What's your problem?

Programming is hard. It requires that developers not only learn a new language, with new rigorously enforced syntax and semantics, but for many individuals it requires a new way of thinking with a rigidity and internalised consistency that they might not be used to. With binary computers everything is, very literally, black and white, high or low, one or zero. How can we reduce the cognitive burden on these individuals and help them reach their goals? **Somebody think of the developers!**

Code is poetry

Writing software is a skill, but it is also an art. So how exactly does creativity, design, and presentation fit into the raw, digital environment of an IDE? Aesthetics have long been used as a key influencer of human behavior – the entire print and multimedia advertising industries are built around this very concept – so can we use the aesthetic properties of the text developers type to influence them to write better code, faster? Is there is more to it than mono-space fonts?

Empowering people

As understanding and control of digital technology becomes a basic requirement to survive the modern world, programming gains a higher profile. While platforms like the Raspberry Pi may bring flexible computing power to the masses, harnessing it remains a problem. For those that wish to take the plunge, coding must be easy to understand. How can code be made simpler and easier to understand, without changing the content itself?

He shoots! He scores?

The goal of this research is to improve the coding experience, not just of the experienced developer but also of those new to programming. By determining the presentational features that have measurable impact on developer performance and code perception, the aim is to use these features to inform the development of an assistive editor and improve the user experience of developers.

Good, fast, or cheap. Pick three.

Developers are expensive. And every one is unique. Attempting to discover general principles from a limited pool is difficult. Crowdsourcing has been shown to be a valuable way of gathering data for user studies quickly and cheaply. With carefully planned controls and objective filtering, crowdsourced studies have been show to be qualitatively and quantitatively equivalent to experiments with lab-based participants, making crowdsourcing a 'holy grail' for some types of user studies.

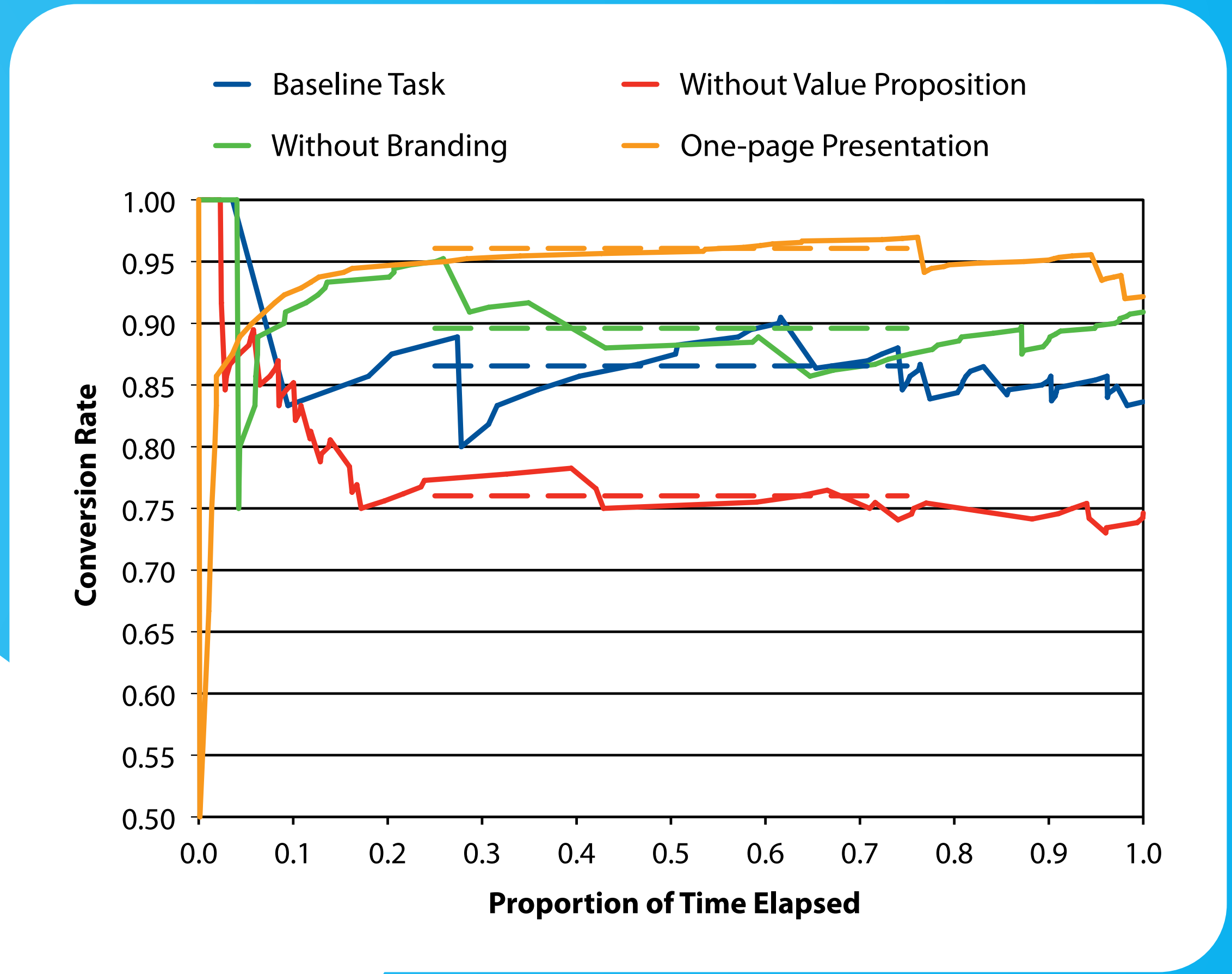
Why does the crowd work?

