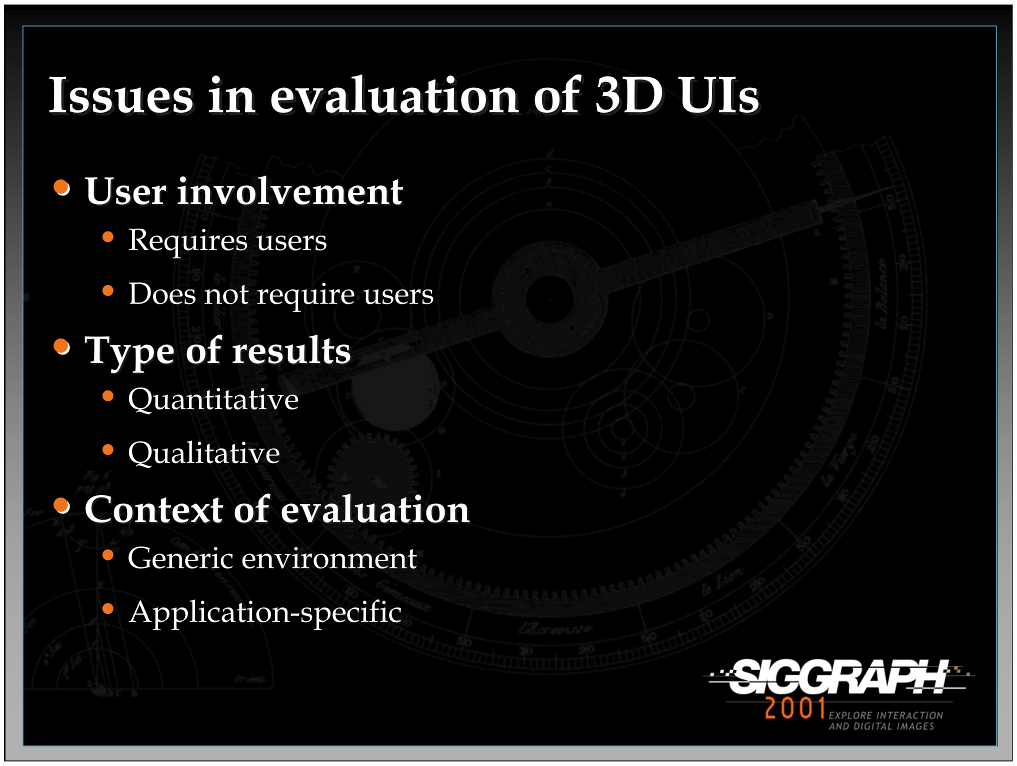
Heuristic evaluation refers to an evaluation by interface experts, using a well-defined set of heuristics or guidelines. Experts examine the interface visually, via a written description, or through actual use, and determine whether or not the interface meets the criteria set forth in the heuristics. For example, the interface might be checked to see if it meets the guideline: "Eliminate extraneous degrees of freedom for a manipulation task."

Formative evaluations are used to refine the design of a widget, an interaction technique, or a UI metaphor. Observational user studies are informal sessions in which users try out the proposed interface. They may be asked to simply explore and play around, or to do some simple tasks. Often users' comments are recorded ("think out loud" or verbal protocol), and the evaluator watches the user to see if there are parts of the interface that are frustrating or difficult. Post-hoc questionnaires and interviews may be used to get more detailed information from users about their experiences with the system.

Summative evaluations compare various techniques in a single experiment. A task-based usability evaluation is more structured. Users are given specific tasks to perform. Often, users are timed as they perform the tasks, and evaluators may keep track of errors made by the user. This information is then used to improve the interface. Formal experiments have a formal design including independent and dependent variables, subjects from a particular subject pool, a strict experimental procedure, etc. The results of formal experiments are usually quantitative, and are analyzed statistically.

We will be talking about two specific evaluation approaches in this section. Sequential evaluation spans a wide range of evaluation types. Testbed evaluation involves summative techniques.

Note: Information in the first 12 slides is drawn from: Bowman, D., Gabbard, J., and Hix, D. Usability Evaluation in Virtual Environments: A Comparison and Integration of Methods. Submitted to the *ACM Transactions on Computer-Human Interaction*, 2001.



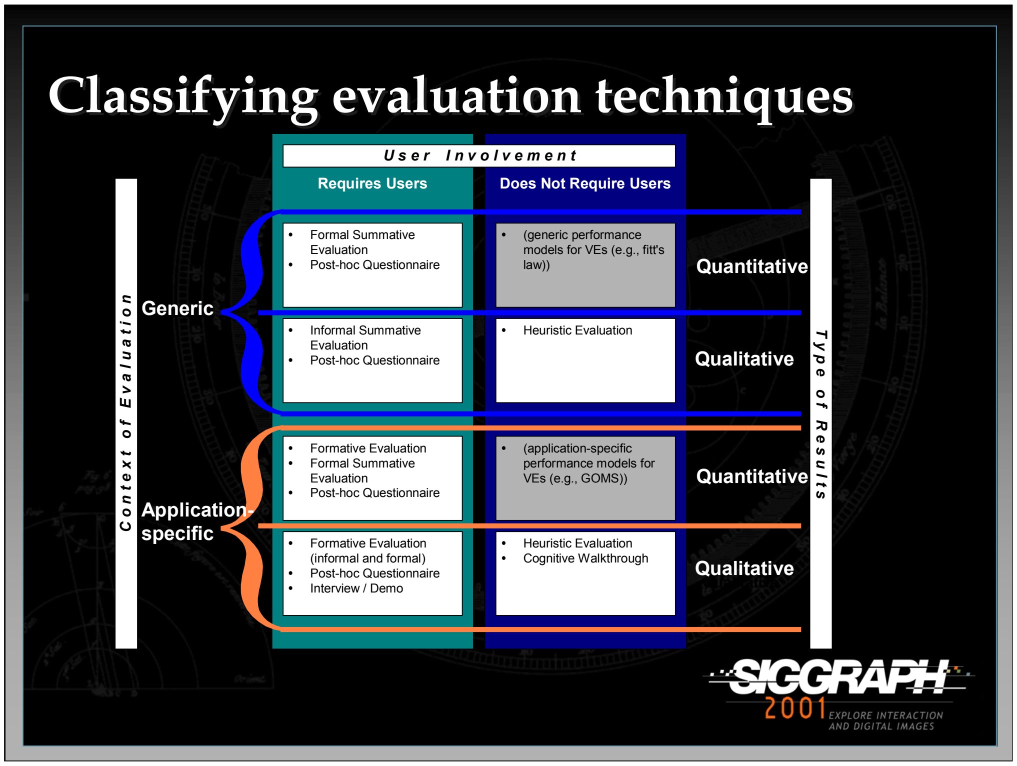
Issues in evaluation of 3D UIs

- User involvement
  - Requires users
  - Does not require users
- Type of results
  - Quantitative
  - Qualitative
- Context of evaluation
  - Generic environment
  - Application-specific

Evaluation methods can be classified according to three major issues:

1. User involvement: Some evaluation approaches use interface experts to make recommendations and suggestions for a system based on UI guidelines or principles. Other approaches utilize subjects drawn from the user population.
2. Type of results: Interface evaluations can produce quantitative (numeric) or qualitative (descriptive) results, or both. Both types can be extremely useful in refining or analyzing usability. Many approaches produce both types of results.
3. Context of evaluation: Some evaluations, especially formal experiments, are done in a generic testing environment, where the results can then be generalized to many applications. Other evaluations are application-specific, meaning that the results are more narrowly focused, but also that they may be more accurate.

Choosing an evaluation method can be seen as making a decision about each of these three issues. The choices you make depend on your specific goals and situation. For example, early in the design process, a qualitative evaluation not requiring users is cheap and fast, and will likely produce significant gains in usability. If I am designing general interaction techniques, a quantitative evaluation with users in a generic environment should produce general results regarding the situations for which the techniques are useful.



