Neil Satra

Sketching Charts

Computer Science Tripos, Part II

Pembroke College

May 4, 2014

Proforma

Name:	Neil Satra
College:	Pembroke College
Project Title:	Sketching Charts
Examination:	Computer Science Tripos, Part II, 2014
Word Count:	TODO
Project Originator:	Alan Blackwell, Neil Satra
Supervisor:	Alistair Beresford

Original Aims

This project aims to explore if users are able to create content faster, or experiment with it more, if given the tools to directly manipulate their creation. Specifically, it involved:

1. Building an application that lets users create graphical visualisations of their data by simply sketching their desired output, like they would on paper.

2. Evaluating the learnability of the interface, and how it compares to existing tools for creating charts, through a user study.

Work Completed

I have completed all core work items by building a Chart component for desktop applications that successfully runs sketch recognition on user input to determine what they intend to create, and then generates those chart graphics for them. I have also designed and built an interface that makes the learning curve for interacting with this component short and gentle, by applying HCI principles.

A few extensions were also completed, including allowing edits to the formal chart to flow back and manipulate the raw sketches.

Special Difficulties

None.

Declaration of Originality

I, Neil Satra of Pembroke College, being a candidate for Part II of the Computer Science Tripos, hereby declare that this dissertation and the work described in it are my own work, unaided except as may be specified below, and that the dissertation does not contain material that has already been used to any substantial extent for a comparable purpose.

Signed

Date

Contents

Chapter 1

Introduction

1.1 Motivation

This project is an exploration of Human Computer Interface concepts governing the interactions of users with tools that let them explore and visualise data.

The design of currently available charting tools were constrained by the input devices available previously: mouse and keyboard. Thus, they usually allow graph generation through one of two interfaces:

1. A series of dialog boxes and wizards to walk the user through a number of choices.

2. Writing code that is interpreted to process data and generate graphics.

Because these designs were constrained, they cannot offer certain benefits, such as easier learnability and direct manipulation.

1.2 Project Description

This paper describes a different interface, which allows the user to sketch a subset of a chart on their computer touch screen like they would on paper. The hypotheses are that:

H1 This interface is more 'learnable' over time

H2 It allows easier modification and exploration of visualisations compared to other charting applications.

Both these hypotheses were investigated through a user study.

The end result is a proof-of-concept charting application that works as below:

1. The user imports data from a Microsoft Excel file.
2. They sketch a rough indication of a chart.
3. They drag the data onto elements of the chart to actually bind the data to the chart.
4. The tool then creates a 'formal' chart.
5. The tool transforms the user's original sketch to more closely match the formal chart, making the mapping between sketch and formal chart elements evident to the user.
6. Any changes on either the sketch or formal chart is fed through to the other view. For example, erasing the a sketched bar removes a data series from the formal bar.

1.3 Background and Related Work

Sketching inputs have been studied since the 1960s [Sutherland, 1964] as more natural interfaces to computers, especially for graphics-related tasks. This has largely been motivated by the widely recognised importance of interaction to Information Visualisation (InfoVis) [Lee and Isenberg, 2012]. Additionally, the metaphor of sketching on paper can encourage exploratory work due to the ease of creating changes and visually expressing what sort of change one is trying to make [**?**], minimising the gap between a person's intent and the execution of the intent.

Meanwhile, there has been increasing adoption of touch-enabled phones and multi-touch slates amongst the general public, demonstrating people's affection for what have been referred to as Natural User Interfaces Lee and Isenberg [2012].

Chapter 2

Preparation

This project involved vast exploratory design work, in preparation of the actual implementation.

2.1 Requirements

This project could go in numerous different design directions from the start. Since the functionality and its benefits over existing systems depended heavily on the design chosen, it was hard to separate what benefits the program would achieve from what features the program would include and how it would expose them to the user.

However, in order to focus our exploration and make sure that the focus was on achieving some real deliverables for end users, a requirement analysis was necessary. The following functional goals were listed based on the project proposal, which focus on what basic tasks the system must help the user achieve, while allowing leeway in how the program exposed and implemented these features.

The system must allow the user to:

Core 1: Use any data they have in reasonably arranged formats in common file types.

Core 2: Specify the type of chart they want by drawing a likeness of it on screen using a stylus.

Core 3: Bind the data to the chart using an interface that makes it clear how the data is affecting the visualisation.

Core 4: Specify visual, size and positioning properties of the chart through the sketches.

Core 5: Manipulate the visual appearance of the created chart.

In addition, time permitting, the system may:

Extension 1: Transform user-drawn sketches to show the visual link to the formal chart elements.

Extension 2: Feed back manipulations applied to the formal layer back to the user drawn sketches, in order to keep the visuals of the formal and sketch layers in synchronisation.

Extension 3: Allow users to undo actions by erasing sketches, and remove the corresponding formal elements without throwing errors.

Extension 4: Allow users to manipulate any property of chart elements, not just one, so that the domain of visualisations they can create is infinite. For example, allow them to bind not just the height of bars in a bar chart to data, but also their width and colour.

Extension 5: Analyse the data and infer properties that may allow it to automatically suggest properties of the chart, such as which field belongs on which axis, or whether a data series should be log scale or linear scale.

Extension 6: The user must be able to export the chart as a Microsoft Chart object that can be embedded as a dynamic object in Microsoft Office files, not just as a raster image.

