Touchstone: Exploratory Design of Experiments

Wendy E. Mackay[1,2], Caroline Appert[2,1], Michel Beaudouin-Lafon[2,1], Olivier Chapuis[2,1], Yangzhou Du[2,1], Jean-Daniel Fekete[1,2], Yves Guiard[3]

[1]INRIA F-91405 Orsay, France

[2]LRI, Univ. Paris-Sud & CNRS F-91405 Orsay, France

[3]LMP, CNRS & Univ. Mediterranee F-13288 Marseille, France

{mackay, appert, mbl, chapuis, du, fekete}@lri.fr, yves.guiard@univmed.fr

ABSTRACT

Touchstone is an open-source experiment design platform designed to help establish a solid research foundation for HCI in the area of novel interaction techniques. Touchstone includes a *design* platform for exploring alternative designs of controlled laboratory experiments, a *run* platform for running subjects and a limited *analysis* platform for advice and access to on-line statistics packages. Designed for HCI researchers and their students, Touchstone facilitates the process of creating new experiments, as well as replicating and extending experiments in the research literature. We tested Touchstone by designing two controlled experiments. One illustrates how to create a new experiment from scratch. The other replicates and extends a previous study of multiscale pointing interaction techniques: OrthoZoom was fastest, followed by bi-manual Pan & Zoom; SDAZ and traditional Pan & Zoom were consistently slower.

Author Keywords: Benchmarking, Experimental Design, Interaction Techniques, Touchstone

ACM Classification Keywords

H.5.2 [User Interfaces]: Benchmarking, Evaluation/ methodology, D.2.2 [Software Engineering]: Design Tools & Technique –User interfaces, H.1.2 [Models & Principles]: User/Machine Systems – Human factors

INTRODUCTION

In well-established empirical fields such as biology and experimental psychology, with over a century of experience conducting experiments, graduate students are taught that experiments build upon each other and that no individual experiment provides the ultimate answer to the question being studied. Researchers form theories, operationalize hypotheses based on those theories and perform controlled experiments to determine cause and correlation. Individual studies are published in the literature; other researchers repeat those experiments to confirm or refute them, and then extend them. No one experiment is ever definitive, although statistically significant results provide strong support for the theory in question. This type of research is what Kuhn [15] refers to as the puzzle-solving aspect of "normal science".

Gaines [10] describes the evolution of technology, beginning with breakthroughs and the replication of ideas, then empirical models and theories, and finally automation and maturity. Human-Computer Interaction (HCI) is a young research area and is, for the most part, still at the breakthrough and replication stages, despite Card et al.'s [5] ground-breaking theoretical work in the 1980's. Great value is still placed on novelty and innovation, as when developing novel interaction techniques, and research is not always based on either theoretical or empirical foundations. MacKenzie et al. [17] argue that, despite the abundance of published evaluations in the HCI and human factors literature, the methodologies are ad hoc and the experimental procedures are inconsistent from one study to the next, which "greatly diminishes our ability to understand the results or to undertake comparisons between studies".

Replicating and extending experiments is difficult in practice, even for experienced researchers. In order to compare a new interaction technique to existing ones, experimenters must not only program their own techniques, but also re-implement or obtain running versions of other existing techniques based on the information published in the literature. This is time-consuming and does not guarantee generalizability, since tiny differences in the details of the implementation may have major effects on performance. As a result, the most common practice is to compare a new interaction technique to a single 'standard' technique on a single task and ignore other possible contenders.

The goal of Touchstone[1] is to help establish a solid research foundation for HCI in the area of novel interaction techniques, by providing a repository and a set of tools to design, run and analyze controlled experiments. We first present related work and then

CHI 2007, April 28–May 3, 2007, San Jose, California, USA.

[1] Historically, a "touchstone" is a black stone on which soft metals leave a visible trace. A goldsmith would compare the trace left by a new gold coin against a standard stripe of known quality. *Wikipedia* also defines a Touchstone as "Any physical or intellectual measure by which the validity or otherwise of a concept can be tested".

describe the key components of Touchstone. We then illustrate the use of the design and run platforms with two experiments. We describe the results of the first experiment and conclude with a discussion of Touchstone and directions for future research.

RELATED WORK

Several on-line 'experiment generators', e.g., WeXtor and EDGAR[2] are designed to help students learn how to design specified types of experiments. While they offer a start, none provide researchers with the 'what-if' style of experiment design exploration that we sought. As researchers, we would like to see what happens if, for example, we add extra values to a particular factor: What is the effect on how many subjects are required for a properly counter-balanced experiment? What is the effect of replicating trials on the overall length of the experiment for any individual subject?

Some statistical analysis packages, such as JMP (SAS Institute) include experiment design modules. We have also found websites[3] that provide code for implementing particular designs. Unfortunately, these assume that the experimenter has already made all the relevant design decisions and is simply plugging in the factors. In addition, modules from statistics packages are very comprehensive, covering a huge variety of possible experimental designs that go far beyond the scope of most HCI studies and are often confusing for non-specialists.

Finally, we have found many on-line courses and textbooks on experimental design[4]. Some are excellent, but they are rarely presented with HCI experiments as examples. Understandably, they tend to specialize according to a particular application domain and differences in terminology and assumptions about what constitutes a typical experiment may create a significant barrier. For example, the HCI literature uses the terms *within-subjects* and *repeated measures*, which are common in psychology, but rarely *correlated samples*, from engineering statistics. Yet all mean the same thing. We also find confusion about terms within the HCI literature. For example, a *repeated* measures design refers to a single measure being repeated once for each subject; *replicated* measures refer to multiple presentations of the same trial to the same subject. Since few standard introductory psychology courses involve repeated measures experiments that also replicate trials per subject, many researchers find it difficult to obtain appropriate advice on the design and analysis of such experiments.

With respect to the actual running of experiments, despite the large number of user interface toolkits, the only systems we know that share Touchstone's goals are the Generalized Fitts' Law Model Builder [20] and the Shakespeare platform [13]. Both are specifically targeted at Fitts experiments and could be used in the context of ISO standard 9241 [21] on input devices, but neither supports interactive, exploratory experiment design. We need more general tools that help a wider range of HCI researchers to design, run and analyze HCI experiments.

