

Evaluation and metrics: Measuring the effectiveness of virtual environments

Doug Bowman

This lecture is on the topic of evaluation of VEs and ways to measure the effectiveness of VEs.

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

(C) 2006 Doug Bowman, Virginia Tech

2

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.

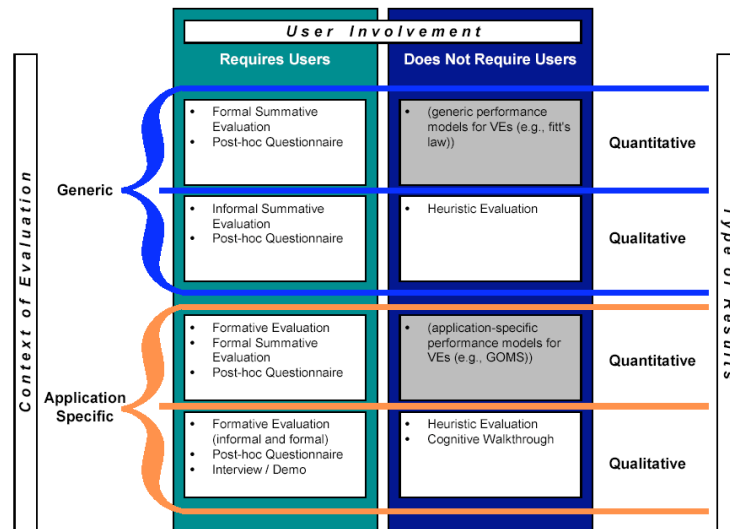
Heuristic evaluation refers to an evaluation by interface experts, using a well-defined set of heuristics or guidelines. Experts examine the interface visually, via a written description, or through actual use, and determine whether or not the interface meets the criteria set forth in the heuristics. 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

Classifying evaluation techniques



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 VE evaluation is different

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

There are a number of ways in which evaluation of virtual environments 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 VE applications use speech input, which of course is incompatible with this evaluation method.

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 VE systems are so complex (hardware and software) and because VE users have much more freedom and input bandwidth than users of a traditional UI.

How VE 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

Third, we look at user issues. One problem is the lack of users who can truly be considered “experts” in VE usage. Since the distinction between expert and novice usage is important for interface design, this makes recruiting an appropriate subject pool difficult. Also, VE 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 VE needs to be monitored.

Fourth, issues related to the type of evaluation performed. Heuristic evaluation can be problematic, because VE 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.

How VE 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 VE systems

Finally, there are some miscellaneous issues related to VE 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 VE 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.

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.

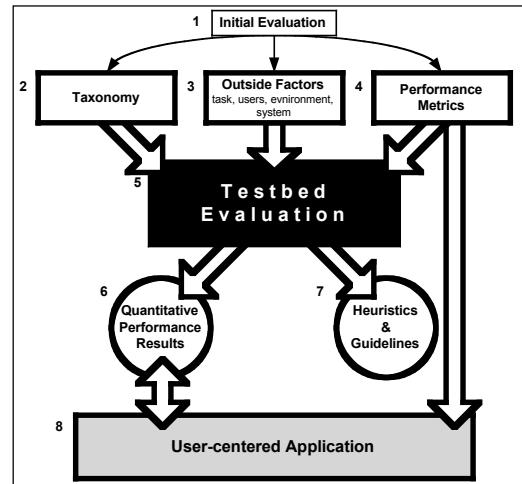
You also measure multiple dependent variables in such experiments to try to get a wide range of performance data. Here we use performance in the broader sense, not just meaning quantitative metrics. The more metrics you use, the more applications can use the results of the experiment by listing their 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, Billingham, 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



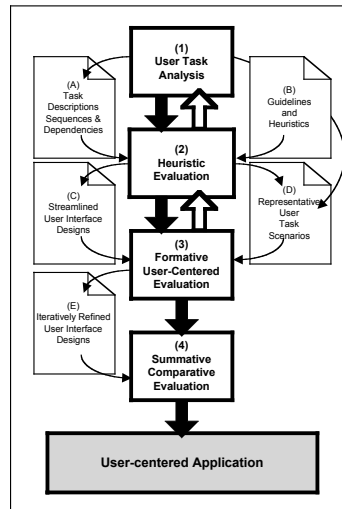
(C) 2006 Doug Bowman, Virginia Tech

8

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



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

(C) 2006 Doug Bowman, Virginia Tech

9

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.