The core of this project focusses on making more usable software, rather than providing additional functionality, compared to existing systems. Hence, some usability goals were also specified:

Usability 1: Users must be able to create charts at least as quickly as they can using current systems.

Usability 2: Users must be able to build a mental model of the software's behaviour within 2 uses of it. They should thus be able to accurately predict the consequences of any action taken within the software.

Usability 3: Changes to the information visualisation must occur through directly manipulating the visual representation of the chart, rather than through disconnected User Interface widgets.

Usability 4: The user must be able to easily try out changes to the visualisation, see what that would look like, and undo them if needed.

2.2 Work Items

An iterative development process, similar to the Spiral Model was adopted for this project. This allowed early experimentation with and user testing of the various components and different interface designs. The following broad work items were identified as necessary to achieve the objectives above:

1. Assess the various methods to build a classifier for ink recognition, including using the RATA Framework
2. Run an initial user study to see how people naturally draw graphs, and also use it to collect training examples for the classifier.
3. Build a UI that accepts strokes, runs them through the classifier, and shows the user feedback to indicate successful recognition.
4. Build the UI widget that lets users import their spreadsheets in Microsoft Excel or Comma Separated Value files. It must then expose the various fields detected.
5. Build the charting component to convert the recognised sketches and the ink into a finished visualisation.
6. Run a pilot study followed by a user study to evaluate the system.

2.3 Building the classifier

Core to the system is an ink recognition component that identifies the chart element (e.g. bar, axis) that the user has sketched. This must work above a certain accuracy threshold or the system will prove frustrating to users [Frankish et al., 1995]. However, since the project scope included other components too, the time available to build this classifier was limited. Hence,

we decided to build a classifier using existing tools rather than implementing one from scratch.

2.3.1 Tools used

Microsoft provides a digital ink library that includes an ink stroke recognizer. There are a number of other ink recognition tools available, such as $1, Rubine, PaleoSketch and CALI. However, while these recognizers work well for the recognition tasks they were designed for, the RATA framework is explicitly designed to allow generation of a custom recogniser for a new domain using a few example images [Chang et al., 2010]. It thus outperforms a number of these recognisers on data sets other than the examples they ship with. By choosing to use RATA, we ensured we got a high accuracy rate for a domain of chart elements that may need to expand over time to incorporate more types of charts.

Rather than using manually specified features and thresholds for these features, RATA uses machine learning to automatically select features and identify relationships between them. It uses the WEKA tool, which provides numerous different data mining algorithms, and combines the results of these algorithms to provide higher accuracy. The authors of RATA also provide a 'Data Manager' tool to facilitate easy collection and labelling of training data.

2.3.2 Data collection

After acquiring RATA, we inspected the code and did manual testing to find some blocking bugs. Since we were in contact with the authors of the software, including Beryl Plimmer who until recently worked at the University of Cambridge Computer Lab, we were able to confirm with them that these were indeed bugs. We implemented fixes for them and contributed them back to the authors, and are working towards getting the code ready for to be published Open Source.

With a working version of RATA, an initial study was run to collect training data. We asked 10 participants (20-26 year old Cambridge students, studying a large variety of subjects) to draw 2 charts each.

First, they were shown a spreadsheet file containing sample data as below:

Year	Births	Deaths
2010	50,000	30,000
2011	40,000	35,000
2012	36,000	37,000
2013	34,000	35,000

Next, I spoke out the following task prompt.

> Say you are a government official trying to visualise trends in birth and death rates over time. Could you draw a bar hart on this screen that you would need to visualise this. No need to plot the actual data, just a rough sketch of what this chart would look like.

They were asked to draw the same chart 2 times, in order to get 20 training samples in total, and to observe how much variation there is between multiple sketches by the same user.

2.4 Design

2.5 Planning

Chapter 3

Implementation

3.1 Overview

At a high level, the program is composed of 3 components - data handling, sketch processing and charting (in increasing order of complexity).

3.2 Data import and management

Since the application is targeted at the average user, their data is most likely to be stored in spreadsheet format. Thus, it is important to allow them to import data from .xlsx and .csv files. For the sake of simplicity, the code assumes that the data is well-formed. Specifically, it works on the following assumptions:

1. The data is arranged as records in the rows of the spreadsheet.
2. The first row contains the names of the various fields.
3. No data is missing (if there are m columns and n rows, there are $m \cdot n$ data values.

Under these assumptions, import tabular data is a common use case, so I studied a number of existing libraries and methods to do this in C# and ultimately settled on

3.3 Sketch Processing Workflow

3.4 Charting

Bibliography

SHH Chang, Beryl Plimmer, and Rachel Blagojevic. Rata. ssr: Data mining for pertinent stroke recognizers. *Proceedings of the Seventh . . .*, 2010. URL `http://dl.acm.org/citation.cfm?id=1923380`.

Clive Frankish, Richard Hull, and Pam Morgan. Recognition accuracy and user acceptance of pen interfaces. *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '95*, pages 503–510, 1995. doi: 10.1145/223904.223972. URL `http://portal.acm.org/citation.cfm?doid=223904.223972`.

Bongshin Lee and Petra Isenberg. Beyond mouse and keyboard: Expanding design considerations for information visualization interactions. *Visualization and . . .*, (October), 2012. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6327275`.

I. E. Sutherland. Sketchpad a Man-Machine Graphical Communication System, 1964.