StudyCaster: A Tool for Automatic and Anonymized Recording of Online User Studies

Eirik Bakke, David R. Karger, and Robert C. Miller
MIT Computer Science and Artificial Intelligence Laboratory
32 Vassar Street, Cambridge, MA, USA
{ebakke,karger,rcm}@mit.edu

ABSTRACT
In recent years, online crowdsourcing markets like oDesk and Amazon Mechanical Turk have made it easy to recruit human subjects for online user studies, anonymously and in large numbers. A major limitation of online user studies lies in the difficulty of actually observing the actions of the subject as he or she solves the tasks presented, and much work may be necessary on the part of the experimenter in order to instrument the system under test or develop questionnaires that can shed light on the subject's approach to the problem at hand. In this note, we present StudyCaster, a Java-based tool that allows subjects to stream complete recordings of their computer screens to an experimenter's server with a minimum of effort. A novel and crucial feature is anonymization, which blurs any window on the screen that does not match predefined criteria.

ACM Classification Keywords
H.5.2 User Interfaces: Evaluation/methodology

General Terms
Experimentation, Measurement

INTRODUCTION
Online user studies offer many advantages over traditional in-house lab studies. They can be run on a large number of subjects without taking up a proportionally greater amount of the experimenter's time, and they can yield results faster by allowing multiple subjects to work in parallel. Online experiments may also enable subjects to participate anonymously, often allowing the experiment to undergo simplified Institutional Review Board (IRB) review[1]. Finally, online crowdsourcing markets like oDesk[2] and Amazon Mechanical Turk[3] have made it radically easier to recruit and compensate a larger number of subjects for an experiment.

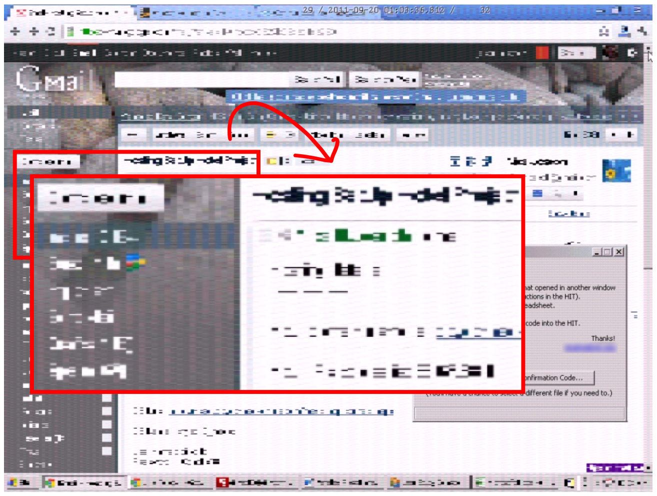


Figure 1. A subject is checking email while working on a study. The screen recorder garbles the browser window content before uploading any data.