To better understand our potential participants, a more complete picture of the marketplace in which they work is required. In our long-term demographic study of crowd workers, comprising in excess of 3,000 participants, we have gathered both basic demographics and socio-economic status of US-based crowd workers in a way that is comparable to US labour statistics. This allows us to better understand crowd worker behaviour in the current economy and identify any causal relationship.

Making studies repeatable

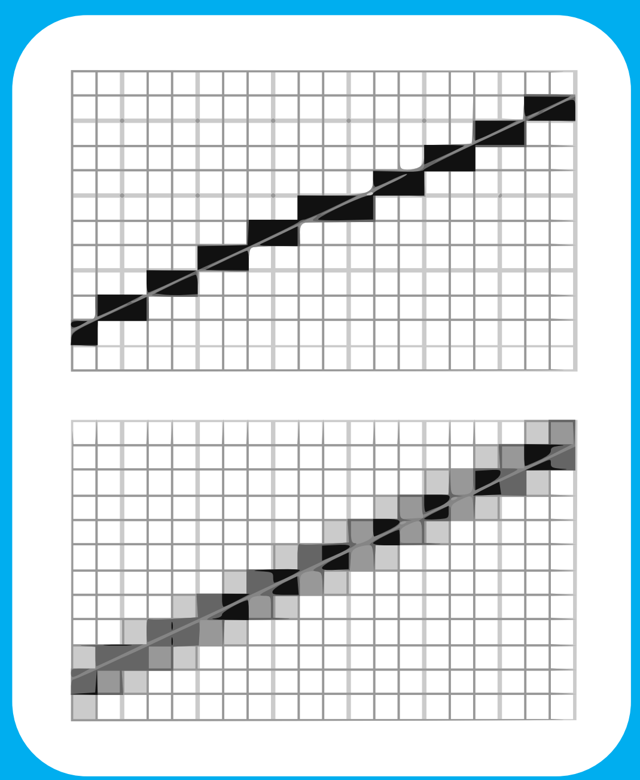
For any results to be taken seriously, data collection must be repeatable. Part of the problem with crowdsourced data is the huge potential variability in participants due to factors such as time zone and culture. What do we really know about how workers choose to undertake a task?

To improve understanding of the worker pool and allow a more comprehensive understanding of worker behaviour we proposed a new methodology and new metrics to allow data to be captured about those workers who preview a task, but choose *not* to accept. Our key metric, conversion rate, allows the evaluation of how many potential workers decided to undertake the task. This insight is essential when attempting A/B testing using the crowd: do workers really prefer A, or were they all just asleep when B was tested? The ability to dynamically determine the effective worker population also provides advantages for commercial tasks, allowing requesters to measure the effect of actions taken to manipulate speed, cost, and accuracy.



Smoothing over the cracks

Our first crowdsourced code perception study required participants to answer questions about a series of code snippets. Code was displayed to workers as bitmapped graphics, with and without anti-aliasing, alongside questions that were designed to require an understanding of the potential execution paths. The study indicated no statistically significant difference between code with and without anti-aliasing applied. While the result superficially appears to be of little interest, anti-aliasing has been repeatedly shown to be beneficial to both word recognition and general reading speed.



Illuminating the dark

Our second planned study investigates developer understanding of operator precedence. We intend to use our existing methodology to study the effect of operator luminance as a function of precedence on user performance when solving both simple and complex expressions.