TOUCHSTONE ARCHITECTURE

We developed Touchstone as an open-source tool for creating controlled, counterbalanced experiments, especially (but not solely) for comparing interaction techniques. The design is modular (Figure 1), with three main components that can be used independently, although they are designed to work together.

The *design platform* provides a simple web-based interface to guide experimenters through all phases of designing a controlled experiment, including blocking, counterbalancing, and estimating the timing. It produces an XML *script* that describes the experiment, as well as text descriptions of different aspects of the experiment design and sample data logs. The current version of the design platform covers the most common experimental designs used in HCI. If needed, users can edit the script file or use their own counterbalancing strategies.

The *run platform* consists of the *experiment launcher* and the *Touchstone Development Environment (TDE)*. The experiment launcher is a Java-based application framework that loads *experiment components* (blocks, factors, etc.) from a repository and then runs the experiment, presenting trials to the user according to the experimental protocol defined by the above XML script and collecting data into *data logs*. The *Touchstone Development Environment* is used to implement new experiment components and register them with the design and run platforms. For example, the current set of experiment components implements pointing and

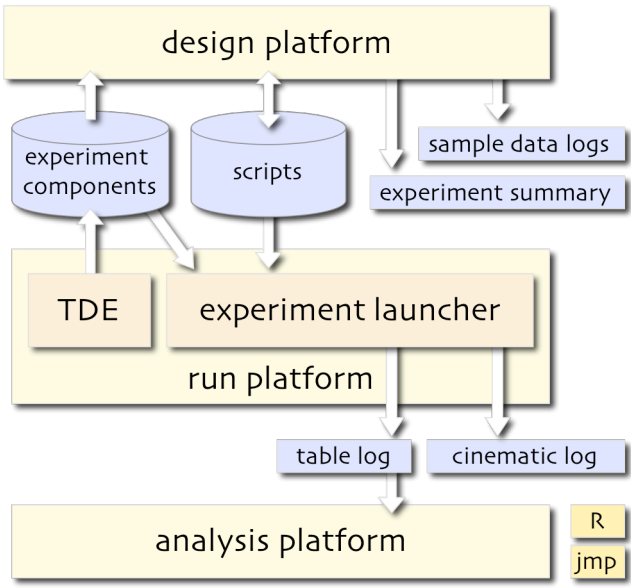

Figure 1. Touchstone architecture: *design*, *run* and *analysis* platforms

[2] Wextor: *http://psychwextor.unizh.ch/wextor/*
EDGAR: *http://www.uea.ac.uk/nrp/jic/edgar*

[3] *http://home.nc.rr.com/schabenb/Designs.html*
http://statpages.org/#WhichAnalysis

[4] For example: *http://faculty.vassar.edu/lowry/webtext.html*

navigation techniques and the building blocks to create Fitts experiments [9]. New components can easily be added to handle other types of experiments, e.g., how task context affects the efficacy of interaction techniques [16] or comparisons of non-traditional interaction techniques such as those involving computer vision [8].

The *analysis platform* helps the user analyze the data logs produced by the run platform and provides input to well-known statistical packages. The current version provides only limited advice about which statistical techniques are appropriate for the design chosen by the experimenter. We refer users to two statistics packages, R and JMP, which can read the table logs directly.

Design Platform

Touchstone's design platform supports *exploratory experiment design* by enabling the experimenter to examine the consequences of alternative design decisions through a 'what if' style of interaction. For example: What if I add a new value to this factor: how many additional subjects will I need? What if I treat this factor as secondary and only present it to some subjects rather than to all subjects? How many trial replications can I afford, without making the experiment too long? A help facility, with pointers to other on-line resources, provides advice about the advantages and disadvantages of each choice and the issues that should be considered in the subsequent analysis of the chosen design.

The challenge in creating the design platform was to find the appropriate balance between simplicity, i.e., targeting the optimal subset of possible experiment designs with a minimalist, efficient interface for creating them, and power, i.e., enabling researchers to conduct the necessary range of experiments with minimum additional programming or design adjustments. Table 1 shows our distillation of the experiment design process into six key steps, each with one or more design decisions (Table 1).

Step	Decisions
1	*Identify experiment*
2	*Specify factors* How many and which factors and values? Which factors are primary?
3	*Specify blocks* Which block type (complete, mixed or pure)? How many replications?
4	*Specify timing* Which experiment components to include? What time estimates for events, trials, pauses? Which end-criteria for trials, blocks, practice?
5	*Specify ordering* How to order trials and blocks?
6	*Specify measures* Which measures to include in which log file?

Table 1. Summary of Touchstone design decisions.

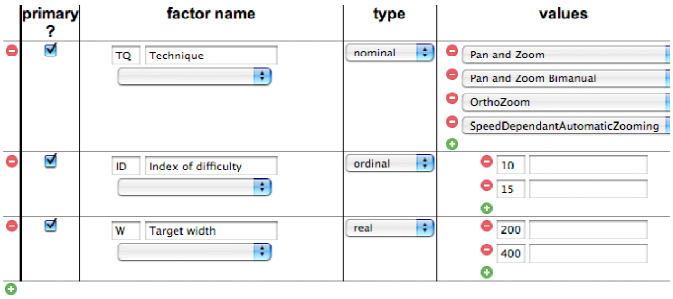

Figure 2. Specifying factors and values

The Touchstone interface presents the six design steps as tabs to encourage iterative design. The experimenter can move among these tabs at any time and changes within a tab propagate to other tabs as necessary. We now illustrate the design process using the experiment we ran to evaluate the platform. It compares four multiscale navigation techniques (unimanual and bimanual Pan & Zoom, OrthoZoom, SDAZ) with two indices of difficulty (*ID*=10, 15) and two target widths (*W*=200, 400 pixels).

Step one: The experimenter provides basic information about the experiment: name, filename for summary files, authors and a brief description of the experiment.