Classifying evaluation techniques

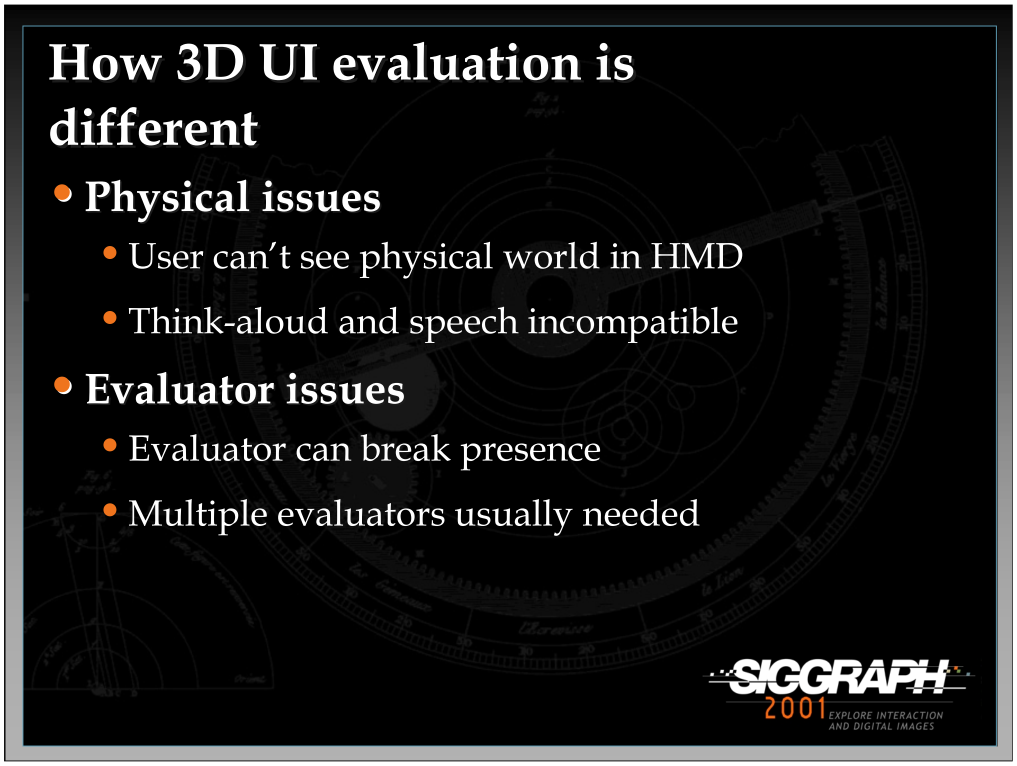
Context of Evaluation	Type of Results	Requires Users	Does Not Require Users
Generic	Quantitative	• Formal Summative Evaluation • Post-hoc Questionnaire	• (generic performance models for VEs (e.g., fitt's law))
	Qualitative	• Informal Summative Evaluation • Post-hoc Questionnaire	• Heuristic Evaluation
Application-specific	Quantitative	• Formative Evaluation • Formal Summative Evaluation • Post-hoc Questionnaire	• (application-specific performance models for VEs (e.g., GOMS))
	Qualitative	• Formative Evaluation (informal and formal) • Post-hoc Questionnaire • Interview / Demo	• Heuristic Evaluation • Cognitive Walkthrough

User Involvement

SIGGRAPH 2001 EXPLORE INTERACTION AND DIGITAL IMAGES

This slide shows various evaluation techniques classified according to the scheme from the previous slide.

The gray boxes represent parts of the design space that have not yet been explored in the context of the evaluation of 3D interfaces. We have suggested some possibilities for filling in these gaps. Both gaps have to do with the application of performance models for 3D interfaces. Such models do not yet exist due to the youth of the field.



How 3D UI evaluation is different

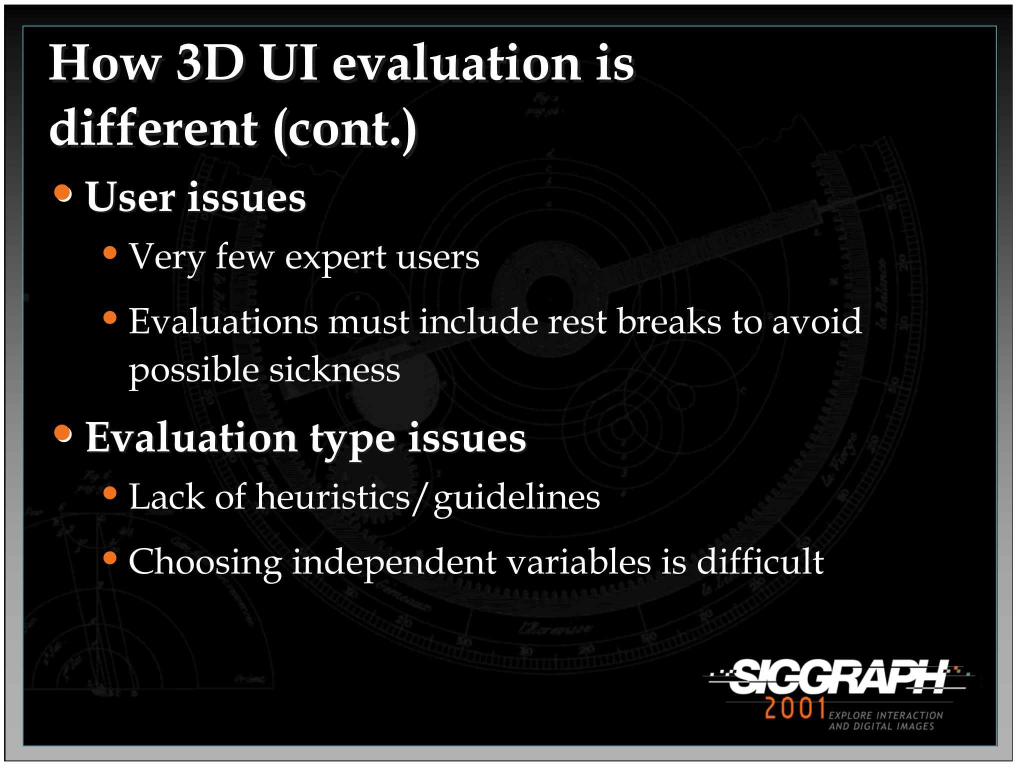
- **Physical issues**
  - User can't see physical world in HMD
  - Think-aloud and speech incompatible
- **Evaluator issues**
  - Evaluator can break presence
  - Multiple evaluators usually needed

SIGGRAPH 2001 EXPLORE INTERACTION AND DIGITAL IMAGES

There are a number of ways in which evaluation of 3D interfaces is different from traditional user interface evaluation.

First, there are physical issues. For example, in an HMD-based VE, the physical world is blocked from the user's view. This means that the evaluator must ensure that the user does not bump into objects or walls, that the cables stay untangled, and so on. Another example involves a common method in traditional evaluation called a "think-aloud protocol". This refers to a situation in which the user talks aloud about what he is thinking/doing in the interface. However, many 3D applications use speech input, which of course is incompatible with this evaluation method unless there is an explicit "push-to-talk" technique. Even in this case, the user could not invoke a command while simultaneously describing his thoughts/actions to the evaluator.

Second, we consider issues related to the evaluator. One of the most important is that an evaluator can break the user's sense of presence by talking to the user, touching the user, making changes to the environment, etc. during an evaluation. If the sense of presence is considered important to the task/application, the evaluator should try to avoid contact with the user during the tests. Another example of an evaluator issue is that multiple evaluators are usually needed. This is because 3D systems are so complex (hardware and software) and because users of 3D UIs have much more freedom and input bandwidth than users of a traditional UI.



