COMP 3008: Assignment 2
Testing Memorability of Unistroke Gestures

Jeff Wilson
Nicolas Porter
Andy Zemancik

April 6, 2012

1 Abstract

With recent changes to the HTML5 specification and with Microsoft's Windows 8 design trending toward touch interaction, gesture-based interfaces may soon become more common on the web and the desktop. This broadening reach, as well as the need for more abstract gestures in handheld devices, demand empirical testing to validate the memorability of command-gesture sets. This is particularly important as developers with limited interaction design experience find themselves with web-based tools at their disposal. For this class project we have designed a web-based testing framework for performing gesture set evaluations. We present the framework and a pilot study used to evaluate two different sets of command-gesture pairs.

2 Introduction

According to Wendy Yee [5], it is possible to classify touch gestures into "direct" interaction, where we leverage the familiarity of interacting with objects in the world, and "indirect" interaction, where we must leverage users' mental models for more abstract input commands. In the more abstract case, the semantics of the command must align with the shape of the gesture if we have any hope of providing a satisfying user experience.

While mass-market development might be expected to meet this challenge with teams of interaction designers. Guidelines for methodology suggested by interaction designers like Saffer [3] recommend research and empirical observation to arrive at appropriate insights for application design. However it seems probable that the ease of development and deployment on the Internet will give rise to more ad-hoc gestural vocabularies appearing on web applications.

At the University of Washington, Wobbrock has recognized the need to make empirical gesture testing available to a wider audience of developers and designers [4]. They developed a simple gesture recognition algorithm that could be implemented in JavaScript. The $1 Unistroke Recognizer makes it possible to rapidly develop and deploy low-cost gesture testing tools using widely available web frameworks.

3 Test Framework

We designed a gesture testing application to capture study gestures in general, and the $1 Unistroke Recognizer in particular. Screenshots for the testing interfaces are shown in the appendix.

Our design goals were to:

- permit ad-hoc mapping of gestures to commands
- enable testing on desktops and tablets
- support "null tests" such as distractors and participant specific surveys
- present gesture-command pairs in randomized order for each participant
- support gathering of statistics for alternative gesture sets
- enable exporting of statistical data for analysis

For the backend, we chose the Django Web Framework [1]. Based on the Python programming language, it promised to meet the needs of "perfectionists with deadlines", a fitting description of our team.

For the frontend, we rely on some of the older and widely supported HTML5 and CSS3 features like the *article* and *canvas*, and the *box-shadow* property. We also utilize the ubiquitous *jQuery* javascript library.

3.1 Client Side Architecture

Most of the client side logic is for the scenario test step page. Communication between objects is done via *jQuery* events, which means that it is very easy to add, remove or modify modules. The page communicates asynchronously with the server.

Here's the list of JavaScript objects found in the client side code:

3.1.1 Scenario Test Step

This object deals with the server communication, which is done asynchronously. This object either sends events to the server, or checks if the correct gesture has been drawn. This object also fires off an event if the gesture does not pass server validation (i.e. the user did not draw the correct gesture).

The events sent to the server are the following:

- When the "Show gesture list" button is clicked, a *ShowGestureList* event is sent.
- When the "Hide gesture list" button is clicked, a *HideGestureList* event is sent.
- When a gesture is drawn, a *Gesture* event is sent.

3.1.2 Gesture

This object represents a gesture that was drawn on the touch surface, and encapsulates the recognizer of the dollar library. An instance is constructed everytime a gesture is drawn.

3.1.3 GestureWidgetView

This objects modifies the HTML page according to various events. For example, when it receives an *invalid-gesture* event, it displays a notification bubble informing the user that he did not draw the correct gesture. It is also responsible for sending events when the user presses the *show* or *hide* gesture list button as well as the *skip* button.

3.1.4 GestureRecognizerWidget

This object processes mouse and touch events. It uses the canvas to display the path being traced by the user. It also creates a gesture object and then sends it to the rest of the system via a *gesture-recognized* event.

3.2 Server Side Architecture

3.2.1 Test Models

We designed the system to be easily extended. The current test framework is shown in Figure 1. The Test model works with the abstract TestStep class. One can simply implement a new type of TestStep if needed. A TestStep, as the name suggests, is a step in our test. These test steps are ordered, which means that the Test model can be thought of as a sequence of ordered test steps, associated with a User.