Step two: The experimenter specifies the desired factors and their values (Figure 2). As a convenience, the design platform can list the factors defined by the run platform (see TDE below), but experimenters can also specify their own. In this example, the experimenter adds the *technique* factor and specifies its values. Each value has a short code for the log files (PZ = unimanual Pan & Zoom, PZB = bimanual Pan & Zoom, OZ = OrthoZoom, SDAZ = Speed-Dependent Automatic Zooming) with an optional long name. The experimenter adds the *ID* factor and specifies its values (10, 15). She then adds the *W* factor and specifies its values (200, 400).

The experimenter must now decide which of these factors are primary and which are secondary. Primary factors are presented *within subjects*, i.e. every subject receives all values of each primary factor, whereas secondary factors are presented *between subjects*, i.e. only one value of a factor is presented to a particular subject. If the experiment has at least one primary and one secondary factor, the design is *mixed*. Touchstone's help facility describes the different design types (within-subjects, between-subjects and mixed), the advantages and disadvantages of each and how the choice affects the recommended minimum number of subjects required and the type of analyses to perform.

Step three: Touchstone feeds back the results of step two, specifying the experiment design, here a 4x2x2 within-subjects design, and generating the list of unique trial types (the experimental conditions) with the suggested minimum number of subjects (Figure 3). Touchstone lists all possible ways of blocking factors within subjects, from a complete-block design, in which a single block contains all the conditions for that subject, to a pure-block design, in which there is one block per condition. In our example,

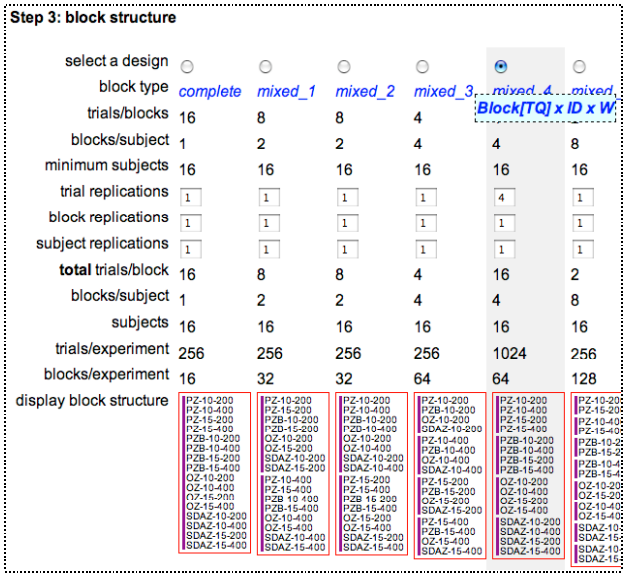

Figure 3. Subset of possible block structures.

Block 1	*Block 2*	*Block 3*	*Block 4*
PZ-10-200	PZB-10-200	OZ-10-200	SDAZ-10-200
PZ-10-400	PZB-10-400	OZ-10-400	SDAZ-10-400
PZ-15-200	PZB-15-200	OZ-15-200	SDAZ-15-200
PZ-15-400	PZB-15-400	OZ-15-400	SDAZ-15-400

Figure 4. Detail of block structure.

we block by *Technique* to ensure that all trials for a single technique are presented in sequence to the subjects. This grouping typically reduces negative skill transfer in within-subject designs. The corresponding design is listed as "Block[Technique] x ID x Width". Since it is a within-subjects design, it has a single subject group with four blocks, corresponding to the generic, i.e. not yet counter-balanced run for each subject (Figure 4).

This step also allows the experimenter to define the number of replications of trials, blocks and subjects, again with advice on the consequences of different decisions. For example, replicating trials for an individual subject decreases the overall variability of the results and thus increases the likelihood of obtaining a significant result (if there is indeed an effect). On the other hand, it increases the running time of the experiment and requires a modified statistical analysis. In our example, we chose to replicate each trial four times per block.

Step four: The experimenter estimates the running time for each subject, including pre- and post-experiment events. For example, the experimenter may decide to give a background questionnaire and a practice session before the experiment and a satisfaction questionnaire and a debriefing session after the experiment. The experimenter also estimates the average length of experimental trials, the inter-trial and inter-block intervals, and specifies the end criteria for ending trials and blocks. Together, this information provides a rough estimate of how long it will take each subject to go through the experiment. If the experiment, including pre- and post-experiment events,

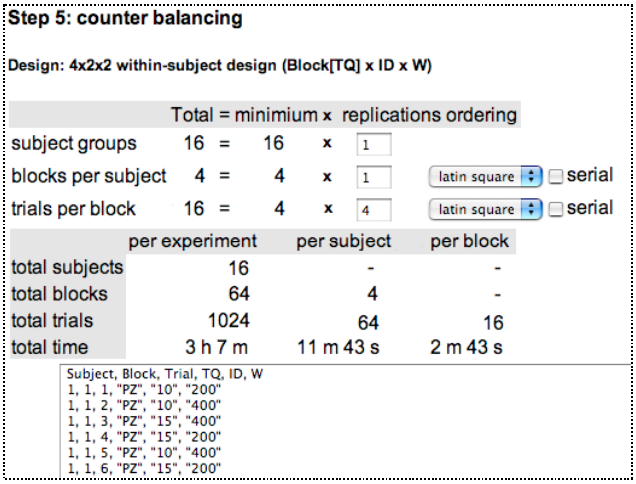

Figure 5. Counter balancing

lasts more than an hour, the experimenter is encouraged to reconsider some earlier design decisions, such as the number of replications, the number of factors or values, or the choice of primary and secondary factors.

Step five: The experimenter decides the counterbalancing strategy for the order of trials and blocks within and across subjects (Figure 5). Counterbalancing helps assure that possible order effects, such as practice or fatigue, are evenly distributed throughout the experiment. The most common choice is a *Latin square* which ensures that each subject receives each condition in a different order and that the orders appear exactly once. Our algorithm also ensures that pairs of trials are counterbalanced across subjects. If the experiment has a very small number of conditions, the experimenter can specify a *complete design* that presents each subject with all possible orders of trials. With very large numbers of conditions, usually when a factor has many possible values, the experimenter can present trials in *random order*. The experimenter can also specify that replicated trials are presented serially. This is a common choice for Fitts experiments that focus on motor tasks, such as reciprocal pointing.