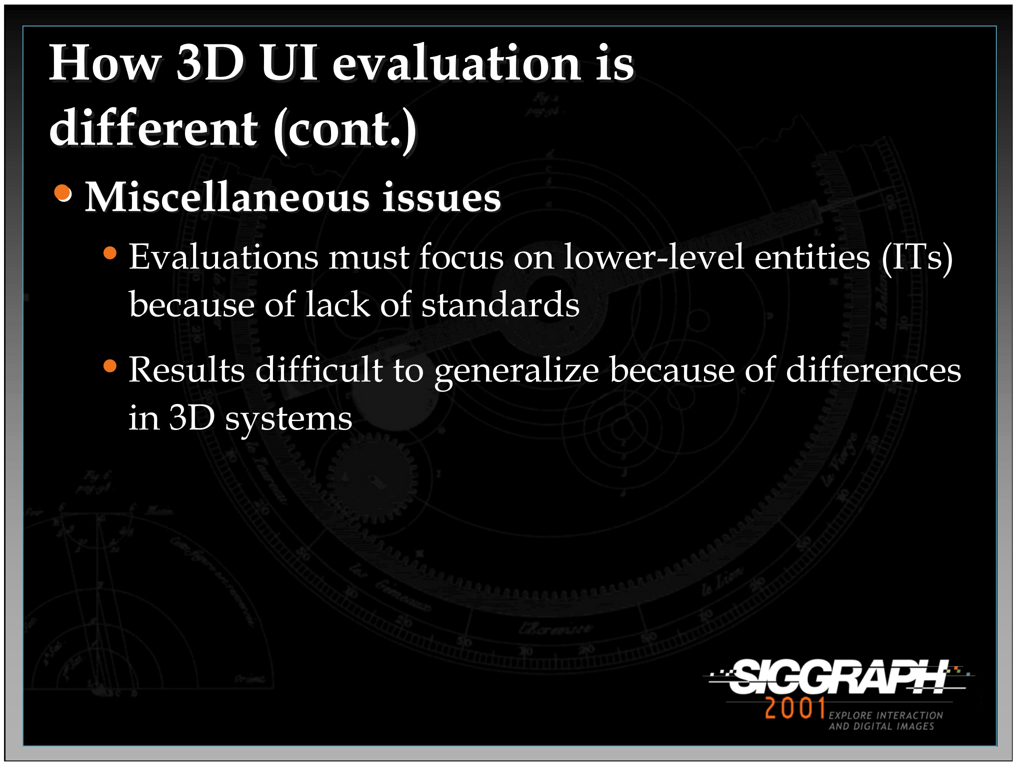
How 3D UI evaluation is different (cont.)

- **User issues**
  - Very few expert users
  - Evaluations must include rest breaks to avoid possible sickness
- **Evaluation type issues**
  - Lack of heuristics/guidelines
  - Choosing independent variables is difficult

SIGGRAPH 2001 EXPLORE INTERACTION AND DIGITAL IMAGES

Third, we look at user issues. One problem is the lack of users who can truly be considered "experts" in 3D application usage. Since the distinction between expert and novice usage is important for interface design, this makes recruiting an appropriate subject pool difficult. Also, 3D systems have problems with simulator sickness, fatigue, etc. that are not found in traditional UIs. This means that the experimental design needs to include provisions like rest breaks and the amount of time spent in the system needs to be monitored.

Fourth, issues related to the type of evaluation performed. Heuristic evaluation can be problematic, because 3D interfaces are so new that there is not a large body of guidelines from which to draw, although this is changing. Also, if you are doing a formal experiment, there are a huge number of factors which might affect performance. For example, in a travel task, the size of the environment, the number of obstacles, the curviness of the path, the latency of the system, and the spatial ability of the user all might affect the time it takes a user to travel from one location to the other. Choosing the right variables to study is therefore a difficult problem. It's also sometimes difficult to realize which factors must be held constant to avoid excessive variability in the experimental results.

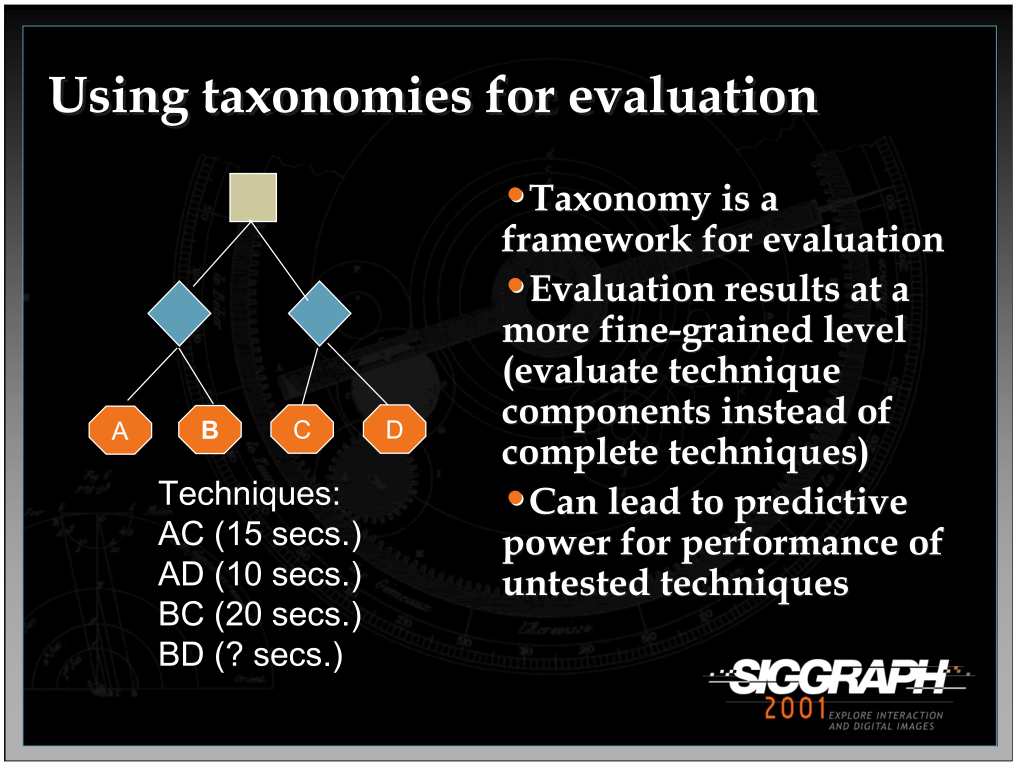


How 3D UI evaluation is different (cont.)

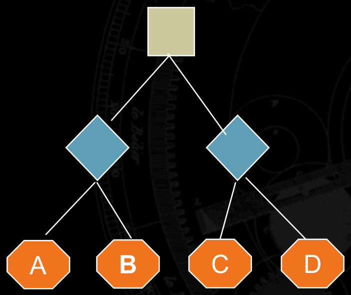
- **Miscellaneous issues**
  - Evaluations must focus on lower-level entities (ITs) because of lack of standards
  - Results difficult to generalize because of differences in 3D systems

SIGGRAPH
2001
EXPLORE INTERACTION AND DIGITAL IMAGES

Finally, there are some miscellaneous issues related to 3D UI evaluation. Most interface evaluation focuses on subtle details of the interface, such as the placement of items within menus, or on the overall metaphor used in the interface. In 3D systems, however, evaluation often focuses on the basic interaction techniques, because we simply don't know yet what ITs should typically be used. Also, it's hard to generalize the results of an experiment or evaluation, because usually the evaluation is done with a particular type of hardware, a single type of environment, etc., but in real usage, a wide variety of different devices, software systems, and environments will likely be encountered.



Using taxonomies for evaluation



Techniques:
AC (15 secs.)
AD (10 secs.)
BC (20 secs.)
BD (? secs.)

- Taxonomy is a framework for evaluation
- Evaluation results at a more fine-grained level (evaluate technique components instead of complete techniques)
- Can lead to predictive power for performance of untested techniques

One way to impose some order on the huge design space for 3D interaction techniques is to create a classification or taxonomy of techniques. The type of taxonomy shown here is based on a hierarchic decomposition of a task. The *task* (tan box) is divided into *subtasks* that must be completed. For example, the task of object selection might be viewed as two subtasks: indicating an object, then giving a command to select that object. For each subtask, the taxonomy can list *technique components* that might be used to complete that subtask. In our example, pointing to the object is a component for the first subtask, and pressing a button is a component addressing the second subtask. A complete interaction technique is composed of a component for each lowest-level subtask. Therefore, in the figure, there are 2x2 = 4 possible interaction techniques that can be created.

Taxonomies can be used as a framework for evaluation. This not only means that you can design evaluations based on the structure given by a taxonomy, but that you can predict performance based on this structure as well.