A test step must implement two methods: One to verify that the test step is complete (i.e. verify that the user can proceed to the next test step), and the other to render itself. We have two test step implementations: *ScenarioTestStep* and *SimpleTestStep*. The *ScenarioTestStep* model is used for the gesture scenarios. It is built from a *ScenarioType*, a model that describes some scenario or task the user needs to perform (e.g. draw the “Accept” gesture). The verification method of this step checks whether the gesture drawn by the user is the correct one for the current scenario. The other test step implementation, the *SimpleTestStep*, merely displays an html page. This type of step is used to display the page that indicates the user should perform the distraction test.

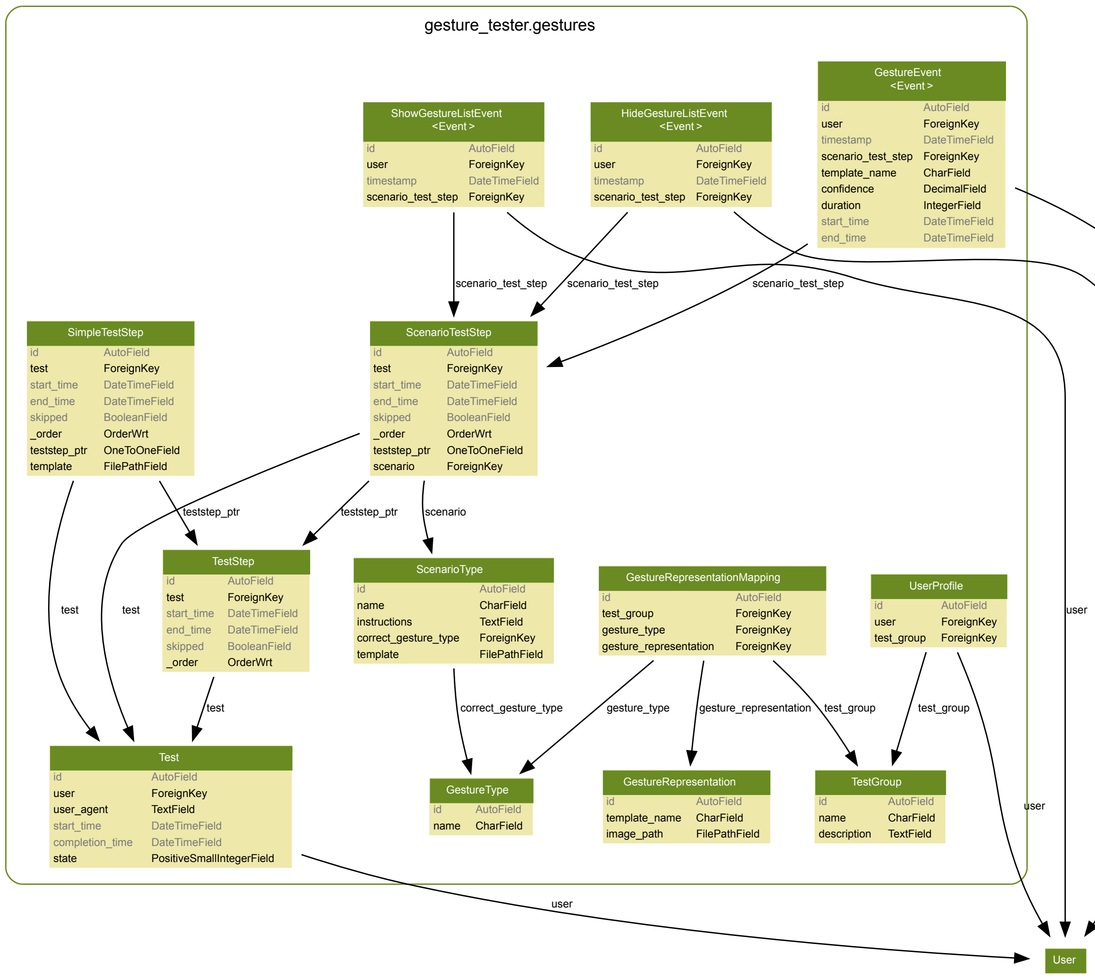
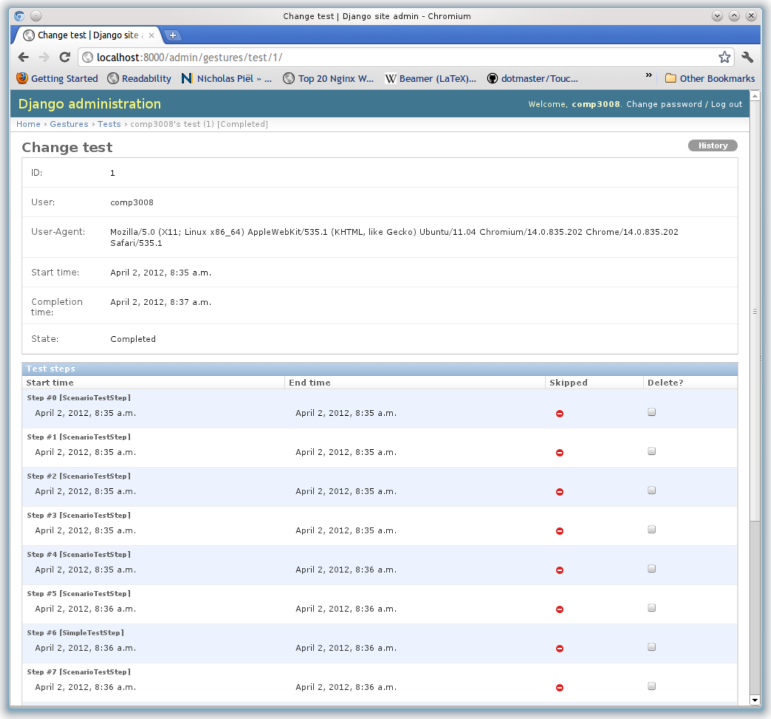