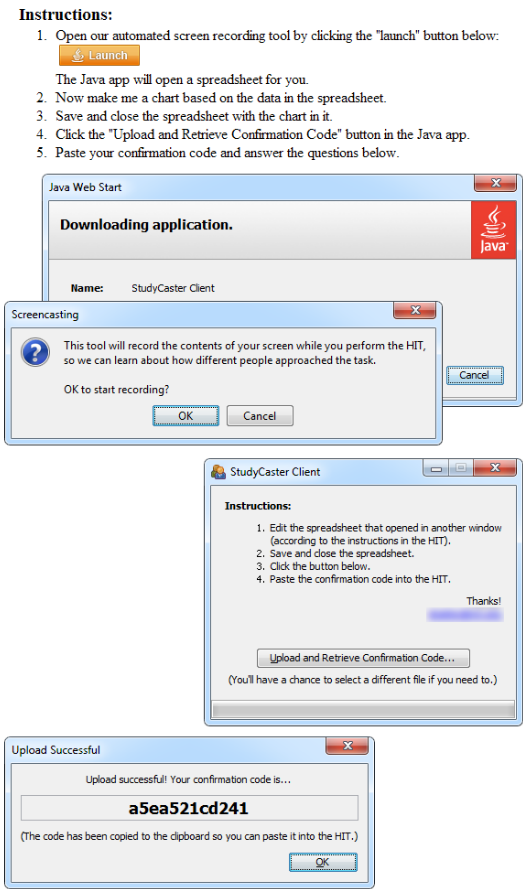
Assuming the system under test is a web site or a piece of software that can be run on a subject's computer, a major limitation of online user studies lies in the difficulty of actually observing the actions of the subject as he or she solves the tasks presented. From a platform like Mechanical Turk, the only experimental data retrieved automatically is the answer to questions given in simple HTML forms, as well as the time elapsed from when the user first accepts a posted task until he or she submits an answer. Other data, such as features used in the application under test, more accurate timing information, or information about the particular approach taken by the subject to solve a task, must be gathered through separate channels. This could mean instrumenting the software under test, instrumenting a survey form (e.g. with JavaScript code), or developing additional survey questions that address these issues and hoping that the subject responds faithfully. In either case, both collection and interpretation of data require a significant effort, and the results may be far less informative than had the experimenter actually been watching over the shoulder of the subject doing the study.

[1]E.g. under "exempt" status according to US Federal guidelines 45 CFR Part 46.101(b)(2).
[2]`http://www.odesk.com`
[3]`http://www.mturk.com`

In this note, we present StudyCaster, a Java-based tool that allows user study subjects to stream complete recordings of their computer screens to an experimenter's server with a minimum of effort. Our primary contribution is an integrated anonymization feature that blurs any window on the screen that does not match predefined criteria. Because data is uploaded asynchronously as the subject is performing the study, screen recordings can be of arbitrary length, and the tool is designed to be robust against temporary network failure, firewall limitations, and unexpected errors. Other features include the ability to download and open files (e.g. a PDF document or an Excel spreadsheet) with the default associated program on the subject's computer, and the ability to let the subject upload a modified or new file from his or her computer. The StudyCaster is launched from a Java Web Start link that can be embedded in any HTML page or email message.

RELATED WORK

Existing work has discussed both topics of demographics and evaluation techniques for Mechanical Turk-based studies [4, 5]. A number of recent startups offer remote usability testing services, including Loop11[4], Usabilla[5], OpenHallway[6], and UserTesting[7]. StudyCaster is a free open source alternative to these, particularly designed for researchers. It provides the additional advantage of anonymization. StudyCaster's anonymization is similar to the screen obscuring features of RoleBased Views[2] and PrivateBits[3]. These systems are not designed to work for remote users, however, and can thus not be used for an end-to-end anonymous online user study.



Instructions:

1. Open our automated screen recording tool by clicking the "launch" button below:

   Launch

   The Java app will open a spreadsheet for you.
2. Now make me a chart based on the data in the spreadsheet.
3. Save and close the spreadsheet with the chart in it.
4. Click the "Upload and Retrieve Confirmation Code" button in the Java app.
5. Paste your confirmation code and answer the questions below.

Java Web Start

Downloading application.

Name: StudyCaster Client

Screencasting

This tool will record the contents of your screen while you perform the HIT, so we can learn about how different people approached the task.

OK to start recording?

StudyCaster Client

Instructions:

1. Edit the spreadsheet that opened in another window (according to the instructions in the HIT).
2. Save and close the spreadsheet.
3. Click the button below.
4. Paste the confirmation code into the HIT.

Thanks!

Upload and Retrieve Confirmation Code...

(You'll have a chance to select a different file if you need to.)

Upload Successful

Upload successful! Your confirmation code is...

a5ea521cd241

(The code has been copied to the clipboard so you can paste it into the HIT.)

Figure 2. A StudyCaster-based user study, from the subject's point of view.

THE SUBJECT'S PERSPECTIVE

We first explain the operation of the StudyCaster tool as seen from the user study subject's perspective. As an example, we use the recruiting phase of a study published in CHI 2011[1]. In this study, which was conducted on Mechanical Turk, subjects were asked to edit a provided Excel spreadsheet, save their changes, and upload the modified file, all while running the StudyCaster client application.