This idea has been demonstrated as a proof-of-concept while the main study is under development, with a focus on defining suitable test cases.

```
int x = 5 + 7 >> 4 - 1 * 2 & 6;
```

Colour as an influencer

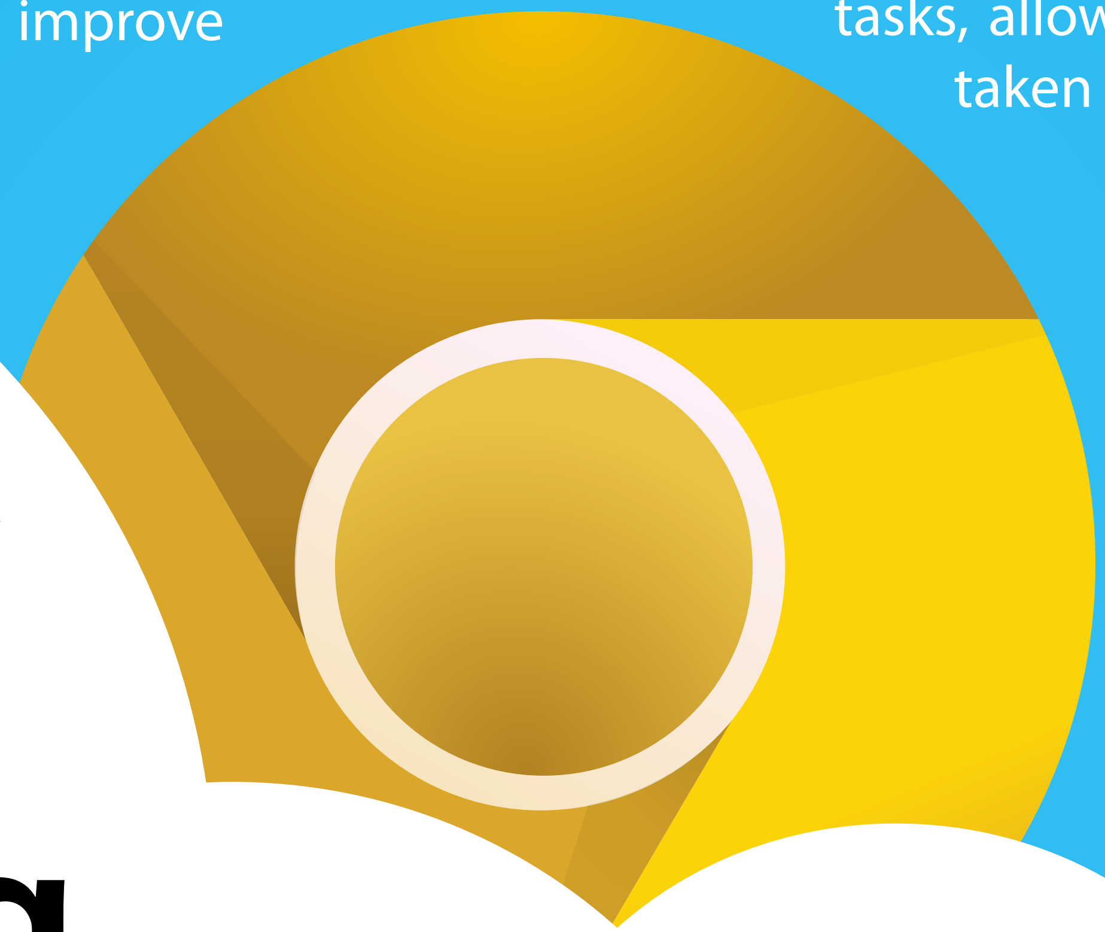
Syntax highlighting is a common feature of IDEs and editors. Our third code perception study will evaluate syntax highlighting and its effect on developer performance. Both providing a contrast to existing research on code perception and continuing with our existing evaluation methodology, this experiment will help ground our work.

This study is under development, with further evaluation of existing literature and careful selection of colour schemes and code snippets the current focus.

All work and no play...

In addition to the work carried out directly in pursuit of my PhD studies, to maintain a varied skill-set and bolster the profile of Computer Science at the University of St Andrews, I've contributed to Google's SPDY network protocol, and entered design completions for both the technology website Slashdot and CHI 2013.

Slashdot Logo
02/10/2012



Have an accepted contribution to Google's SPDY protocol!

Crowdsourcing & Code Perception

by Jason T. Jacques supervised by Per Ola Kristensson

jtj2@st-and.ac.uk www.cs.st-and.ac.uk/~jtj2/go/1

EPSRC
Pioneering research and skills

University of St Andrews
Scotland's first university

600 YEARS
1413 – 2013