Figure 1: Graph of the models, with relationships

When a user logs in and clicks on the *Proceed* button, the system builds sequence of Scenario test steps, which are generated from the available *ScenarioType* models. The sequence is then shuffled, and a *SimpleTestStep* for the distraction test is inserted in the middle. The sequence is then assigned to a new *Test* instance.

3.2.2 Gesture Models

The gesture system is split up in 3 different models: *GestureType*, *GestureRepresentation* and *GestureRepresentationMapping*. The *GestureType* model represents the meaning of a gesture, independently from its representation. The *ScenarioType* model uses this model. The *GestureRepresentation* model represents the physical representation of a gesture. Its name maps directly to the gesture template name of the dollar library. In a future implementation, the gesture representation instances should populate dollar's gesture template vocabulary. The *GestureRepresentationMapping* model is a group-specific mapping from gesture type to gesture representation. It allows us to change a gesture's representation according to the user's group.



Change test | Django site admin - Chromium
Change test | Django site
localhost:8000/admin/gestures/test/1/
Getting Started
Readability
Nicholas Piël » ...
Top 20 Nginx W...
Beamer (LaTeX)...
dotmaster/Touc...
Other Bookmarks
Django administration
Welcome, comp3008. Change password / Log out
Home › Gestures › Tests › comp3008's test (1) [Completed]
History

Change test

ID:	1
User:	comp3008
User-Agent:	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/535.1 (KHTML, like Gecko) Ubuntu/11.04 Chromium/14.0.835.202 Chrome/14.0.835.202 Safari/535.1
Start time:	April 2, 2012, 8:35 a.m.
Completion time:	April 2, 2012, 8:37 a.m.
State:	Completed

Test steps

Start time	End time	Skipped	Delete?
Step #0 [ScenarioTestStep]			
April 2, 2012, 8:35 a.m.	April 2, 2012, 8:35 a.m.	⛔	☐
Step #1 [ScenarioTestStep]			
April 2, 2012, 8:35 a.m.	April 2, 2012, 8:35 a.m.	⛔	☐
Step #2 [ScenarioTestStep]			
April 2, 2012, 8:35 a.m.	April 2, 2012, 8:35 a.m.	⛔	☐
Step #3 [ScenarioTestStep]			
April 2, 2012, 8:35 a.m.	April 2, 2012, 8:35 a.m.	⛔	☐
Step #4 [ScenarioTestStep]			
April 2, 2012, 8:35 a.m.	April 2, 2012, 8:36 a.m.	⛔	☐
Step #5 [ScenarioTestStep]			
April 2, 2012, 8:36 a.m.	April 2, 2012, 8:36 a.m.	⛔	☐
Step #6 [SimpleTestStep]			
April 2, 2012, 8:36 a.m.	April 2, 2012, 8:36 a.m.	⛔	☐
Step #7 [ScenarioTestStep]			
April 2, 2012, 8:36 a.m.	April 2, 2012, 8:36 a.m.	⛔	☐