Based on these choices, Touchstone generates the complete set of counterbalanced trials required for each subject and displays it as a comma-separated list that can be easily loaded into analysis software such as Excel, R or JMP. This allows the experimenter to test Touchstone's counter-balancing strategy, e.g., to assess the distribution of certain trials across particular blocks. The experimenter may also decide to apply a different counterbalancing strategy and paste the result back into the Touchstone XML file, for use by the run platform. Making the results of the experimenter's design decisions transparent assures that even experimenters with different experimental design philosophies or training can take advantage of Touchstone's exploratory design facility.

Step six: The experimenter specifies the measures (dependent variables) and the type of data log to record them in. As with factors, the design platform can list the

measures defined in the run platform, but experimenters can also specify their own.

Touchstone provides two types of logs: a *table log* and a *cinematic log*. The table log is designed to be analyzed by a statistics package such as JMP or R. Each line is independent and lists the subject, block and trial numbers, the condition (value of each factors) and the measures chosen. The cinematic log is designed to 'replay' the experiment and enables a more fine-grained analysis of each trial. Touchstone produces samples of both log files, which the experimenter can compare to the actual data produced by the run platform. In our example, we specify the movement time, hit/miss and distance-to-target measures to be included in the table log[5] and the x/y position of the cursor and current scale of the view to be included in the cinematic log.

Note that, although we presented this process linearly, the platform is designed to be iterative, making it easy to revisit and change earlier decisions. For example, if step 4 shows that the average running time of the experiment is about 90 minutes, we can return to step 2 and make *W* (target width) into a secondary factor, shifting from a within-subjects to a mixed-subjects design and cutting the number of experimental trials per subject in half. (This would, however, increase the number of subject groups.)

The resulting XML experiment script (Figure 6) appears in the *Result* tab and can be interpreted directly by the run platform to run the experiment. It can also be saved to a local file or into the Touchstone repository, enabling an experimenter to reuse or modify existing experiments. The current version of Touchstone provides several sample experiments and we hope that future users will make their experiment designs available in this format.

The design platform can generate, from the XML script, a text summary of the experiment, using the format of a

```
1 <interblock class="Message({New block})"
                    criterion="Key(Space)">
2  <block values="TQ=SDAZ" criterionTrial=
       "Dwell(1000) | (TimeOut(180000)=>{Too Long})" >
3     <intertrial class= "Message({Touch the target
                          as quickly as possible!})" …>
4          <trial values="ID=15, W=400"
                    class="PointingBlock" />
5          <trial values="ID=10, W=200"
                    class="PointingBlock" />
6          <trial …/>
7     </intertrial>
8  </block>
9  <block …> ... </block>
10 </interblock>
```

Figure 6. Excerpt from experiment script.

[5] Fitts experiments typically manipulate three variables: target *Width*, target *Distance* and index of difficulty (*ID*). The Fitts components of the run platform are designed so that when any two of these are specified as factors (here, *ID* and *Width*) the third one (here, *Distance*) is generated as a measure.

standard procedure section that includes experiment design type, number of subjects, a complete list of the experimental conditions, the blocking and counter-balancing strategies, as well as an ordered list of trials per subject and sample logs. These summary files capture the design rationale for each experiment, which not only helps experimenters remember why they made certain decisions in earlier experiments, but also provides a teaching aid, facilitating design trade-off discussions between senior researchers and students. Touchstone's help facility includes explanations of basic design concepts and trade-offs, links to relevant on-line courses, a glossary of technical terms and explanations of naming conventions and counterbalancing strategy.

Run Platform

The *run platform* is a 10,000-line Java program that runs experiments created with the design platform. The details of the experiment, including the precise conditions for presenting each trial, are defined in the XML script generated by the design platform (or possibly edited by the experimenter). Figure 6 shows an excerpt of the script generated by the example in the previous section.

The run platform includes an *experiment launcher* and the *Touchstone development environment (TDE)*. The experiment launcher reads the XML script, loads the *experiment components* mentioned in the script and runs the experiment. Touchstone comes with a set of preexisting components available in an on-line repository. New components can be created using the TDE.

Experiment Launcher

The experiment launcher is a Java-based graphical application that presents the trials to the subjects and collects data measures. When launched, it displays a window with three tabs: *Run*, *Summary* and *Input*.

The experimenter begins by entering the subject id and the name of the script file in the *Run* tab (Figure 7) and clicks the Run button. The experiment launcher fetches the *run* corresponding to this subject from the XML script (Figure 6). A run is organized into one or more *blocks*, each with a specified sequence of *trials*. The condition for each trial is specified by the "values" attributes (lines 2 and 4). Values specified at the block level (line 2) are valid for the whole block. *Interblocks* and *intertrials* separate the successive blocks and trials that appear within their scope (lines 1-10 for interblock, 3-7 for intertrial). Usually, they display instructions or provide a

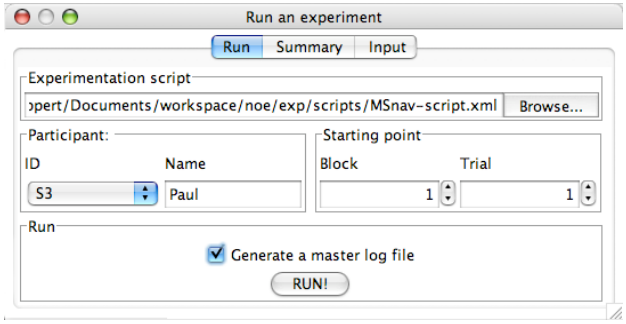

Figure 7. Experiment launcher (*Run* tab).

rest period for the subject. The script may contain references to *experiment components* such as Message (line 3) or PointingBlock (line 4). These components are Java objects that implement trials, inter-titles (interblock or intertrial) and end criteria. They are loaded dynamically by the experiment launcher and are the primary mechanism for extending the run platform.