-continued on the next page

Here's an example of predictive power:

Task: changing the color of an object

Subtasks: selecting object, selecting color

Technique components for object selection: pointing (A), choosing from list (B)

Technique components for color selection: RGB sliders (C), 3D RGB cube (D)

Technique 1: AC (measured to take an average of 15 seconds)

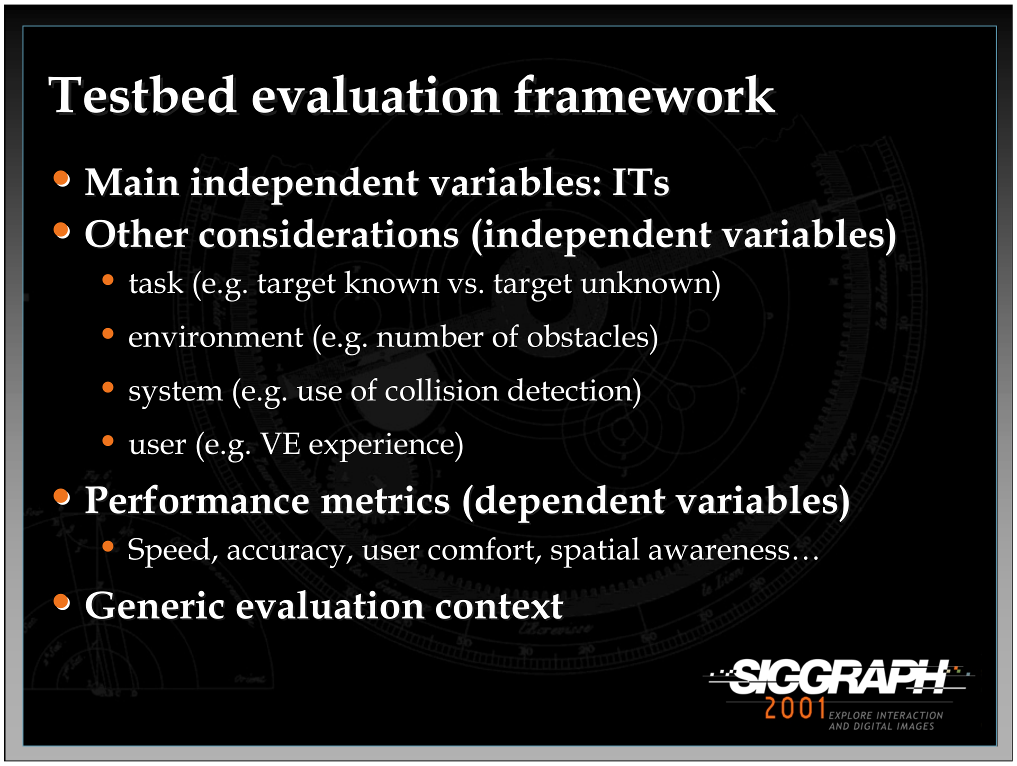
Technique 2: AD (10 seconds)

Technique 3: BC (20 seconds)

Technique 4: BD (not evaluated)

We can infer that component D takes 5 seconds shorter than C, and that B takes 5 seconds longer than A.

So BD can be calculated as AD+5 = 15 seconds, or BC-5 = 15 seconds.



Testbed evaluation framework

- Main independent variables: ITs
- Other considerations (independent variables)
  - task (e.g. target known vs. target unknown)
  - environment (e.g. number of obstacles)
  - system (e.g. use of collision detection)
  - user (e.g. VE experience)
- Performance metrics (dependent variables)
  - Speed, accuracy, user comfort, spatial awareness…
- Generic evaluation context

An evaluation testbed is a generalized environment in which many smaller experiments or one large experiment can be run, covering as much of the design space as you can. Like other formal experiments, you're evaluating interaction techniques (or components), but you also include other independent variables that could have an effect on performance. These include characteristics of the task, environment, system, and user.

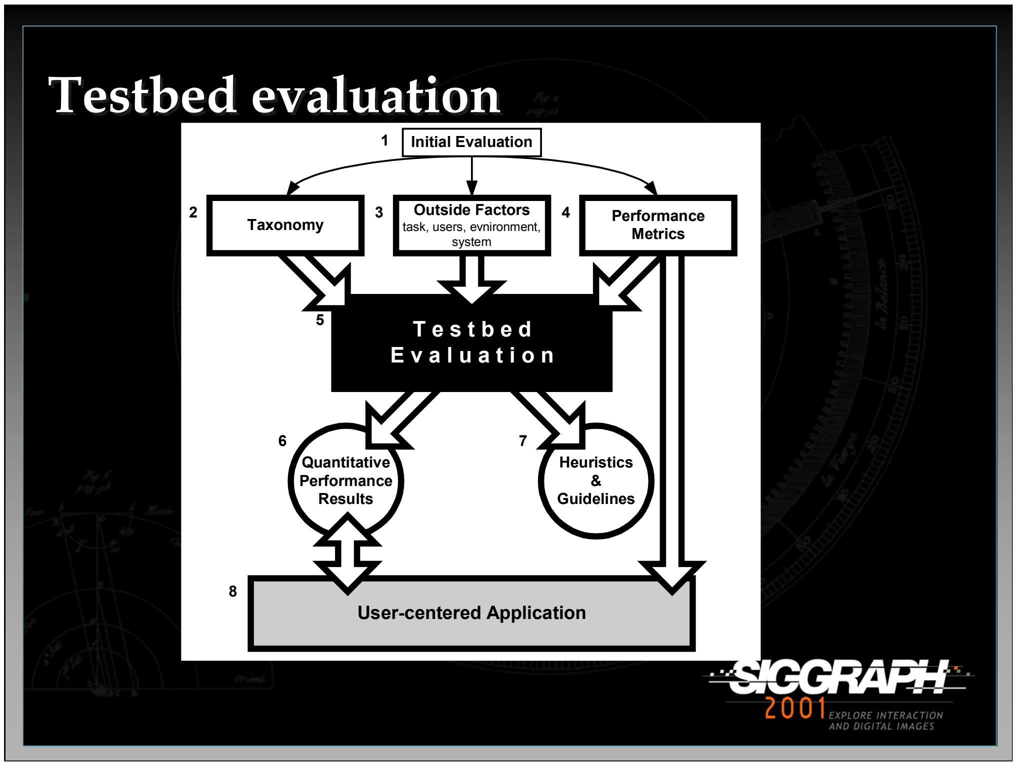
You also measure multiple dependent variables in such experiments to try to get a wide range of performance data. Here we use performance in the broader sense, not just meaning quantitative metrics. The more metrics you use, the more applications can use the results of the experiment by listing their requirements in terms of the metrics, then searching the results for technique(s) that meet those requirements.

Doug Bowman performed such evaluations in his doctoral dissertation, available online at: http://www.cs.vt.edu/~bowman/thesis/. A summary version of these experiments is in this paper:

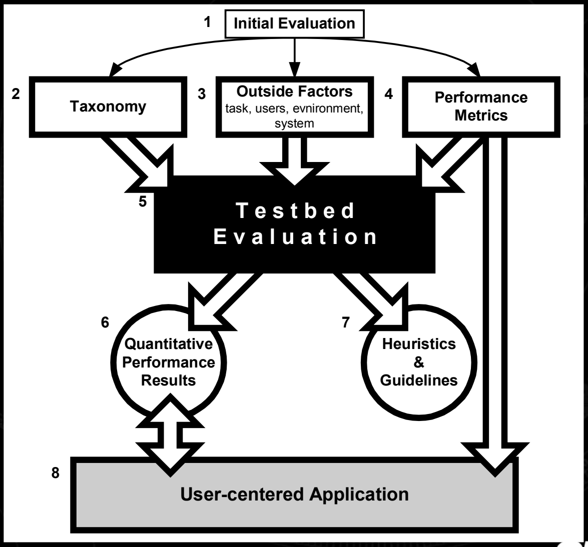
Bowman, Johnson, & Hodges, Testbed Evaluation of VE Interaction Techniques, Proceedings of ACM VRST '99

Also see: Poupyrev, Weghorst, Billinghurst, and Ichikawa, A Framework and Testbed for Studying Manipulation Techniques, Proceedings of ACM VRST '97.

In terms of our three issues, testbed evaluation involves users, produces quantitative (and perhaps qualitative) results, and is done in a generic context.