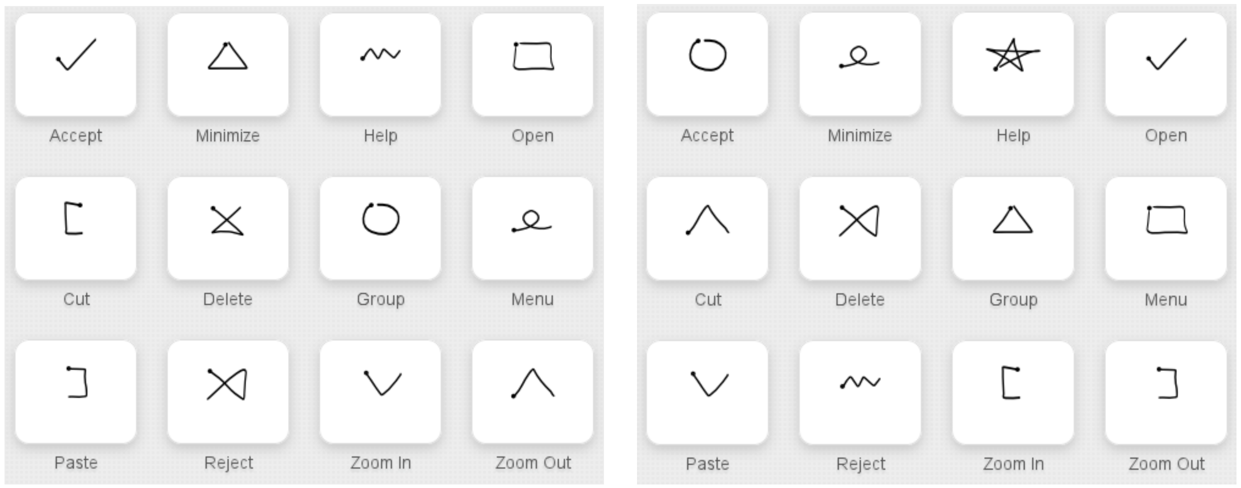
Figure 2: Metrics as they appear in our Django app.

3.3 Metrics

Figure 2 shows a screenshot of the metrics for one participant's test. The framework relates test events to participant tests automatically. This makes it fairly simple to write a variety of export tools that leverage the framework classes. Of particular interest were the stats derived from the events shown in the diagram such as the number of successes on first attempt, the number of times the participant consulted the gesture list, and the duration of any given test.

We supplemented the framework statistics with a post-test survey with the following goals:

- capture details about the user that we could not from the tests
- get feedback about gestures without having to examine the data
- validate the test enviroment as well as the gestures sets
- determine users overall feeling about the exercise
- get users opinion on use of the gestures
- determine users exisitng familiarity with gestures
- get user feedback about the test enviroment (through additional comments)



(a) Natural (b) Unnatural

Figure 3: Gesture sets for the two test conditions.

4 Evaluation

For our pilot study we were interested in validating that our process could successfully distinguish the memorability of two different gesture mappings. We used a between-subjects design, with one group presented with the *"natural"* mapping condition (Figure 3a), and the other group using the relatively *"unnatural"* mapping condition (Figure 3b).

Participants were not informed in advance of the quality of the mappings. We modeled success on the number of correct first guesses where the participant did not have to consult the gesture list for hints. We also tested for the number of times the participant needed to return to the gesture listing and the overall completion time of each task.

Each participant was given a brief written introduction to the concept of gestures. They were then asked to try to commit the twelve gestures to memory. Then they were presented with a sequence of six gesture recall tasks, randomly ordered from the twelve mappings they had studied. We then interrupted the recall task with a Mental Rotations Test [2] presented on paper and lasting 5 minutes. After the distraction task the participant completed the six remaining gestures and then finished up with a brief post-test survey.

Time constraints restricted this pilot study to convenience sampling. We had a total of 21 participants with estimated ages ranging from 20 to 75. Approximately 40% of the participants used a large wall-mounted touch display and the remainder used mouse or touchpad to draw the gestures. We did not control for access to the touch display, however in practice this subset was evenly split between the two conditions.

5 Observations

Paired-samples t-tests were conducted to compare *successful first attempts* (Figure 4a), *lookups required* (Figure 4b), and *completion times* (Figure 4c) in the natural and unnatural conditions.

There was a significant difference in *successful first attempts* between the natural (M=7.92, SD=1.98) and unnatural (M=4.58, SD=2.02) conditions; t(11)=5.6, p=0.00016. This suggests that the two gesture sets really were different, with the natural mapping appearing to improve the probability of recalling gestures correctly on the first attempt.

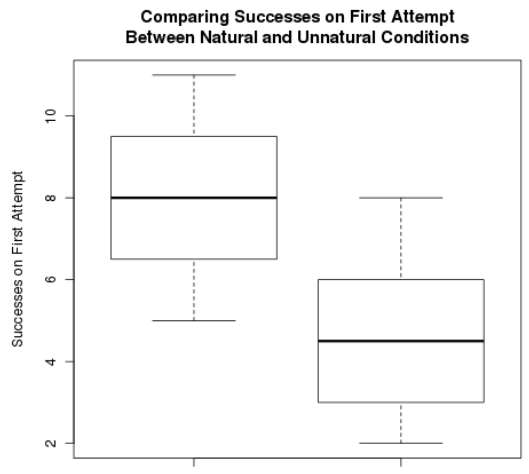
Similarly there was a significant difference in the number of lookups required: natural (M=3.5, SD=2.43), unnatural (M=6.25, SD=2.18); t(11)=-4.37, p=0.001. This suggests that the natural mapping reduced the need to seek reminders from the gesture list.

