SPECIAL ISSUE

7th SA COMPUTER RESEARCH SYMPOSIUM
The South African Computer Journal
An official publication of the South African Computer Society and the South African Institute of Computer Scientists

Die Suid-Afrikaanse Rekenaartydskrif
'n Amplele publikasie van die Suid-Afrikaanse Rekenaarvereniging en die Suid-Afrikaanse Instituut vir Rekenaarweënskaplikes

Editor
Professor Derrick G Kourie
Department of Computer Science
University of Pretoria
Hatfield 0083
Email: dkourie@dos-lan.cs.up.ac.za

Subeditor: Information Systems
Prof John Schochot
University of the Witwatersrand
Private Bag 3
WITS 2050
Email: 035ebrs@witsvma.wits.ac.za

Productio.. Editor
Prof G de V Smit
Department of Computer Science
University of Cape Town
Rondebosch 7700
Email: gds@cs.uct.ac.za

Editorial Board
Professor Gerhard Barth
Director: German AI Research Institute

Professor Judy Bishop
University of Pretoria

Professor Donald Cowan
University of Waterloo

Professor Jürg Gutknecht
ETH, Zürich

Professor Pieter Kritzinger
University of Cape Town

Professor F H Lochovsky
University of Toronto

Professor Stephen R Schach
Vanderbilt University

Professor S H von Solms
Rand Afrikaanse Universiteit

Subscriptions
Annual Single copy
Southern Africa: R45,00 R15,00
Elsewhere $45,00 $15,00
to be sent to:
Computer Society of South Africa
Box 1714 Halfway House 1685
PROCEEDINGS

Guest Editor: Judy M Bishop

Organised by the SA Institute of Computer Scientists
in association with the Computer Society of SA
Sponsored by Persetel and the FRD
When the first SA Computer Symposium was held at the CSIR in the early eighties, it was unique. There was no other forum at the time for the presentation of research in computer science. In the intervening decade, conferences, symposia and workshops have sprung up in response to demand, and now there are several successful ventures, some into their third or fourth iteration. Each of these addresses a specific topic - for example, hypermedia, expert systems, parallel processing or formal aspects of computing - and attracts a specialised audience, well versed in the subject and eager to learn more. For the main part, the proceedings are informal, and certainly not archival.

SACRS, though, is still unique, in that it deliberately covers a broad spectrum of research in computing, and in addition, seeks to provide a lasting record of the proceedings. To achieve the second aim, we negotiated with the SA Institute of Computer Scientists for the proceedings to form a special issue of the SA Computer Journal, and the copy you have in front of you is the result. The collaboration between the symposium committee and the journal’s editorial board placed high standards on the refereeing and final presentation of the papers, to the symposium’s benefit, while we were still able to maintain a fresh, audience-oriented approach to the selection of papers.

This is SACJ’s first such special issue, and the largest issue (at 145 pages) to date. We hope that it is only the beginning of future such collaborations.

In all 29 papers were received, all were refereed twice, and 19 were chosen for presentation by the programme committee. All the papers were thoroughly revised by the authors on the basis of the referee’s comments, and the committee’s suggestions aimed at making the material more accessible to a broadly-based audience. Papers had to be new, and not to have been presented elsewhere, a requirement that is still unusual within the SA conference round.

A third goal of SACRS has been to invite keynote speakers, usually from overseas. This year, we are fortunate to present Dr Vinton Cerf, the father of the Internet and a world-renown expert on computer networks. Although his paper is not available for this special issue, it will appear later in SACJ. Through the good offices of Professor Chris Brink of UCT, we also have three other speakers from Germany, Canada and the US adding interest to the event, and two of their papers appear in this issue.

The programme committee originally devised a theme for the symposium - "Computing in the New South Africa". We received several queries as to the meaning of this theme, but unfortunately few papers that addressed it directly. One prospective author went as far as to enquire whether computer research would survive in the new South Africa. Another felt that his work was definitely not in the theme, as it was genuine, old world, basic, theoretical science! Nevertheless, there are two papers that consider one of South Africa’s key issues, that of language. Others look at the success we have achieved in applying technology to mining, and the future of low-cost operating systems. In all, the mix of papers represents a balance between the theoretical and the practical, the past and the future, all firmly based in the computing of the present.