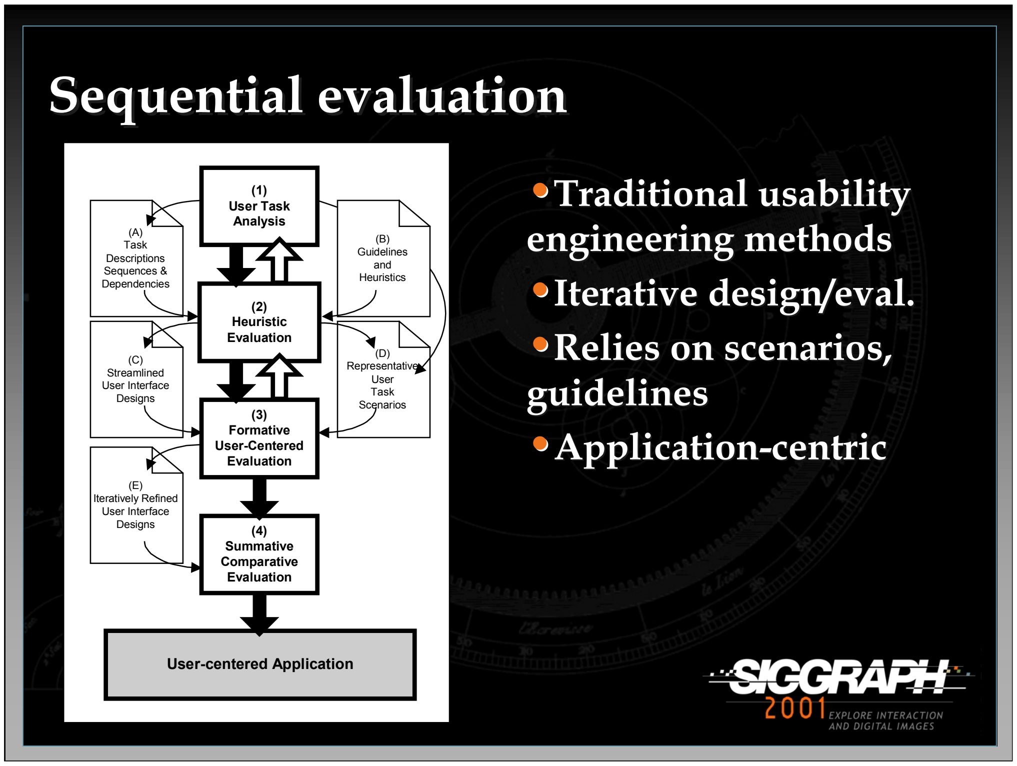
Testbed evaluation


1 Initial Evaluation
2 Taxonomy
3 Outside Factors
task, users, evnironment, system
4 Performance Metrics
5 Testbed Evaluation
6 Quantitative Performance Results
7 Heuristics & Guidelines
8 User-centered Application

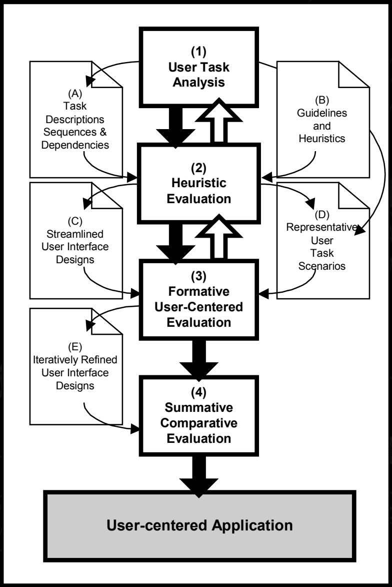
SIGGRAPH 2001 EXPLORE INTERACTION AND DIGITAL IMAGES

This figure shows the process used in testbed evaluation. Before designing a testbed, one must understand thoroughly the task(s) involved and the space of interaction techniques for those tasks. This understanding can come from experience, but it's more likely to come from some initial (usually informal) evaluations. This can lead to a taxonomy for a task, a set of other factors that are hypothesized to affect performance on that task, and a set of metrics (discussed later).

These things are then used to design and implement a testbed experiment or set of experiments. The results of running the testbed are the actual quantitative results, plus a set of guidelines for the usage of the tested techniques. The results can be used many times to design usable applications, based on the performance requirements of the application specified in terms of the performance metrics.



Sequential evaluation


(1) User Task Analysis
(A) Task Descriptions Sequences & Dependencies
(B) Guidelines and Heuristics
(2) Heuristic Evaluation
(C) Streamlined User Interface Designs
(D) Representative User Task Scenarios
(3) Formative User-Centered Evaluation
(E) Iteratively Refined User Interface Designs
(4) Summative Comparative Evaluation
User-centered Application

- Traditional usability engineering methods
- Iterative design/eval.
- Relies on scenarios, guidelines
- Application-centric

A different approach is called sequential evaluation. See the paper:

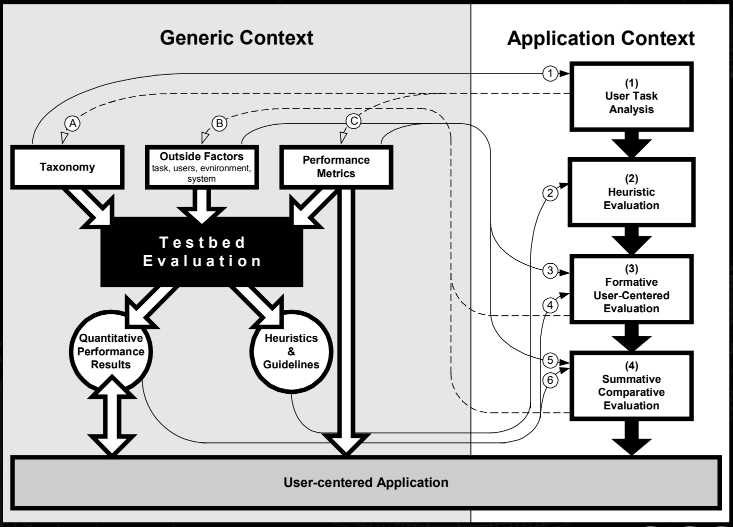
Gabbard, J. L., Hix, D, and Swan, E. J. (1999). User Centered Design and Evaluation of Virtual Environments , *IEEE Computer Graphics and Applications*, *19*(6), 51-59.

As the name implies, this is actually a set of evaluation techniques run in sequence. The techniques include user task analysis, heuristic evaluation, formative evaluation, and summative evaluation. As the figure shows, the first three steps can also involve iteration. Note that just as in testbed evaluation, the goal is a user-centered application.

In terms of our three issues, sequential evaluation uses both experts and users, produces both qualitative and quantitative results, and is application-centric.

Note that neither of these evaluation approaches is limited to being used for the evaluation of 3D UIs. However, they do recognize that applications with 3D UIs require a more rigorous evaluation process than traditional 2D UIs, which can often be based solely on UI guidelines.

Combined approach



We have proposed to combine and integrate the testbed and sequential evaluation approaches. Recall that the major difference between the two is the context of evaluation: a generic context for testbed evaluation and an application context for sequential evaluation. This allows the results of the testbed process to be used as inputs for the sequential process, and vice-versa.

[letters and numbers refer to the arrows in the figure]

Using testbed evaluation as an input to sequential evaluation:

> User task analysis, a critical part of the sequential evaluation approach, requires an understanding of tasks users must perform and possible ITs that could be used to accomplish those tasks. Taxonomic structures from the testbed approach provide both of these (1). Taxonomies provide a standard way to organize and decompose a task, and they contain a design space from which many ITs can be built.
>
> The general guidelines produced by testbed evaluation can serve as input for heuristic evaluation in the sequential evaluation approach (2). In fact, this addresses a potential problem with using heuristic evaluation for VEs: a lack of heuristics. Since guidelines from the testbed approach are based on experimental evidence, heuristic evaluation using these guidelines should produce a more usable initial design to be fed to the formative evaluation process.

-continued on the next page

The set of factors other than ITs that could influence performance (outside factors) are an important component of the testbed evaluation process, since they are candidates for independent variables in testbed experiments. For example, one could test whether the number of obstacles in an environment affects the speed of traversing a path in that environment. These same factors can play a role in shaping formative and summative evaluation components of the sequential evaluation approach (3 and 5). The evaluator can use these factors to more carefully plan task scenarios that assess the range of potential interactions a user could have with the VE. In a similar way, sets of performance metrics defined for testbed evaluation are useful in formative and summative evaluation. These metrics can be checked to ensure that the evaluator observes all variables that contribute to a usable interface.

Finally, quantitative performance results obtained from testbed experiments can play a role in the sequential evaluation process. In formative evaluation (4), an evaluator is trying to produce one or more usable ITs that can later be compared. If testbed results are available for the task in question, incorporation of these ITs into a VE can begin at a much more refined level based on performance results. In the same way, testbed results can help narrow the set of ITs in summative evaluation (6). The relative performance of two ITs may already be known through testbed evaluation, or a particular IT may be known to perform badly in the situation presented by a particular VE application. In any case, these results should be considered before beginning either type of evaluation.