There was a less extreme but still significant difference in completion times in the natural (M=1167, SD=275) and the unnatural (M=1532, SD=567) conditions; t(11)=-2.07, p=0.06. This suggests that users were able to complete the tasks more efficiently with the natural mapping.

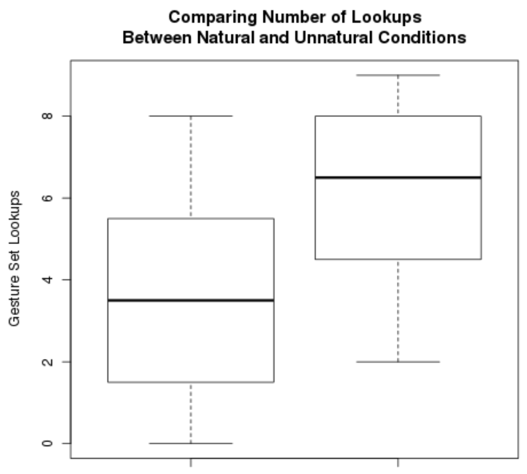
The $1 Unistroke Recognizer also reported scores for the gestures it recognized (Figure 4d). We modeled these as recognizer confidence and averaged them for each participant task. We observed a significant difference in a paired t-test in the two conditions for this metric as well (natural: M=79.58, SD=3.6; unnatural: M=71.8, SD=3.7; Z=7.36, p=0.000014). This suggests that participants in the natural condition were significantly more likely to represent the gesture in a form correctly interpreted by the recognizer.

To confirm that participants' subjective impressions aligned with our instrumented results, we ran a Mann-Whitney's U test on the 5-point Likert scale question relating to the ease of recall. We found a significant effect of the gesture mapping condition (natural: M=3.8, unnatural: M=2.9; Z=2.10, p=0.037), supporting the effects observed in the data.

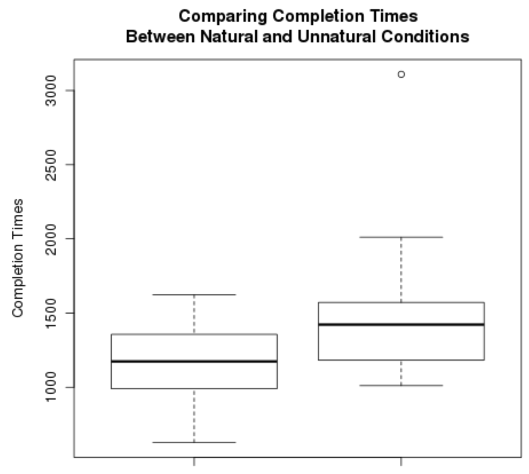
We also ran Mann-Whitney's U tests on the 5-point Likert questions relating to prior confidence with gestures. Again we observed a small effect of gesture mapping but in the opposite direction, indicating that participants reported greater prior confidence with gestures in the *unnatural* condition than in the *natural* condition (Z=-1.49,p=0.138). The same could not be said of their prior familiarity with gestures (Z=-0.66, p=0.55).