The core of the experiment launcher executes a state machine (Figure 8) whose events come both from the human subject and the XML script. Each block (line 2), interblock (line 1), trial (lines 4-5) and intertrial (line 3) specifies a *class* and *end criterion* that reference components. The state machine calls these components during state transitions, e.g., when a trial starts and ends, and in guards to decide if a trial is over.

For example, lines 1 and 3 use the Message component to display a message before each trial ("Touch the target as quickly as possible") and each block ("New block"). The end criterion specifies which action the user has to perform to get to the next step, i.e. to start a block, to start a trial or to end a trial. Several criteria are predefined and can be freely combined with boolean operators. For example, line 1 specifies waiting until the user presses the space bar to start a block, while line 2 specifies that any trial in the block is terminated either when the user has dwelled on an object for one second (1000 ms) or if the trial has lasted more than three minutes (180 000 ms). In the latter case, a fail condition and the measure "Too Long" will be output to the table log.

The run platform supports *factors* and *measures* in a flexible way. A factor is a value set by the script when a block or trial starts, whereas a measure is a value updated by one or more components as the experiment is run. Any experiment component can export measures. For example, a block typically provides the measures *end time* and *success* (hit or miss) for the current trial.

As the experiment is run, the experiment launcher collects the available measures and outputs them to a *table log* and a *cinematic log* as specified in the XML script (not shown

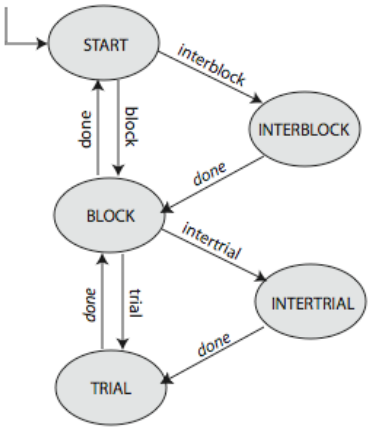

Figure 8. State Machine running the experiment (plain text: script events, *italic*: user events).

```
# MSnav:Compare multiscale navigation techniques
# TQ:technique
# ID:Index of Difficulty
# W: Target Width
# MT:Movement Time
# HIT:end trial
# D:distance
# Date: 2006-09-22-11-00-10
```

exp	subject	block	trial	TQ	W	D	ID	MT	HIT
MSnav	S1	1	1	OZ	200	204600	10	12094	dwell
MSNav	S1	1	2	OZ	400	12094	15	6578	dwell

Figure 9. Example *table* log.

```
# Experiment: MSnav
# Subject: S1
# scale:zoom factor
# x: x-cursor
# y:y-cursor
# Date: 2006-09-22-11-00-10
```

block	trial	time	scale	x	y
1	1	213386	1.0	56.0	92.0
1	1	213401	1.0	69.0	114.0
1	1	213329	1.0	68.0	108.0

Figure 10. Example *cinematic* log.

in Figure 6). Both files are created when the experiment begins with a name that includes the current date and time. Each file starts with comments describing the experiment and the different measures logged. The rest of the *table log* has one line per trial listing the experimental setting (experiment name, subject, block number, trial number) and the values of the factors and the measures (Figure 9). The rest of the *cinematic log* has one line per change of value of a cinematic measure (Figure 10). The two log files can be readily loaded into most analysis tools, including Excel, R and JMP.

Clicking on the *Summary* tab shows which subjects have been run. A separate file stores subject ids mapped to subject names, which keeps data logs anonymous.

The *Input* tab of the experiment launcher specifies the mapping of input devices to virtual controllers (Figure 11). We designed a *generalized input library* (GIL) to access any input device and specify virtual ones because the Java input libraries are limited to a standard mouse and keyboard. Components need not use GIL: they can manage input devices directly if necessary.

GIL is implemented on top of the JInput system[6] and offers simple and flexible naming and configuration mechanisms. JInput describes input devices in terms of controller devices, e.g., a mouse, and components that generate values, e.g., mouse buttons or a mouse's X displacement. JInput can query which controllers are available and generate events when any component changes state. GIL provides access to these physical components values using a syntax of the form <controller name>.<component name>. For example, for a laptop computer with a USB mouse connected, JInput defines three controllers: TRACKPAD, KEYBOARD and USB MOUSE,

[6] http://jinput.dev.java.net/

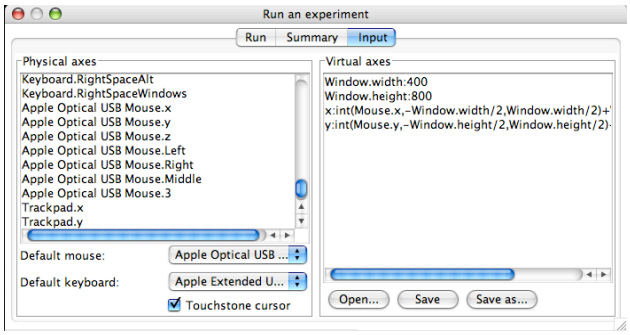

Figure 11. Experiment launcher (*Input* tab).

with several components for each, e.g., X or BUTTON1. Available physical components are listed in the left pane of the *Input* tab (Figure 11).

Virtual controllers are defined in the right pane as a set of *virtual axes*. Each axis is specified by an expression using operators such as *add*, *multiply*, *min*, *max*, *exp, if-then-else*, *integration,* which transforms relative values sent by an axis into an absolute value and *difference*, which does the opposite. For example, the relative input from a mouse can be transformed into an absolute screen position as follows: x=integrate(Mouse.x, 0, Window.width).

A major benefit of using GIL is the ability to test different input devices for the same interaction technique without changing or even recompiling it. For example, a generic Pan & Zoom component could be programmed to use a virtual "scale" axis. Bi-manual Pan & Zoom [4] could then be obtained by defining "scale" as a function of a joystick slider while the Zliding technique [19] would define "scale" as a function of stylus pressure.

TDE: Touchstone Development Environment

The run platform is designed to facilitate the reuse of existing components and the development of new ones. In the previous section, we assumed that the experiment components referenced by the script were already available in the Touchstone repository. This section describes how to create new components (intertitles, blocks, criteria, factors and measures) with Touchstone's Development Environment (TDE).