The subject's interaction with the StudyCaster client is shown in Figure 2. In this example the user study is published as a Human Intelligence Task (HIT) on Mechanical Turk. When the subject clicks the Launch button in the HIT, Java Web Start will load the StudyCaster client application, possibly after a Do you want to open this file? question from the web browser, and after a security warning from Java. The main StudyCaster client window will appear in the lower right-hand corner of the subject's screen, with a centered modal dialog box asking for informed consent to record the user's screen during the study. While the system will be blurring the recording of windows unrelated to the study task, this is not mentioned in the informed consent message, as we would still like subjects to think about their own privacy during the study as an additional safeguard against accidental exposure of identifiable information.

To ensure that subjects run the StudyCaster client before starting work on their task, the client can be made to open a link or document that the subject needs access to. In the example study, the StudyCaster client would automatically download and open an Excel spreadsheet for the subject to work on. This also allowed the experimenter to limit study participation to subjects who had Microsoft Excel installed on their machines. In other experiments, the document opened could be a web page, a PDF file, or a packaged desktop application (not necessarily Java-based) under test. The document opened could also depend on a parameter in the Java Web Start link on the HIT page.

As the subject works on the tasks presented in the user study, the StudyCaster client records the contents of the screen. The screen recording thread automatically limits its frame rate such that its CPU utilization never exceeds 70%, and never exceeds 5 frames per second. During the study, the StudyCaster client window remains always-on-top in the corner of the screen unless moved or minimized by the subject. Depending on the experiment, the window may be used to pro-

[4]`http://www.loop11.com`
[5]`http://usabilla.com`
[6]`http://www.openhallway.com`
[7]`http://www.usertesting.com`

vide instructions that the subject may need to refer back to while completing a task.

After completing the tasks, the subject presses the Upload and Retrieve Confirmation Code button. In the example experiment, this prompts the subject to upload either a modified version of the originally opened spreadsheet file, or a different file. Other experiments might disable this feature, or use it for different purposes. Next, the client will show a progress bar as it finishes transferring whatever screen recording data is still in the pending upload buffer. Since recording data is streamed continously to the server for the entire duration of the study, the wait for the final buffer flush is usually only a few seconds, and as another feature of the automatic frame rate limiting, there will never be more than 4 megabytes of data left to upload at the end. Finally, the user is shown a confirmation code, which can be pasted into the appropriate HIT form field.

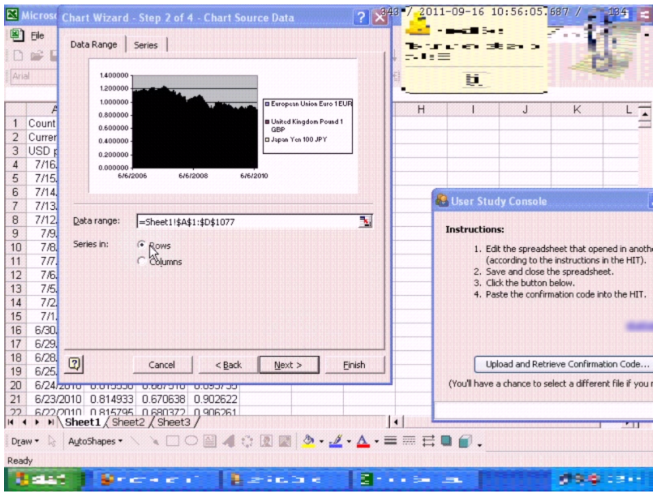
In this example, the StudyCaster client was launched from a Mechanical Turk HIT. The launch button could just as easily have been embedded in any HTML page, or subjects recruited through other means could be sent an email with a static link to the StudyCaster client.

THE EXPERIMENTER'S PERSPECTIVE

The StudyCaster system, once built with appropriate customizations for a particular experiment, is distributed as a Web Archive (WAR) file which may be deployed to any Java-based web server. Additionally a MySQL or other database is required for logging purposes and a local writable directory to store and access screen recordings. For a hosted solution, Amazon Web Services[8] can be used for all of these requirements.