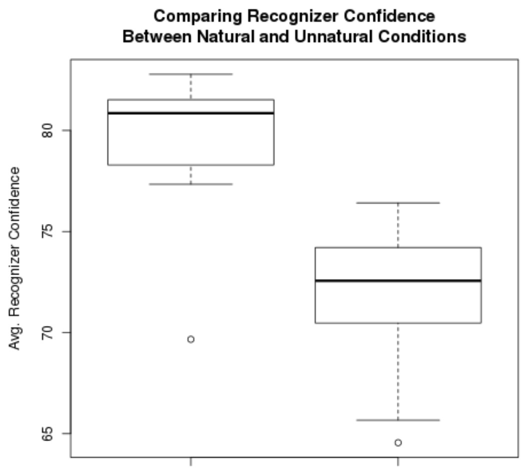
(a) Success on First Attempt



(b) Lookups Required

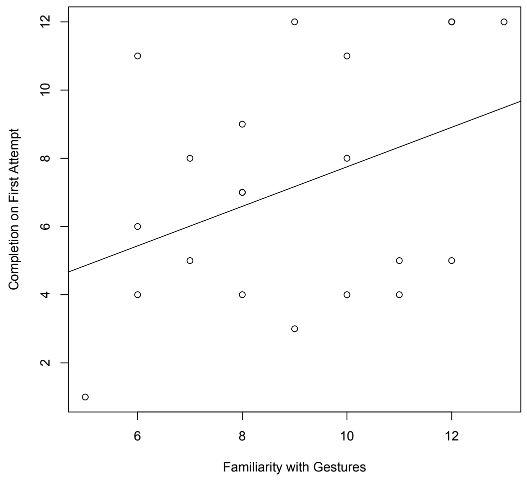


(c) Completion Times

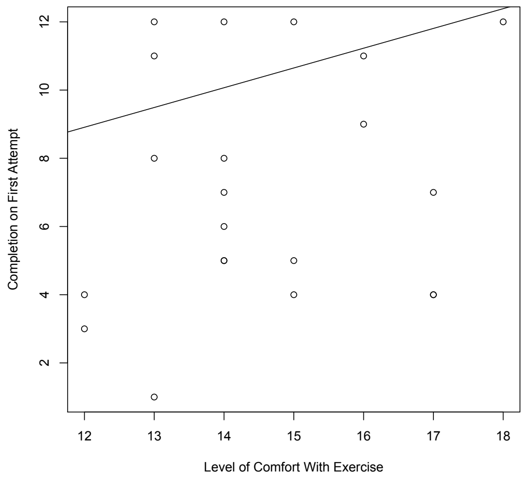


(d) Completion Times

Figure 4: Observations comparing *natural* and *unnatural* mapping conditions. Note: for each box plot the leftmost image is the *natural* condition.



(a) Correlation of Familiarity



(b) Correlation of Comfort

Figure 5: Observations showing a correlation between success and self-reported familiarity and comfort with gestures.

6 Discussion

While our initial conclusions appeared to show an advantage in the *natural* condition, several participants reported issues with the $1 Unistroke Recognizer. We witnessed several cases of the participant offering a minor variation of the correct gesture that was incorrectly rejected. The recognizer was observed to be invariant to rotation, however we saw several instances of participants providing gestures reflected in the x or y axis. These gestures were not successfully interpreted.

When an unsuccessful recognition occurred, the confidence score provided by the recognizer was lower. This permitted us to compare the severity of the problem across the two conditions. Figure 4d show that this problem was more prevalent in the unnatural condition. Since it conferred an advantage on the natural group, this bias represents an alternative explanation for the results found in the other tests. Our hope had been that the algorithm would effect both conditions equally. The fact that it favoured one condition calls our initially strong conclusions into question.

In Figures 5a and 5b we show moderately positive correlations between both participant prior familiarity and prior comfort with gestures and first-time successes. This is not surprising, and if it could be shown that these users were presented with the unnatural condition then it might have gone some way toward reducing the effect of incorrect recognition on the part of the system. Unfortunately these effects were marginal (see last paragraph of Observations section) and so they cannot begin to restore confidence in the differences between groups.

Participants also commented that the number of gestures to be learned and the sequence of gestures in the mapping lists had effects on their confidence, and this may have led to more frequent use of the lookup option. The presentation order was consistent between the two groups, and should therefore not affect our results. However future tests could be designed to validate different learning regimes.

We did not control for time constraints on the training period. This was discussed in the design phase and it was felt that placing a timer on the learning phase might distract from the memorization task. In practice most participants lingered on this phase for under a minute before proceeding to the test. Some controls should be imposed, though perhaps this can be done by the test administrator.

The two sets of gesture mappings were designed subjectively by the authors to confer an advantage to the *natural* condition, however efforts were made to make the *unnatural* mapping seem a plausible, if not desirable, alternative. Our goal of verifying that the testing framework was capable of distinguishing the two mapping sets initially appeared hopeful based on the results of successes and lookups. The fact that we had sufficient information to call our initial optimism into question can be considered a successful validation of the framework even if the particular gesture mappings themselves could not be validated.

7 Future Work

The most pressing issue must be to address the effect of incorrect interpretation of correctly recalled but poorly executed gestures. Some of these problems could have been improved with gesture drawings that more clearly imparted the required direction of movement or by showing animated representations of the gestures. Another possible approach might have been to require a higher confidence threshold from the recognizer before passing judgment on the user's offered gesture. Additional work would be required to determinw the correct threshold. This value must also be presumed to be distict for different recognition algorithms.