Organising the symposium has involved the hard work of several people, and I would like to thank in particular

- Derrick Kourie, my co-organiser, and the editor of SACJ for his invaluable advice and hard work throughout the planning and implementation stages;
- Riel Smit, the production editor, for attaining such a high standard in such a short time for so many papers;
- Gerrit Prinsloo and the staff at the CSSA for their efficient and quite delightfully unfussy organisation;
- Persetel for their very generous sponsorship of R25000, and Tim Schumann for taking a genuine interest in our events;
- the Foundation for Research Development for sponsoring Vint Cerf’s visit;
- and finally the Department of Computer Science of the University of Pretoria for providing the ideal working conditions for undertaking ventures of this kind, and especially Roelf van den Heever for his unfailing encouragement and support.

Judy M Bishop
Organising Chairman, SACRS 1992
Guest Editor, SACJ Special Issue
Referees

The journal draws on a wide range of referees. The following were involved in the refereeing of the papers selected for this special issue. Their role in certifying the papers and their contribution to enhancing the quality of papers is sincerely appreciated.

John Barrow  
Ronnie Becker  
Danie Behr  
Sonia Berman  
Liesbeth Botha  
Theo Bothma  
Chris Brink  
Peter Clayton  
Ian Cloete  
Antony Cooper  
Elise Ehlers  
Quintin Gee  
Andy Gravell  
Wendy Hall  
David Harel  
Scott Hazelhurst  
Derrick Kourie  
Willem Labuschagne  
Doug Laing  
Dave Levy  
Graham McLeod  
Hans Messerschmidt  
Deon Oosthuizen  
Ben Oosthuysen  
Neil Pendock  
D Petkov  
Martin Rennhakkamp  
Cees Roon  
Jan Roos  
John Shochoh  
Morris Sloman  
Riel Smit  
Pat Terry  
Walter van den Heever  
Lynn van der Vegte  
Herman Venter  
Herna Viktor

UNISA
University of Cape Town
University of Pretoria
University of Cape Town
University of Pretoria
University of Pretoria
University of Cape Town
Rhodes University
Stellenbosch University
CSIR
Rand Afrikaanse University
DALSIG
University of Southampton
University of Southampton
The Weizmann Institute of Science
University of the Witwatersrand
University of Pretoria
UNISA
ISM
University of Natal
University of Cape Town
University of the Orange Free State
University of Pretoria
University of Pretoria
University of the Witwatersrand
University of Natal
Stellenbosch University
University of Pretoria
University of Pretoria
University of the Witwatersrand
Imperial College, London
University of Cape Town
Rhodes University
University of Pretoria
University of Pretoria
University of Fort Hare
Stellenbosch University
Galileo: Experimenting with Graphical User Interfaces

R T Apteker* J M Bishop†

* Computer Science Department, University of the Witwatersrand, Johannesburg, 2050
ronnie@concave.cs.wits.ac.za
† Computer Science Department, University of Pretoria, Pretoria, 0002
jbishop@cs.up.ac.za

Abstract

Graphic User Interfaces (GUIs) are the key ingredient in the development of a new era of computer software. Pioneered at Xerox, launched at Apple, pursued by Microsoft, and challenged by a host of others, the age of the GUI has dawned. GUIs emphasise the logical relationship of things and let intuition play a larger role in decision making – intuition as opposed to bulky documentation. Efficiency, uniformity and ease of use are the driving factors that make GUIs worthwhile. GUIs are concerned with interfacing between a user and a computer and interface consistency is a premier design goal. Yet constructing a GUI is no trivial exercise, and user psychology plays a large role in the development of this new wave of human-computer interfaces.

This paper discusses a series of experiments in GUIs under the Galileo Project. Galileo began in 1989 with the aim of exploring the natural talent of students to incorporate GUIs in ordinary programs. From the results gathered, we can classify GUIs according to the support software and implementation effort they require. The paper concludes with some observations on the place of GUIs in the new South Africa.

Keywords: Graphical user interfaces, human-computer interaction

1 Introduction

"The world is complex, dynamic, multi-dimensional; the paper is static, flat. How are we to represent the rich visual world of experience and measurement on mere flatland? To envision information – and what bright and splendid visions can result – is to work at the intersection of image, word, number, art. The instruments are those of writing and typography, of managing large data sets and statistical analysis, of line and layout and colour."

Taken from Envisioning Information by Edward R. Tufte, 1990

Computer science has experienced many revolutions. Revolutions bring about change – often for the better. We have witnessed the development of high-level programming languages, database-management systems, open networks, the ever-expanding object-oriented worlds, and numerous advances in applications including word-processing, spreadsheets and computer-aided design. All these factors require a greater understanding of human-computer interaction.