The implementation of the run platform uses the *factory* design pattern [11] so that each experiment component can be referenced by a simple name. Touchstone contains three factories: one for blocks, one for intertitles and one for criteria. To create new components, the experimenter extends Java classes such as Block and overrides some methods such as those called by the state machine from Figure 8. To register components with the factories, the experimenter annotates the Java code with comment tags such as @touchstone.block. A TDE tool then generates the appropriate code to register the components with the factories and an XML description file that makes them available to the design platform.

Figure 12 shows how to define a block consisting of a series of pointing tasks. First, the tag @touchstone.block registers this class with the block factory so this block can be referenced by one of the two names PointingBlock or

```
1  /**    @touchstone.block FittsBlock
2         @touchstone.block PointingBlock
3  */
4  public class FittsBlock extends Block {
5     public FittsBlock() { super(); }
6     public void beginBlock() { /* create view */ }
7     public void beginTrial() { /* create target */}
8     public void endTrial(EndCondition ec) { ... }
9     public void endBlock() { ... }
10 }
```

Figure 12. PointingBlock code

FittsBlock (Figure 12, lines 1 and 2). Next, the methods begin/endBlock and begin/endTrial are overridden to create/delete the view and the object representing the target (Figure 12, lines 6-8).

For graphical output, Touchstone provides a scene graph and a zoomable viewer similar to but simpler than those of Piccolo [3] and ZVTM [18]. However the experimenter can use any Java toolkit such as Swing or an OpenGL-based toolkit for graphically demanding techniques such as perspective pointing [12] or Fisheye views [6]. Non-graphical output (and input) can also be used: the launcher does not depend in any way on the use of a graphical display or input device to run the experiment.

Factors and measures are registered by the components with a global Platform object. For example, the FittsBlock class above queries the Platform object to get the values of the *ID* and *W* factors and creates an object representing the target of the right size at the right location (Figure 12, line 7). It also queries the Platform object for measures such as movement time and hit/miss and updates them when the end condition of the trial is met. By default, the values are simple strings, but new factor types and values can be created. For example, Figure 13 shows the definition of the Technique (TQ) categorical factor and one of its values (SDAZ). The TDE uses the tags @touchstone.factor and @touchstone.value to export this factor to the design platform. Measures are created similarly.

```
1  /**    @touchstone.factor TQ
2         name: "Technique"
3  */
4  public class TechniqueFactor extends Factor {
5     public TechniqueFactor() { super("TQ"); }
6     public void setValue(Object value) {
7        // init technique parameters
8     }
9  }
10
11 /**    @touchstone.value SDAZ
12        factor: TQ
13        name : "SpeedDependantAutomaticZooming"
14        help: "rate-based scrolling with zoom factor
15                  adapted to the scrolling speed"
16 */
17 public class SDAZTechnique {
18    public SDAZTechnique() { … }
19    // methods for SDAZ
20 }
```

Figure 13. The technique factor (TQ) and the SDAZ value

When using the design platform, it is often convenient to manipulate factors and measures that have not yet been implemented in the run platform. The experimenter need only define empty Java classes for the corresponding components and specify the appropriate @touchstone tags. The TDE generates the proper input to the design platform so that, for example, the Technique factor displays a menu with the proper list of techniques.

In summary, Touchstone's run platform is applicable to a wide variety of experiments implemented in Java. The state machine runs the experiment by interpreting the XML script and user input and calling components, while the TDE is used to create new components.

Analysis Platform

The analysis platform is provided as a convenience for the experimenter and offers limited advice on appropriate statistical tests for the current experiment. It refers to R and JMP[7], two data analysis packages that can read the table logs generated by the run platform.

EVALUATION

Evaluating Touchstone involves determining whether or not it effectively supports experimenters in the design and implementation of controlled experiments. We used Touchstone to create two experiments: the first replicates and extends a published experiment and the second illustrates how to create a new experiment.

Replicating and Extending an Experiment

We first used Touchstone to replicate the design of a recent experiment [2] comparing OrthoZoom Slider (OZ) and Speed-Dependent Automatic Zooming (SDAZ) [14]. We then extended it, adding two additional techniques: standard unimanual Pan & Zoom (PZ) and a bimanual version of Pan & Zoom (PZB) that had been tested in a different experiment [4].

Method

We conducted a 4x2x2 within-subjects experiment to compare the relative efficiency of four interaction techniques on a 1-dimensional multistage pointing task. The original experiment [2] had three independent variables: *technique*, *index of difficulty* (*ID*) and *target width* (W*)*. We used these same factors, but added two new values for technique (PZ and PZB). Once entered into the design platform, we quickly realized that we had to reduce the number of *ID*s and *W*s to keep the experiment length and number of subjects manageable. We settled for 16 unique trial types: 4 techniques (PZ, PZB, OZ, SDAZ[8]) x 2 IDs (10, 15) x 2 *W* (200, 400).

Experimental trials were organized into four technique-specific blocks to avoid disturbing subjects with successive changes among techniques. The block presentation order was counterbalanced across subjects using a Latin square. All four combinations of *ID* and *W* appeared in each block. We tried several numbers of replications and checked its effect on the length of the experiment. We selected four replications, i.e. 32 trials per block, counterbalanced with a Latin square.

In an ideal world, we would simply use Touchstone's recommendation of 16 subjects, with all orders of all values of each factor counterbalanced for order via a Latin Square. But we were interested in seeing whether or not Touchstone could help an experimenter who faced a constraint, in this case, access to 12 instead of 16 subjects.

Touchstone allows 'what if' exploration of alternative solutions. For example, if we omit *W* (target width) as a factor, giving it a constant value for all four combinations of technique and *ID*, it would limit our ability to make claims about other values of *W*, but would require only four subjects for a complete Latin square, which could be replicated three times to get 12 subjects. A better alternative is to 'devalue' *W* at the block level. This results in a well-distributed, but not complete set of orders of each *W* value. Before the experiment, we used Touchstone's experiment summary to test the resulting distribution of trials in JMP to ensure they were balanced. We also ran post-hoc analyses of the experimental data to ensure that there were indeed no order effects due to *W*.