Using sequential evaluation as an input to testbed evaluation:

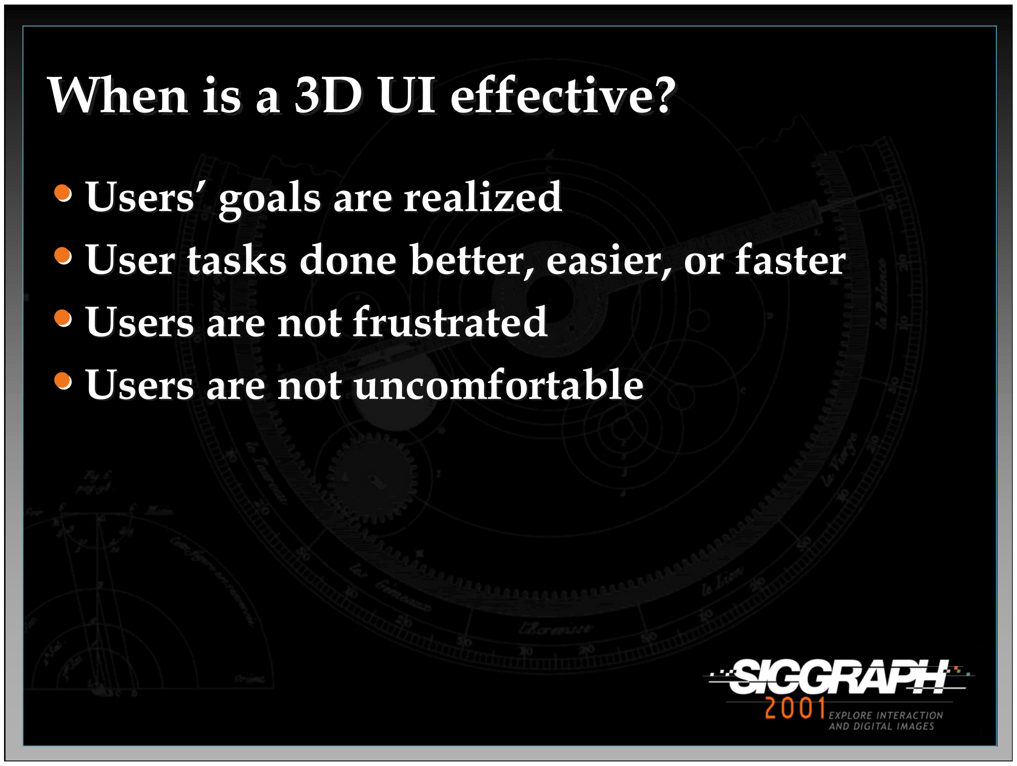
In all three of these cases, the experiences of analyzing a real-world application help to refine the generic model used for testbed evaluation.

One way this can occur involves the process of user task analysis (A). Task analysis takes place in the context of a particular application, and can also be refined as the sequential evaluation approach is iterated. This can result in a quite detailed understanding of user tasks, intentions, and mental models for a specific VE. This understanding is exactly what is needed to create good taxonomies of ITs for a particular task, since taxonomies in the testbed approach are based on task decomposition. If taxonomies more closely fit the user's model of a particular task, when this taxonomy is used as a framework for evaluation the results obtained should be a better predictor of user performance in real systems.

-continued on the next page

Subsequent to the process of user task analysis, usability goals and associated metrics can be determined. It is important for a user to complete tasks efficiently, correctly, without frustration, and in comfort. These characteristics match some of the possible performance metrics given by the testbed approach. However, it is possible that in the process of user task analysis and subsequent setting of usability goals, evaluators will find that a VE has a requirement whose fulfillment cannot be determined using any of the listed performance measures (C). The requirement may suggest a new metric to be added to the list and included in future testbed experiments.

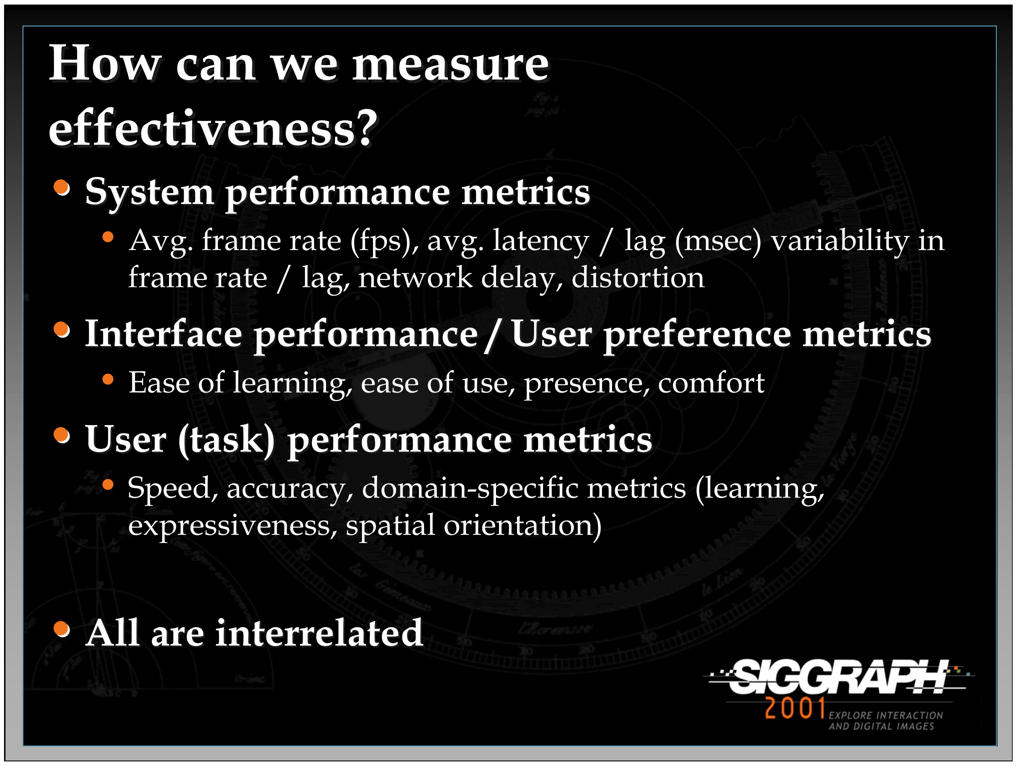
It is difficult in the testbed approach to come up with complete lists of the outside factors that could affect performance. This is often done based on intuition alone. However, experiences of evaluators performing formative and summative evaluations can add to and refine these lists (B). Evaluators may notice that a user performing a particular task is greatly affected by some characteristic of the environment. This would suggest that this characteristic should be studied in a future testbed experiment to determine the extent of its effects more generally. If that variable has already been studied in a general experiment, it may be possible to give more weight to this factor in analysis of the results.



When is a 3D UI effective?

- Users' goals are realized
- User tasks done better, easier, or faster
- Users are not frustrated
- Users are not uncomfortable

Now we turn to metrics. That is, how do we measure the characteristics of a 3D UI when evaluating it? I will focus on the general metric of *effectiveness*. A 3D UI is effective when the user can reach her goals, when the important tasks can be done better, easier, or faster than with another system, and when users are not frustrated or uncomfortable. Note that all of these have to do with the *user*. As we will see later, typical performance metrics like speed of computation are really not important in and of themselves. After all, the point of the application is to serve the needs of the user, so speed of computation is only important insofar as it affects the user's experience or tasks.



How can we measure effectiveness?

- **System performance metrics**
  - Avg. frame rate (fps), avg. latency / lag (msec) variability in frame rate / lag, network delay, distortion
- **Interface performance / User preference metrics**
  - Ease of learning, ease of use, presence, comfort
- **User (task) performance metrics**
  - Speed, accuracy, domain-specific metrics (learning, expressiveness, spatial orientation)
- **All are interrelated**

We will talk about three different types of metrics, all of which are interrelated.

System performance refers to traditional CS performance metrics, such as frame rate.

> As mentioned earlier, the only reason we're interested in system performance is that it has an effect on interface performance and user performance. For example, the frame rate probably needs to be at "real-time" levels before a user will feel present. Also, in a collaborative setting, task performance will likely be negatively affected if there is too much network delay.

Interface performance (the user's preference or perception of the interface) refers to traditional HCI metrics like ease of learning.

> These metrics are mostly subjective, and are measured via qualitative instruments, although they can sometimes be quantified. For VE systems in particular, presence and user comfort can be important metrics that are not usually considered in traditional UI evaluation.