As systems are developing, so is the need for more effective interaction between humans and computers. The need for graphic user interfaces is becoming more apparent, causing a revolution that demands new approaches to “developing software of a different sort” [Hix 1989].

The story of the graphic user interface (GUI) began at Xerox’s Palo Alto Research Center [Perry 1989]. Steve Jobs realised the greatness behind GUIs and enshrined it in the Apple Macintosh. Thereafter Bill Gates saw the potential behind GUIs which culminated in the creation of Windows 3.0. The trend began with the Star from Xerox, then the Macintosh from Apple and Windows from Microsoft, followed by Presentation Manager, X Windows, NewWave, Motif, DeskMate, and now Steve Jobs has returned with NextStep. GUIs have clearly made their mark, and the world will never look back [Murphy 1991].

"Along the way the entire world went GUI in one form or another, including such unlikely operating-system converts as Unix.” [Barron et al 1991]

The Macintosh’s GUI has captured our imagination, and although there are no official standards as yet, the GUI components encompassed in this computer are becoming accepted as the norm in this new era of intuitive user interfaces. In pre-GUI times a user would have to learn each type of interface pertaining to a particular application; the new wave of graphically orientated interfaces, has one goal: to make life easier. The uniform
approach to rendering information and user interaction provides for a computing *tour de force*.

GUIs should not be confused with graphic applications: using a GUI to control a computer has absolutely nothing to do with "something graphically orientated" [ibid]. The point is that a GUI is a graphic-based user interface that can be applied to any level in the development of "software of a different sort". The goals are simple: efficiency and intuition. Logical relationships are in; lengthy documentation is out. This paper discusses the state-of-the-art of graphic user interfaces. We address the questions: What constitutes a GUI? What is a GUI used for and when? What are their advantages and disadvantages? We then go on to describe, for the first time, the Galileo Project, an on-going student-based experiment in GUIs, which provides a framework for classifying and assessing the development effort a GUI might require. The paper concludes with an assessment of the validity of GUIs in the new South Africa, and a summary of the major GUIs found in today's user community.

2 State-of-the-art of GUIs

What are graphic user interfaces?

"Mice, windows, icons, and menus: these are the ingredients of computer interfaces designed to be easy to grasp, simplicity itself to use, and straightforward to describe. The mouse is a pointer. Windows divide up the screen. Icons symbolize applications, programs and data. Menus list choices of action." [Perry 1989]

Computer science is continually striving to enhance the effectiveness of people [Apple 1989; Hartson and Hix 1989; Sommerville 1985]. The key to real effectiveness as far as computers are concerned is usability by people other than computer professionals [Apple 1989; Hartson and Hix 1989]. Human-computer interaction is defined as communication between a human user and a computer system. The medium for this communication encompasses a vast number of ideas. A graphic user interface is concerned with human-computer interaction using graphics as a vehicle for communication.

According to Hartson and Hix [1989] there are at least two ways of describing the manner in which humans communicate with computers: the conversational world and the model world. Furthermore, these correspond to two general types of dialogue: sequential dialogue and asynchronous dialogue. In the conversational world, the user communicates with the computer by a sequence of commands (sequential dialogue). This paper is concerned with communication in the model world where communication uses a graphic user interface. This scenario involves a user manipulating visual objects. The "manipulation of objects" corresponds to the situation evident in the Apple Macintosh environment where a user initiates a dialogue with the computer by clicking the mouse on an icon thus causing an event to be invoked. The point is that through the use of a graphically oriented interface, all commands and parameters take on a graphic form.

Text is sequential whereas pictures provide the facility of random access. A user can concentrate on the particular picture that holds interest. A simple icon can be very effective – after all, a picture can paint a thousand words. Furthermore, Raeder [1985] states that pictures generally transfer information faster than text – both accessing and decoding are more efficient. However, there are some things that simply cannot be visually portrayed (for example, the exact quantity in bytes of disk space available). A GUI does not attempt to eliminate text totally but rather to substitute it with icons wherever possible. GUIs emphasise recognition, not recall. Learning by pictures is far more stimulating than having to read sentence after sentence.

"Graphics are not merely cosmetic. When they are clear and consistent, they contribute greatly to the ease of learning, communication and understanding. The success of graphics design is measured in terms of the user's satisfaction and success in understanding the interface." [Apple 1989]

A look at the differences between the conversational and the model world clearly illustrates the motivating factors behind the development of GUIs [Hartson and Hix 1989]. These differences are given in Figure 1.