We measured two dependent variables: task completion time and target hit/miss. A miss was recorded when the subject did not end the trial within 3 minutes. This never occurred during the experiment. At the end of the experiment, we asked subjects to rank the techniques by preference order, using a questionnaire.

Subjects

Twelve unpaid adult volunteers, 9 male and 3 female, aged 26 years on average, served in the experiment.

Apparatus

We used Touchstone to design the experiment and create the XML script used by the run platform. We used an HP workstation with a 2 GHz Pentium 4, using a 1280×1024 LCD monitor and an optical mouse. Window size was set to 600×800 pixels and the document length was 2^{30} pixels.

Procedure

After a 2-minute introduction, we demonstrated the task to each subject. Prior to each experimental block, subjects practiced eight randomly-chosen trials of the technique from the upcoming block. Subjects were asked to point as quickly as possible at a series of targets that appeared sequentially in a document too large to be viewed at its natural size without scrolling. Subjects had to scroll vertically to bring the target into view. An arrow indicated the direction where to look for the target. A horizontal orange line, insensitive to the zoom factor, always indicated the target location. The target was also surrounded by concentric circles sensitive to the zoom factor. The task was complete when the cursor remained over the target for one second at a zoom factor of 1. The target then changed from blue to green.

---

[7] R (open-source): *http://www.r-project.org*;
JMP (SAS Institute, Inc): *http://www.jmp.com*.

[8] To calibrate SDAZ, we used Cockburn's formula with k=0.05 and threshold = 20 [7].

As soon as the target turned blue, the end time of the current trial was logged. Another trial began as soon as the subject pressed the mouse button on the target that had just been reached. This target disappeared, a new target appeared at another location, and the start time of the next trial was logged.

Hypotheses

We expected our results to be consistent with those reported in the literature: the original experiment comparing OZ with SDAZ [2] reported that OZ outperforms SDAZ, while Bourgeois and Guiard [4] showed that PZB is more efficient than PZ because it allows parallel input. Therefore our hypotheses were:

H1: OZ < SDAZ

H2: PZB < PZ

Results

As in the OZ vs. SDAZ experiment [2], there was no significant interaction effect of *technique*×*W* on *MT* ($F_{3,173}$<1). We also verified that there was no presentation order effect since there was no significant interaction effect on *MT* of block number by *technique*. Finally, analyzing the effect on *MT* for replication number by *technique* showed that there was no learning effect except for SDAZ ($F_{3,177}$=3.36, p=0.02).

Analysis of variance with the REML method for repeated measures revealed a significant simple effect on movement time (*MT*) for *technique* ($F_{3,177}$=44.9, p<0.0001) and a significant simple effect on *MT* for *ID* ($F_{1,179}$=12.4, p<0.0005) but no significant *technique*×*ID* interaction effect ($F_{3,173}$<1). Tukey post-hoc paired comparisons reveal three groups (Figure 14): OrthoZoom is the fastest technique, followed by Bimanual Pan & Zoom, while SDAZ and Unimanual Pan & Zoom are equally slow: OZ < PZB < {PZ, SDAZ}. Subjects' subjective preferences follow a similar pattern: OZ is most preferred (average rank 1.6), then PZB (2.1), SDAZ (2.8), and PZ (3.6). These results support hypotheses H1 and H2, reinforcing the performance comparisons reported in the literature. Furthermore, they indicate that OrthoZoom is still the fastest technique for multiscale navigation in one dimension and that Unimanual Pan & Zoom performs comparably to SDAZ, although there seems to be a stronger effect of *ID* for SDAZ (Figure 14).

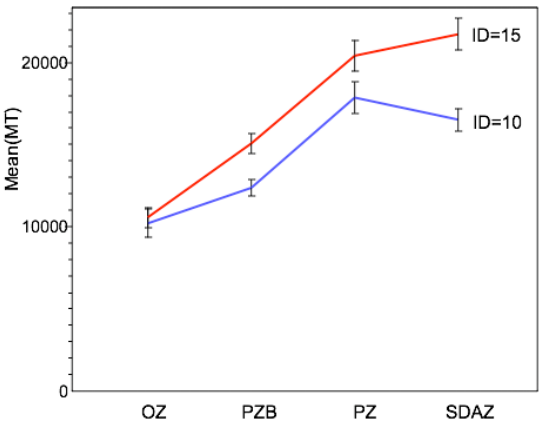

Figure 14. Mean movement time for each technique and ID

Designing and Running a New Experiment

We wanted to test how well Touchstone supports the design and running of a new experiment. We chose to measure the effect of number of distractors on reaction time. Figure 15 shows the steps of a generic run.

A set of shapes, including a red target and a set of black distractors, appeared on the screen. All shapes were equi-distant from the cursor, so *movement time* (estimated using Fitts' law) could be subtracted from *total time* to define a *reaction time* measure. Using TDE, we created the intertitles, blocks, trials and end criteria components listed in the right column that were not already in Touchstone's repository: StartButton, an intertitle that waits until the subject clicks on the start shape and ReactionBlock, which displays a target and distractors and provides a measure of reaction time. Each trial ended with a *hit* (click on target) or a *miss* (click on distractor); PressOnTag(<tag>), a criterion triggered when the subject clicked on a tagged shape, used by StartButton and ReactionBlock to start and end a trial, respectively.

We generated the file describing these new components with the TDE and imported it into the design platform. It took less than five minutes to create a simple, one-factor design (number of distractors) with one measure (RT). Next, we implemented the new components with SwingStates [1] in about 80 lines of code. The experiment was then ready to run with the Launcher.