-continued on the next page

High levels of the user preference metrics generally lead to *usability*. A usable application is one whose interface does not pose any significant barriers to task completion. Often HCI experts will speak of a “transparent” interface – a UI that simply disappears until it feels to the user as if he is working directly on the problem rather than indirectly through an interface. User interfaces should be intuitive, provide good affordances (indications of their use and how they are to be used), provide good feedback, not be obtrusive, and so on. An application cannot be effective unless it is usable (and this is precisely the problem with some more advanced VE applications – they provide functionality for the user to do a task, but a lack of usability keeps them from being used).

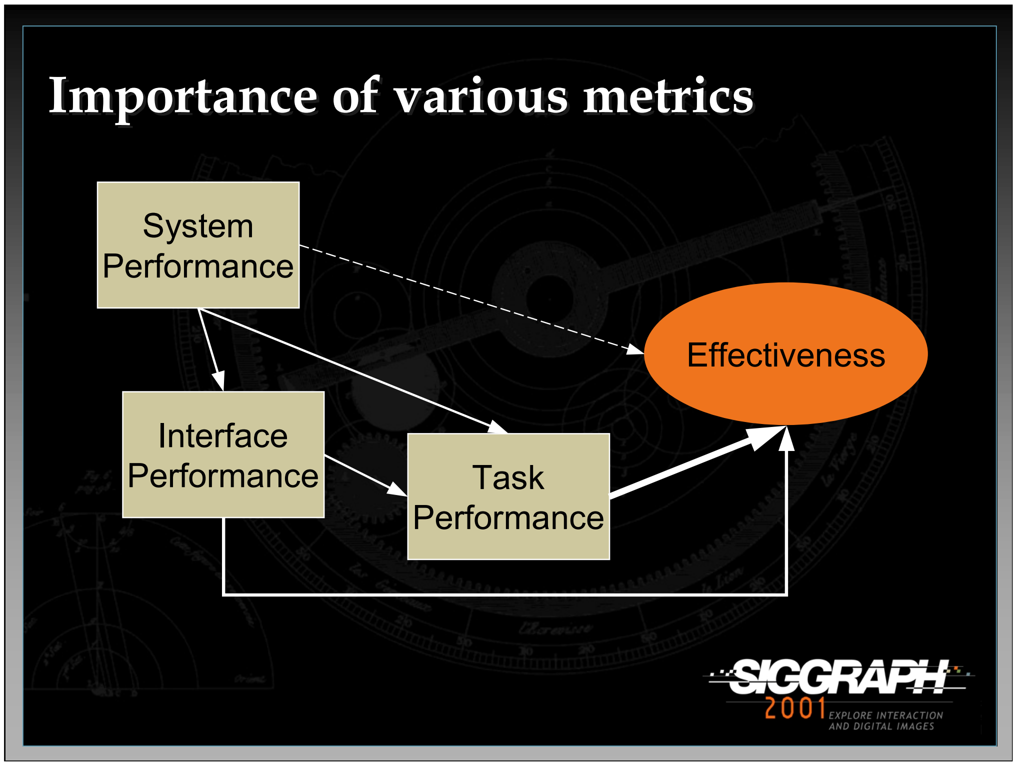
Presence is a crucial, but not very well-understood metric for VE systems. It is the feeling of being there – existing in the virtual world rather than in the physical world. How can we measure presence? A simple measure simply asks users to rate their feeling of “being there” on a 1-100 scale. Questionnaires generally ask many questions, all designed to get at different aspects of presence. Psychophysical measures are used in controlled experiments where stimuli are manipulated and then correlated to user’s ratings of presence (for example, how does the rating change when the environment is presented in mono vs. stereo modes?). There are also some more objective measures. Some are physiological (how the body responds to the VE). Others might look at users’ reactions to events in the VE (e.g. does the user duck when he’s about to hit a virtual beam). Tests of memory for the environment and the objects within it might give an indirect measurement of the level of presence. Finally, if we know a task for which presence is required, we can measure users’ performance on that task and infer the level of presence. Witmer and Singer (*Presence 7*(3), 1998) developed a formal presence questionnaire (PQ), along with an immersive tendencies questionnaire (ITQ). They did a lot of work to validate the design of these questionnaires, and have used them extensively to provide a standard measurement for presence. However, there is still a lot of controversy about their approach.

-continued on the next page

The other novel user preference metric for 3D systems is user comfort. This includes several different things. The most notable and well-studied is so-called "simulator sickness" (because it was first noted in things like flight simulator). This is similar to motion sickness, and may result from mismatches in sensory information (e.g. your eyes tell your brain that you are moving, but your vestibular system tells your brain that you are not moving). There is also work on the physical aftereffects of being exposed to 3D systems. For example, if a VE mis-registers the virtual hand and the real hand (they're not at the same physical location), the user may have trouble doing precise manipulation in the real world after exposure to the virtual world. More seriously, things like driving or walking may be impaired after extremely long exposures (1 hour or more). Finally, there are simple strains on arms/hands/eyes from the use of 3D hardware. User comfort is also usually measured subjectively, using rating scales or questionnaires. The most famous questionnaire is the simulator sickness questionnaire (SSQ) developed by Robert Kennedy (*International Journal of Aviation Psychology, 3*(3), 1993). Kay Stanney has attempted some objective measures in her study of aftereffects – for example by measuring the accuracy of a manipulation task in the real world after exposure to a virtual world.

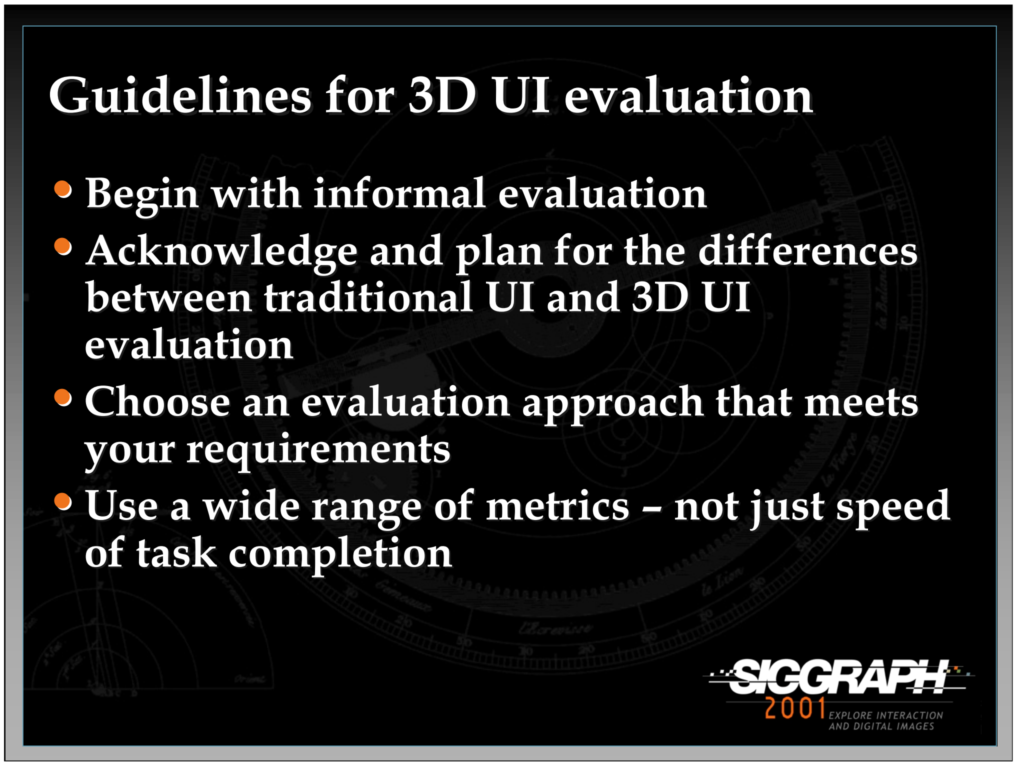
<u>User (task) performance</u> refers to the quality of performance of specific tasks in the 3D application, such as the time to complete a task.

The problem with measuring speed and accuracy is that there is an implicit relationship between them: I can go faster but be less accurate, or I can increase my accuracy by decreasing my speed. It is assumed that for every task there is some curve representing this speed/accuracy tradeoff, and users must decide where on the curve they want to be (even if they don't do this consciously). So, if I simply tell my subjects to do a task as quickly and precisely as possible, they will probably end up all over the curve, giving me data with a high level of variability. Therefore, it is very important that you instruct users in a very specific way if you want them to be at one end of the curve or the other. Another way to manage the tradeoff is to tell users to do the task as quickly as possible one time, as accurately as possible the second time, and to balance speed and accuracy the third time. This gives you information about the tradeoff curve for the particular task you're looking at.