Once the StudyCaster server application is deployed either locally (for development) or on a public HTTP server, the experimenter can navigate to its URL, which will contain a password-protected status page. The status page includes an HTML template for a link or Java Web Start button that can be pasted into a user study instruction page, for instance in a Mechanical Turk HIT. From the status page, the experimenter can also generate a report that summarizes all current log entries in the database. The report shows a list of all remote invocations of the StudyCaster client application, their time and duration, and the total amount of screen recording data uploaded during the session. Also available are transcripts of each client invocation's Java console output, which is uploaded for debugging and data collection purposes. Finally, the report generator uses a combination of short IP address hashes and cookies stored on client computers to group multiple invocations of the client application by what is assumed to be the same subject into a single coherent timeline. This makes it possible to find all screen recordings made by a single subject even if, say, the StudyCaster client or the computer it was running on crashed and the subject had to reopen it to get a new random confirmation code. It is also useful in order to detect subjects that should be excluded from a study

[8]`http://aws.amazon.com`



Data range: =Sheet1!$A$1:$D$1077

Series in: [✓ Rows] Columns

Instructions:

1. Edit the spreadsheet that opened in anoth (according to the instructions in the HIT).
2. Save and close the spreadsheet.
3. Click the button below.
4. Paste the confirmation code into the HIT.

(You'll have a chance to select a different file if you r

Figure 3. Clippy has been censored (upper right). The main Excel and StudyCaster windows are left legible. Popup windows frequently contain personal information, such as instant messages or email notifications. (This subject has an unusually low screen resolution.)

due to the possibility of cheating, such as individual Mechanical Turk workers operating multiple accounts. Along with each subject is also shown their geographic country and region based on a lookup in a GeoIP database at the time of each request. At no point is an actual IP address stored in the database, as this could conceivably be used to identify the subject.

Screen recordings are stored as files in a server directory using a custom-designed codec[9]. Mouse movements are sampled independently and more frequently than full screenshots, giving the final screen recordings a smooth appearance. A command-line utility uses Xuggler[10] to convert the internal video format to more standard ones (though with poorer compression), with the option of increasing the speed of the output video by an integer multiple. In our experience, it is possible to do a meaningful review of a user study recording while viewing it at up to 15x the original speed. This is a great time saver for an experimenter scanning through dozens of recordings in search for observations. As an aid to experimenters wishing to make timing measurements based on screen recordings, each frame of the output video file is annotated with an input frame number, an absolute timestamp calibrated to the local time of the server at the beginning of the recording, and the number of seconds since the beginning of the recording, all independent of any speedup applied.

Screen recordings are anonymized before they are uploaded to the server. The anonymization works by blurring the contents of any area on the subject's screen that does not belong to a window created by a whitelisted process. A process is whitelisted if one of its visible windows' titles contain one of a set of strings predefined by the experimenter. For

[9]The codec is inspired by that used in the Java Remote Control project (`http://code.google.com/p/java-remote-control`).

[10]`http://www.xuggle.com/xuggler`

instance, in the spreadsheet study example, processes were whitelisted if they possessed a window with a title containing the strings ”Excel”, ”OpenOffice.org Calc”, or ”StudyCaster”. Thus a legible recording would be made of any Excel window or dialog box as well as the StudyCaster window itself, but any other parts of the screen would be blurred to make any regular-size text illegible. See Figures 3 and 1. The StudyCaster client also maintains a blacklist of window titles which should always trigger blurring regardless of parent process; this is mainly used to protect file chooser dialogs, which may sometimes expose the subject's local login name.

EXPERIENCES WITH THE TOOL

The first study to use our tool, referred above, involved short screen recordings made by 64 Mechanical Turk workers in its first phase, as well as longer sessions by 36 subsequent subjects, about 30 minutes in length on average, but occasionally as long as two hours. A second study, currently in progress, has involved recordings made by 159 distinct workers. We now report on some of the experiences we have gathered with the StudyCaster tool to this date.