When is a VE 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 VE when evaluating it? I will focus on the general metric of *effectiveness*. A VE 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 CS performance metrics like speed of computation are really not important in and of themselves. After all, the point of the VE 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
- Interface performance / User preference
- User (task) performance

- All are interrelated

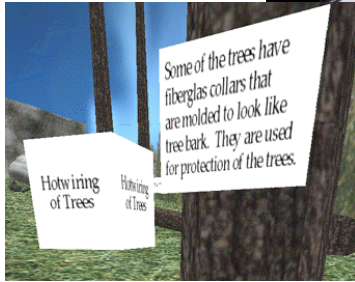
We will talk about three different types of metrics, all of which are interrelated.

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

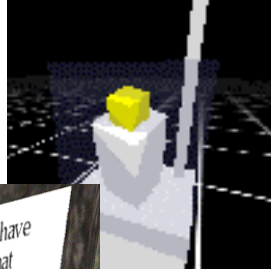
Interface performance (the user's preference or perception of the interface) refers to traditional HCI metrics like ease of learning.

User (task) performance refers to the quality of performance of specific tasks in the VE, such as the time to complete a task.

Effectiveness case studies



(C) 2006 Doug Bowman, Virginia Tech



- Watson experiment: how system performance affects task performance
- Slater experiments: how presence is affected
- Design education: task effectiveness

12

The three types of metrics will be illustrated by three case studies.

System performance metrics

- Avg. frame rate (fps)
- Avg. latency / lag (msec)
- Variability in frame rate / lag
- Network delay
- Distortion

Here are some possible system performance metrics. Note that they are fairly general in nature, that they are measurable, and that they apply to any type of VE system or application.

System performance

- Only important for its effects on user performance / preference
 - frame rate affects presence
 - net delay affects collaboration
- Necessary, but not sufficient

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.

Case studies - Watson

- How does system performance affect task performance?
- Vary avg. frame rate, variability in frame rate
- Measure perf. on closed-loop, open-loop task

e.g. B. Watson et al, Effects of variation in system responsiveness on user performance in virtual environments. Human Factors, 40(3), 403-414.

Ben Watson performed some experiments (see the reference on the slide) where he varied some system performance values and measured their effect on task performance. For example, one experiment varied the frame rate and also the variability in the frame rate over time. He also looked at different types of tasks. Closed-loop tasks are ones in which users make incremental movements and use visual, proprioceptive, and other types of feedback to adjust their movements during the task. A real-world example is threading a needle. Open-loop tasks do not use feedback during the task – they simply make some initial calculations and then proceed with the movement. A real-world example is catching a ball thrown at a high speed. He found that these tasks were affected in different ways by the various system performance values.



User preference metrics

- Ease of use / learning
- Presence
- User comfort
- Usually subjective (measured in questionnaires, interviews)

Here are some examples of interface performance (user preference) metrics. These 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.

User preference in the interface

- UI goals
 - ease of use
 - ease of learning
 - affordances
 - unobtrusiveness
 - etc.
- Achieving these goals leads to *usability*
- Crucial for effective applications

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).

Case studies - Slater

- questionnaires
- assumes that presence is required for some applications
- study effect of:
 - collision detection
 - physical walking
 - virtual body
 - shadows
 - movement

e.g. M. Slater et al, Taking Steps: The influence of a walking metaphor on presence in virtual reality. ACM TOCHI, 2(3), 201-219.