<table>
<thead>
<tr>
<th>Conversational world</th>
<th>Model world</th>
</tr>
</thead>
<tbody>
<tr>
<td>• sequential dialogue</td>
<td>• asynchronous dialogue</td>
</tr>
<tr>
<td>• communication is via a sequence of commands</td>
<td>• communication is via the manipulation of visual objects</td>
</tr>
<tr>
<td>• communication is solely via the keyboard</td>
<td>• communication is extended to a mouse, light pen, etc</td>
</tr>
<tr>
<td>• information is given in the form of text</td>
<td>• information interchange includes icons, windows, menus</td>
</tr>
<tr>
<td>• remember-and-type</td>
<td>• see-and-point</td>
</tr>
<tr>
<td>• command-line user interface</td>
<td>• graphic user interface</td>
</tr>
</tbody>
</table>

Figure 1 Differences between two communication worlds
The sequential world, best described by DOS, involves memorising a complex command language. The user has to press exactly the right keys in exactly the right order. If a mistake is made, the computer responds with a terse message that is often either unintelligible or confusing. It is sometimes hard to tell that the human user is the master and the computer the servant. GUis attempt to create an environment where the user is in control.

GUI Construction

There are as yet no official standards to adhere to, although there are moves towards a standard, as discussed in Section 3. However, the GUI visible in the Apple Macintosh can be considered as a role model interface. Common GUI features [Murphy 1991] include:

- a pointing device – mouse
- windows for graphical display and grouping information
- icons
- a host of buttons, sliders, checkboxes and other widgets
- see-and-point menus.

Further GUI features pertain to visualising information. These include:

- context-sensitive menu options
- shrink and grow icons and menus
- open and close window buttons
- drag and click concept
- the clipboard
- the desktop.

Windows are perhaps the most important construct behind GUI technology. It was Alan Kay, pioneering Smalltalk-76’s programming environment, who devised a user-interface paradigm called “overlapping windows” [Ambler and Burnett 1989]. The fundamental aspects of this paradigm are an integral part of GUis. Multiple overlapping windows support [ibid]:

- displays associated with several user tasks being viewed simultaneously
- switching between tasks at the push of a button
- no information loss in the process of switching
- the economic usage of screen space.

Windows are an obvious means of adding functionality to a display screen. It has been argued [Perry 1989] that overlapping windows closely match users’ work patterns, since no one arranges papers on a physical desktop in neat horizontal and vertical rows. Furthermore, pull-down menus are an important aspect of overlapping windows. Combined with a pointing device, pull-down menus are now an integral part of GUis – there is no longer a need to memorise available options [ibid].

Microsoft adapted these conventions in Windows 3.0 (except that the Microsoft mouse has two buttons). GUis are definitely portraying refinement in their construction and appearance. As discussed at the end of this section, a universal approach is becoming more and more evident. For instance, a user can easily move between the Macintosh and Windows environments [Murphy 1991].

When a user becomes familiar with a scenario then sometimes a single keystroke may be the fastest way to achieve a desired action. This is the motivation for including keyboard alternatives (hot keys) in GUis. These alternatives are simply logical extensions to the GUI [Apple 1989]. User-friendliness may be the primary objective but one must not forget about efficiency. The inclusion of keystroke alternatives in GUis neatly supplements menus, providing both ease of use for novices and for the less experienced, and speed for those who can type faster than they can point to a menu and click on a selection [Perry 1989].

Feedback is an important issue to consider. For example, when making a choice from a menu the computer can beep so that the user is confident that he or she is not being ignored. With the same sentiments in mind, it is obvious that animation will be appreciated by the user. Animation to show that a request is being carried out can be most effective and makes the user feel in control (a very important psychological issue). Animation can be particularly effective in guiding the user through logical errors in software development [Grafton and Ichikawa 1985].

Furthermore, animation can be entertaining. If an action requires a considerable amount of time then the user should be informed of this. This can be done by simply telling the user to be patient and wait. The graphic user interface can show a picture of a clock (and perhaps display the word “busy” underneath the clock) to inform the user that he/she has not been ignored and that the computer will be open to commands as soon as possible. Again the psychology of the situation involves a user who wants to feel in control. A user does not want to be dictated to by a computer!