We observe that while the blurring algorithm only makes text illegible up to about a 16 point font size, text above this size rarely contains personally identifiable information. For instance, it may be easy to see that a subject is browsing GMail, YouTube, or Mechanical Turk, but hard to make out names or smaller headings. This is still a limitation for users who might use larger-than-normal fonts, such as vision impaired users. Photos of people tend to remain clear even after blurring, especially in the case of desktop backgrounds, which fill the entire screen area. A proposed improvement would be to obscure the desktop background completely. The only disadvantage would be the inability to observe interaction with desktop icons and such.

Another class of anonymization issues deal with the selection of the region to blur. In the first version of our software, the region was always rectangular, based on a single window with a white-listed title or its active child windows. This was vulnerable to popup windows from email and instant messaging clients, which would appear unobscured on top of a whitelisted window. Another problem related to hardware graphics acceleration in Windows 7, where the clipping area of a whitelisted window would appear larger than its actual visible portion, creating a too big unobscured area on the screen. These problems are solved in the current version of the software, which uses higher-level information from the Win32 API to determine window location and Z-order, and which allows non-rectangular blurring areas. On non-Windows platforms, the screen is kept blurred in its entirety.

A more subtle bug in the determination of the blurring area results from the fact that on a sluggish computer, a ”ghost” image of a background window may remain in another window that has newly been promoted to the foreground before the latter has had time repaint itself. If a screenshot is taken at the wrong moment, an unobscured view of the old window may be captured as a whitelisted one is brought to the foreground. A related effect is even more prevalent in Windows 7, where windows may intentionally fade gradually in on top of existing ones. A solution to this problem could be to introduce an explicit delay in the algorithm that determines the blurring area.

A few usability problems were seen and corrected in the tool itself. In particular, confirmation dialogs were added for closing the StudyCaster window without retrieving a confirmation code first, and before concluding the study and retrieving a confirmation code. We also implemented a method to prevent multiple instances of the client from being run at the same time, with a warning to the user if attempted.

Finally, we added several internal features to increase the technical robustness of the tool, including the ability to retry and recover from failed server requests even after the internet connection goes down temporarily, after the subject's computer switches to a different network interface (e.g. from wireless to ethernet), or after the server has been restarted with a new software update.

CONCLUSION

We have presented StudyCaster, a Java-based tool that allows subjects to stream complete recordings of their computer screens to an experimenter's server with a minimum of effort. Our system includes the novel feature of anonymization, which blurs any window on the screen that does not match predefined criteria. A user study conducted using the tool revealed several potential improvements to the anonymization performance, several which have already been implemented.

REFERENCES

1. Eirik Bakke, David R. Karger, and Robert C. Miller. A spreadsheet-based user interface for managing plural relationships in structured data. In *Proceedings of the 29th International Conference on Human Factors in Computing Systems (CHI '11)*, New York, NY, USA, 2011. ACM.
2. Lior Berry, Lyn Bartram, and Kellogg S. Booth. Role-based control of shared application views. In *Proceedings of the 18th annual ACM symposium on User interface software and technology (UIST '05)*, pages 23–32, New York, NY, USA, 2005. ACM.
3. Kirstie Hawkey and Kori M. Inkpen. Privatebits: managing visual privacy in web browsers. In *Proceedings of Graphics Interface 2007 (GI '07)*, pages 215–223, New York, NY, USA, 2007. ACM.
4. Aniket Kittur, Ed H. Chi, and Bongwon Suh. Crowdsourcing user studies with mechanical turk. In *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems (CHI '08)*, pages 453–456, New York, NY, USA, 2008. ACM.
5. Joel Ross, Lilly Irani, M. Six Silberman, Andrew Zaldivar, and Bill Tomlinson. Who are the crowdworkers?: shifting demographics in mechanical turk. In *Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems (CHI EA '10)*, pages 2863–2872, New York, NY, USA, 2010. ACM.