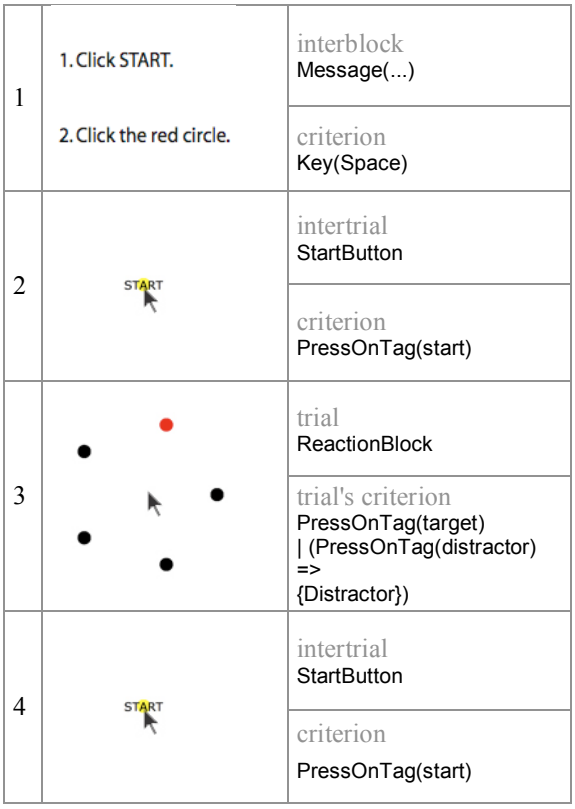

Figure 15. Scenario of an experiment to measure reaction time.

CONCLUSION AND FUTURE WORK

Touchstone is an exploratory experiment design platform with the goal of improving the scientific foundations of research in HCI. It enables experimenters to design and run controlled experiments, encouraging replication and extension of prior research by making experiment designs, components and results available in a repository.

Touchstone's modular structure includes design, run and analysis platforms. Researchers may use only those components relevant to their particular work and extend Touchstone with their own components. Touchstone can handle a substantial proportion of controlled experiments found in the HCI literature related to the evaluation and comparison of interaction techniques. We hope that sharing experiment designs, run-time components and data logs within the HCI community will help build a repository of techniques, corpora, and device configurations, and, in turn, improve the state of the art.

This paper describes the architecture and use of Touchstone and demonstrates that we can both replicate and extend experiments from the literature and generate new experiments from scratch. Our replicated experiment provides new findings that extend previous results: OrthoZoom is significantly faster than bimanual Pan & Zoom, and both are significantly faster than SDAZ and unimanual Pan & Zoom.

Touchstone is an open source project available to the HCI community[9]. We hope it will provide a resource for experimenters, students and educators, from initial experiment design to running experiments and final analysis. Our future work will focus on supporting additional experiment types in the design platform, supporting questionnaires, e.g., for pre-testing and debriefing, enriching the run platform with a wider choice of components and extending the analysis platform.

REFERENCES

1. Appert, C., and Beaudouin-Lafon, M. SwingStates: Adding state machines to the swing toolkit. *Proc. UIST 2006*, ACM Press (2006), 319-322.
2. Appert, C., and Fekete, J. OrthoZoom Scroller: 1D multi-scale navigation. *Proc. CHI 2006*, ACM Press (2006), 21-30.
3. Bederson, B.B., Grosjean, J., and Meyer, J. Toolkit design for interactive structured graphics. *IEEE Trans. Software Eng. 30*, 8 (2004), 535-546.
4. Bourgeois, F., and Guiard, Y. Multiscale pointing: facilitating pan-zoom coordination. *Ext. Abstracts CHI 2002,* ACM Press (2002), 758-759.
5. Card, S., Moran, T., and Newell, A. *The Psychology of Human-Computer Interaction*, Erlbaum, 1983.
6. Carpendale, M.S., and Montagnese, C. A framework for unifying presentation space. *Proc. UIST 2001*, ACM Press (2001), 61-70.
7. Cockburn, A., Savage, J., Wallace, A. Tuning and testing scrolling interfaces that automatically zoom. *Proc. CHI 2005*, ACM Press (2005), 71-80.
8. Eisenstein, J., and Mackay, W.E. Interacting with communication appliances: An evaluation of two computer vision-based selection techniques. *Proc. CHI 2006*, ACM Press (2006), 1111-1114.
9. Fitts, P.M. The information capacity of the human motor system in controlling the amplitude of movement. *J. Exp. Psychology 47* (1954), 381-391.
10. Gaines, B. Modeling and forecasting the information sciences. *Information Sciences 57-58* (1991), 3-22.
11. Gamma, E., Helm, R., Johnson, R., and Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, USA, 1995.
12. Guiard, Y., Chapuis, O., Du, Y., and Beaudouin-Lafon, M. (2006). Allowing camera tilts for document navigation in the standard GUI: A discussion and an experiment. *Proc. AVI 2006*, ACM Press (2006), 241-244.
13. Guiard, Y., Du, Y., Fekete, J.D., Beaudouin-Lafon, M. Appert, C., and Chapuis, O. Shakespeare's complete works as a benchmark for evaluating multiscale document-navigation techniques. *Proc. BELIV 2006, AVI Workshop on Visual Interfaces,* ACM Press (2006), 65-70.
14. Igarashi, T., and Hinckley, K. Speed-dependent automatic zooming for browsing large documents. *Proc. UIST 2000*, ACM Press (2000), 139-148.
15. Kuhn, T. *The Structure of Scientific Revolutions*. Univ. Chicago Press, 1962.
16. Mackay, W.E. Which interaction technique works when? Floating palettes, marking menus and toolglasses support different task strategies. *Proc. AVI 2002*, ACM Press (2002), 203-208.
17. MacKenzie, I.S., and Jusoh, S. An evaluation of two input devices for remote pointing. *Proc. EHCI 2001*, Springer-Verlag (2001), 235-250.
18. Pietriga, E. A Toolkit for addressing HCI issues in visual language environments. *Proc. VL/HCC 2005*, IEEE (2005), 145–152.
19. Ramos, G., and Balakrishnan, R. Zliding: Fluid zooming and sliding for high precision parameter manipulation. *Proc. UIST 2005*, ACM Press (2005), 143-152.
20. Soukoreff, R.W., and MacKenzie, I.S. Generalized Fitts' law model builder. *Conference Companion, CHI 1995*, ACM Press (1995), 113-114.
21. Soukoreff, R.W., and MacKenzie, I.S. Towards a standard for pointing device evaluation: Perspectives on 27 years of Fitts' law research in HCI. *Int. J. Human-Computer Studies 61* (2004), 751-789.

[9] http://touchstone.lri.fr