The case study for this section is the work of Mel Slater, who is arguably the current leading authority on the sense of presence (one reference is listed, but you can find many more). He has also developed (less formal) presence questionnaires, and especially looked at the effect of manipulating various system and interface variables on the reported sense of presence. He has a bunch of papers titled something like, "The influence of X on presence in VR".

User comfort

- Simulator sickness
- Aftereffects of VE exposure
- Arm/hand strain
- Eye strain

The other novel user preference metric for VE 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 VE 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 VE hardware.

Measuring user comfort

- Rating scales
- Questionnaires
 - Kennedy - SSQ
- Objective measures
 - Stanney - measuring aftereffects

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. 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.

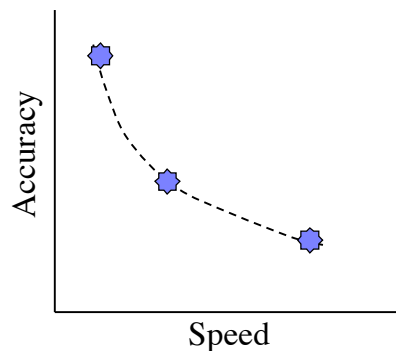
Task performance metrics

- Speed / efficiency
- Accuracy
- Domain-specific metrics
 - Education: learning
 - Training: spatial awareness
 - Design: expressiveness

The last category of metrics are task performance metrics. These include general measures like speed and accuracy, and domain-specific measures such as learning and spatial awareness.

Speed-accuracy tradeoff

- Subjects will make a decision
- Must explicitly look at particular points on the curve
- *Manage* tradeoff



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.

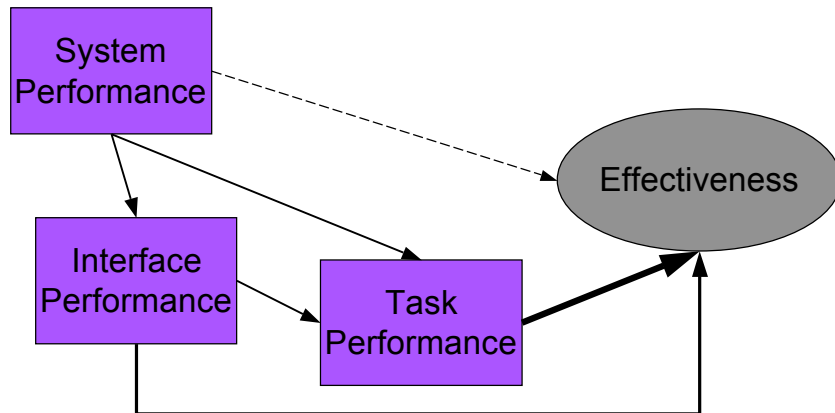
Case studies: learning

- Measure effectiveness by learning vs. control group
- Metric: standard test
- Issue: time on task not the same for all groups

e.g. D. Bowman et al. The educational value of an information-rich virtual environment. *Presence: Teleoperators and Virtual Environments*, 8(3), June 1999, 317-331.

The case study for this section is an experiment that I did as part of my dissertation, where I was interested in learning as a task performance metric. The VE contained information about a particular topic (environmental design). The study had three groups: a control group who heard a traditional lecture on the material, a environment-only group who heard the lecture and also got to navigate the VE (without the extra information), and a study group who heard the lecture, navigated the VE, and saw the information within the VE. The measure was a quiz given in class. The study group performed better on this quiz. However, the study was not perfect, since the study group actually spent more time on the learning task than the control group. It wasn't possible to give them the same amount of learning time, since this was done in the context of a real class (the information could affect their grade). Such experimental design issues are very important when measuring task performance.

Aspects of performance



(C) 2006 Doug Bowman, Virginia Tech

24

This is an imprecise diagram that shows how I relate the three types of metrics. System performance directly affects interface performance and task performance, as we saw in the Watson studies. It only indirectly affects overall effectiveness of the VE (it's possible for a 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 VE will not be tolerated by users. Task performance is the most important factor in determining overall effectiveness, since the goal of the VE 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

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

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

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.