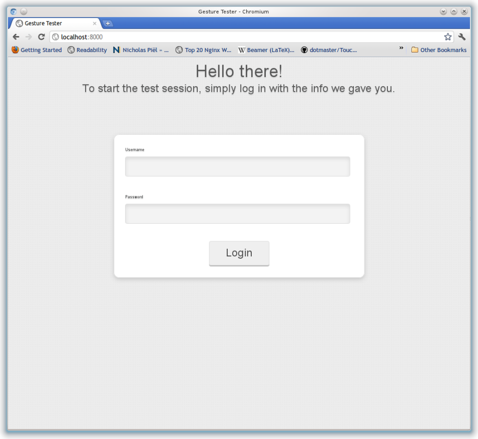
It could be possible to compare the recognition from client-side JavaScript with a more robust recognizer on the server, however this would undercut the advantages offered by making this such a lightweight web-deployable framework. In our study the participants were guided by a human test administrator. Future studies of the memorability could keep track of the number of correct recalls that are not picked up by the software. It is also likely that new JavaScript-based recognizers will emerge that are capable of finer distinctions. Our framework makes it easy to compare different recognizers, and we see an opportunity for future work comparing the robustness of these libraries and their importance in improving user experience.

The flexible nature of the framework lends itself to a number of other study designs. We could explore such factors as variations on training duration, optimal training set sizes, and improved gesture presentation. We also see opportunities for improving test administration and providing reporting and analysis in the browser.

8 Appendix

The source code for this project can be downloaded at the following URL:

http://hotsoft.carleton.ca/repository-files/file.php?h=Rdb7c811b69c4603f80d2cf1c51dce88c



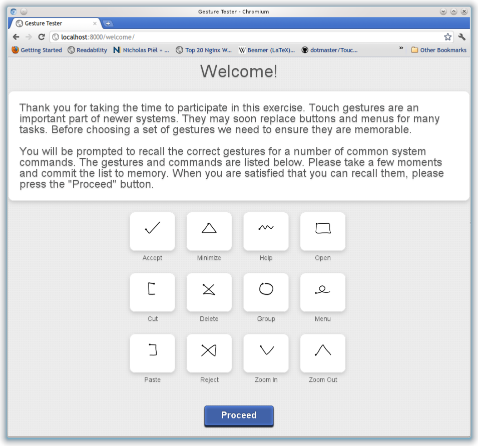
Hello there!

To start the test session, simply log in with the info we gave you.

Username

Password

(a) Login

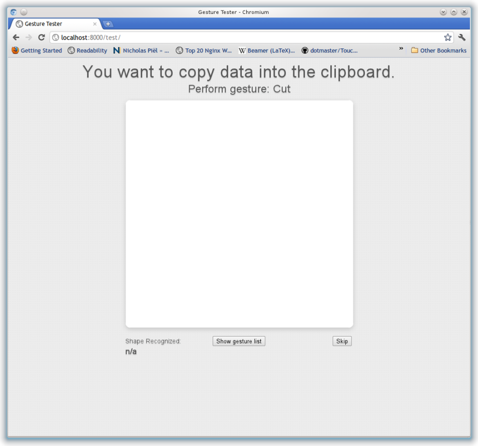


Welcome!

Thank you for taking the time to participate in this exercise. Touch gestures are an important part of newer systems. They may soon replace buttons and menus for many tasks. Before choosing a set of gestures we need to ensure they are memorable.

You will be prompted to recall the correct gestures for a number of common system commands. The gestures and commands are listed below. Please take a few moments and commit the list to memory. When you are satisfied that you can recall them, please press the "Proceed" button.

(b) Welcome



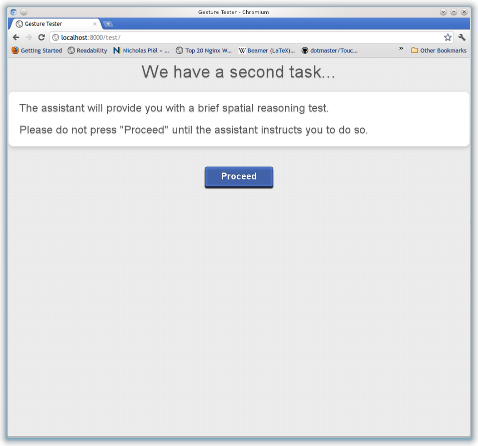
You want to copy data into the clipboard.