Importance of various metrics

This is an imprecise diagram that shows how I relate the three types of metrics. System performance directly affects interface performance and task performance. It only indirectly affects overall effectiveness of the 3D application (it’s possible for the system to perform at low levels but still be effective). Interface performance and usability affect task performance directly, and also affects overall effectiveness directly, since an unusable 3D application will not be tolerated by users. Task performance is the most important factor in determining overall effectiveness, since the goal of the 3D UI is to allow users to do their tasks better, easier, and faster.



Guidelines for 3D UI evaluation

- Begin with informal evaluation
- Acknowledge and plan for the differences between traditional UI and 3D UI evaluation
- Choose an evaluation approach that meets your requirements
- Use a wide range of metrics – not just speed of task completion

SIGGRAPH 2001 EXPLORE INTERACTION AND DIGITAL IMAGES

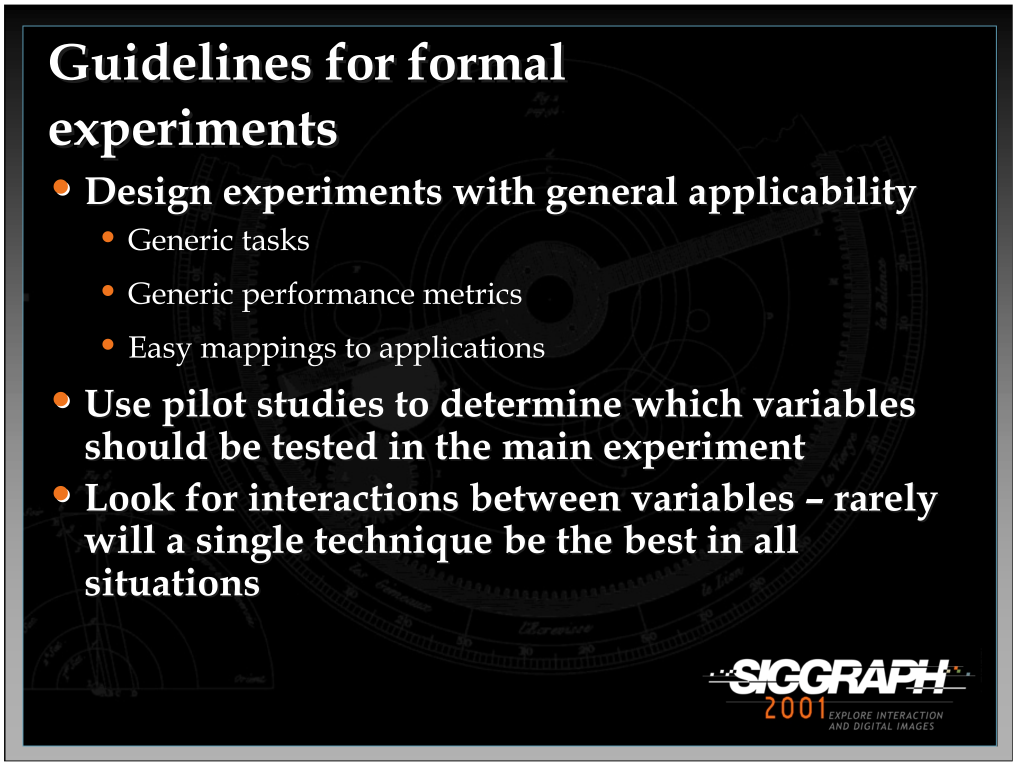
Here are a set of guidelines to be used in any type of evaluation of 3D UIs.

Informal evaluation is very important, both in the process of developing an application and in doing basic interaction research. In the context of an application, informal evaluation can quickly narrow the design space and point out major flaws in the design. In basic research, informal evaluation helps you understand the task and the techniques on an intuitive level before moving on to more formal classifications and experiments.

Remember the unique characteristics of 3D UI evaluation from the beginning of this talk when planning your studies.

There is no optimal evaluation technique. Study the classification presented in this talk and choose a technique that fits your situation.

Remember that speed and accuracy do not equal usability. Also remember to look at learning, comfort, presence, etc.



Guidelines for formal experiments

- Design experiments with general applicability
  - Generic tasks
  - Generic performance metrics
  - Easy mappings to applications
- Use pilot studies to determine which variables should be tested in the main experiment
- Look for interactions between variables – rarely will a single technique be the best in all situations

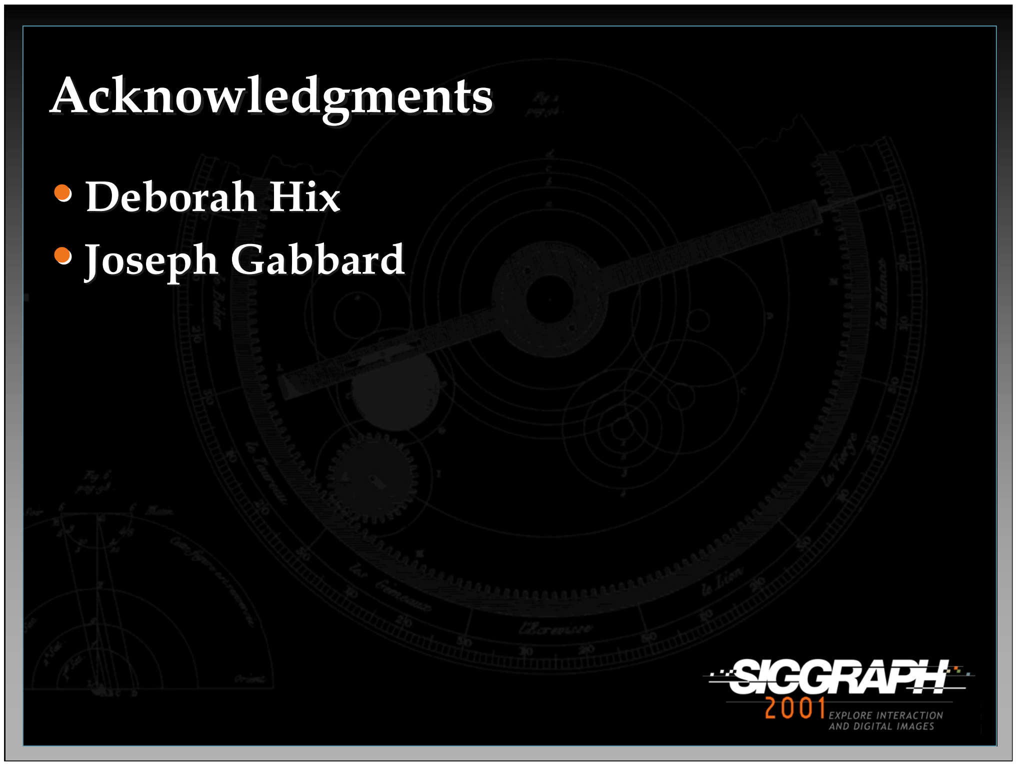
SIGGRAPH 2001 EXPLORE INTERACTION AND DIGITAL IMAGES

These guidelines are for formal experiments in particular – mostly of interest to researchers in the field.

If you're going to do formal experiments, you want the results to be as general as possible. Thus, you have to think hard about how to design tasks which are generic, performance measures that real applications can relate to, and a method for applications to easily re-use the results.

In doing formal experiments, especially testbed evaluations, you often have too many variables to actually test without an infinite supply of time and subjects. Small pilot studies can show trends that may allow you to remove certain variables, because they do not appear to affect the task you're doing.

In almost all of the experiments we've done, the most interesting results have been interactions. That is, it's rarely the case that technique A is always better than technique B. Rather, technique A works well when the environment has characteristic X, and technique B works well when the environment has characteristic Y. Statistical analysis should reveal these interactions between variables.



Acknowledgments

- Deborah Hix
- Joseph Gabbard

SIGGRAPH 2001 EXPLORE INTERACTION AND DIGITAL IMAGES

I worked closely with Joe Gabbard and Debby Hix (both of Virginia Tech) in developing the list of distinctive characteristics of 3D UI evaluation, the classification scheme for evaluation techniques, and the combined testbed/sequential evaluation approach.