Perform gesture: Cut

Shape Recognized:

n/a

(c) Test

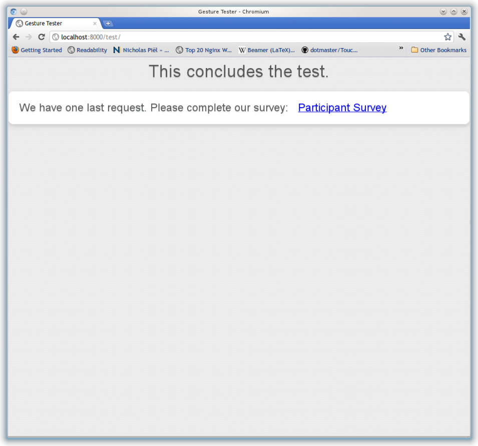


We have a second task...

The assistant will provide you with a brief spatial reasoning test.

Please do not press "Proceed" until the assistant instructs you to do so.

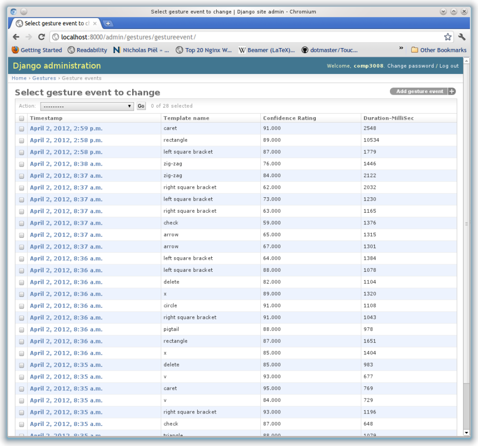
(d) Distractor



This concludes the test.

We have one last request. Please complete our survey: Participant Survey

(e) Thank you



Django administration

Welcome, **comp3008**. Change password / Log out

Home › Gestures › Gesture events

Select gesture event to change

0 of 28 selected

Timestamp	Template name	Confidence Rating	Duration-MilliSec
April 2, 2012, 2:59 p.m.	caret	91.000	2548
April 2, 2012, 2:58 p.m.	rectangle	89.000	10534
April 2, 2012, 2:58 p.m.	left square bracket	87.000	1779
April 2, 2012, 8:38 a.m.	zig-zag	76.000	1446
April 2, 2012, 8:37 a.m.	zig-zag	84.000	2122
April 2, 2012, 8:37 a.m.	right square bracket	62.000	2032
April 2, 2012, 8:37 a.m.	left square bracket	73.000	1230
April 2, 2012, 8:37 a.m.	right square bracket	63.000	1165
April 2, 2012, 8:37 a.m.	check	59.000	1376
April 2, 2012, 8:37 a.m.	arrow	65.000	1315
April 2, 2012, 8:37 a.m.	arrow	67.000	1301
April 2, 2012, 8:36 a.m.	left square bracket	64.000	1384
April 2, 2012, 8:36 a.m.	left square bracket	88.000	1078
April 2, 2012, 8:36 a.m.	delete	82.000	1104
April 2, 2012, 8:36 a.m.	x	89.000	1320
April 2, 2012, 8:36 a.m.	circle	91.000	1108
April 2, 2012, 8:36 a.m.	right square bracket	91.000	1043
April 2, 2012, 8:36 a.m.	pigtail	88.000	978
April 2, 2012, 8:36 a.m.	rectangle	87.000	1651
April 2, 2012, 8:36 a.m.	x	85.000	1404
April 2, 2012, 8:35 a.m.	delete	85.000	983
April 2, 2012, 8:35 a.m.	v	93.000	677
April 2, 2012, 8:35 a.m.	caret	95.000	769
April 2, 2012, 8:35 a.m.	v	84.000	729
April 2, 2012, 8:35 a.m.	right square bracket	93.000	1196
April 2, 2012, 8:35 a.m.	check	87.000	648
April 2, 2012, 8:35 a.m.	triangle	88.000	1079

(f) Events

Figure 6: Screenshots from the project's gesture testing framework.

References

[1] Django home page. `http://www.djangoproject.com/`.

[2] M. Peters, B. Laeng, K. Latham, M. Jackson, R. Zaiyouna, and C. Richardson. A redrawn Vandenberg and Kuse mental rotations test: different versions and factors that affect performance. *Brain and cognition*, 28(1):39–58, June 1995.

[3] Dan Saffer. *Designing for Interaction: Creating Innovative Applications and Devices*. New Riders Publishing, Thousand Oaks, CA, USA, 2nd edition, 2009.

[4] Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*, UIST '07, pages 159–168, New York, NY, USA, 2007. ACM.

[5] Wendy Yee. Potential limitations of multi-touch gesture vocabulary: Differentiation, adoption, fatigue. In Julie Jacko, editor, *Human-Computer Interaction. Novel Interaction Methods and Techniques*, volume 5611 of *Lecture Notes in Computer Science*, pages 291–300. Springer Berlin / Heidelberg, 2009.