Before the advent of graphic user interfaces the visual appearance of the screen had not been of major importance with respect to various applications. It has been argued that the most important area of an interactive system is the user interface [Neelamkavil and Mullarney 1990]. One interesting facet of graphical interfaces is the difficulty involved in constructing icons (graphical objects). There has been research into an area concerned with separating graphics from application in the design of user interfaces [ibid]. The burden of graphics coding can be separated from the interface by using a graphics editor. We must not lose sight of the fact that graphic user interfaces are alternatives, not substitutes, for conversational-based interfaces.

There are numerous points that should not be overlooked when considering the construction of graphic
user interfaces – these refer to the psychology of the situation. In particular, graphic user interfaces:

- must not just serve as a more entertaining medium for communication;
- must be informative, not just impressive and dazzling;
- should provide an alternative form of communication, and not substitute for a prompt-driven interface;
- should be motivated by the concerns of the novice user;
- should strive to aim for an environment that encourages user-friendliness.

Besides abstraction and logic, GUIs also strive to create an environment that is consistent amongst applications. The Apple Macintosh is a staunch supporter of consistent interfacing. The Mac environment yields ease of learning and familiarity across applications [Apple 1989]. Stability is another issue that should be considered when constructing a GUI. Stable reference points build a user’s confidence, especially when coping with complex issues that computers handle with ease. The Mac provides visual stability by defining a finite set of actions across applications – even when a particular action is not available, it is not eliminated from the display but merely dimmed [ibid]. We emphasise that GUIs are not trying to be fancy. Too many things happening at once will make it more difficult to achieve a consistent look. A simple interface is a good interface!

Where are GUIs used?

GUIs provide for simple command availability.

“GUIs are proving that graphical computing metaphors aren’t only for graphic artists or desktop publishers.” [Sheldon et al 1991]

The majority of computer users are not professionals – novices will find that a GUI makes a computer much easier to use. The biggest problem for those who are computer experts is that we are so familiar with our machines that we forget how intimidating they are to the vast majority of users.

“If you want to discuss the capabilities of a GUI, you must consider all computer users, not just writers or number crunchers.” [ibid]

Traditional computing involves a user constantly learning and exploring new environments when switching from one application to another, a time consuming and often frustrating task. Furthermore, the purpose and the result of operations and functions within each application have to be learned as well.

The desktop is an abstraction of a physical concept. The GUI that is captured by the Apple Macintosh and its imitators is reflective of a human workstation – folders, trash can, links to the outside world, etc. Furthermore, when you buy office furniture you can choose a particular colour and pattern. The same is true for the Macintosh. A user can customise the desktop to suit personal taste. On the other hand, does a prompt-driven interface like DOS reflect anything found in the real world?

Figure 2 shows the desktop that is an integral part of the Apple Macintosh. In this example, the user is erasing a file by simply dragging it to the trash can. This illustrates the process of manipulating screen objects to invoke an action. The arrow underneath the file (to be erased) reflects the user’s hand – this arrow is controlled by a mouse.

Not only does the desktop emulate a human work area, but the documents visible on the desktop resemble their paper counterparts. Text can be depicted exactly as it will appear on a printed piece of paper – black text on a white background, with a variation of typefaces, sizes and styles [Chemicoff 1987].

GUIs advocate a uniform approach to visualising information and user interaction. Opponents of GUIs argue that the availability of multiple windows distracts form the task at hand. Time is spent arranging information on the screen and scrolling through individual windows [Shneiderman 1987]. Is something wrong here? No. Users will have to spend time familiarising themselves with the new window environment (GUI). Initially, windows and icons will appear fascinating, and when the initial exploring phase is over, the user community will reap the benefits of this new wave of human-computer interaction. Put quite simply by Apple

“People aren’t trying to use computers – they’re trying to get their jobs done.” [Apple 1989]

Task internationalisation is a sentimental issue that captures another aspect of GUIs. There has always been a language problem when it comes to computer applications. Icons provide a universal language – the language of pictures.

The Use of Colour

The developers of graphic user interfaces continue to strive for better interfaces. Colour can be used to stimulate users and to group entities in a logical manner. Colour has already become an integral component of many systems using windows. In particular, colour:

- can be an important aid to fast and accurate decision making [Smith 1988]
- enhances perception and cognition [Shneiderman 1987; Smith 1988]
- can be an important tool for program debugging [Smith 1988]
• emphasises the logical organisation of information [Shneiderman 1987]
• draws attention to warnings
• evokes more emotional reactions of joy, fear, or anger [ibid].

Although colour adds an exciting dimension to GUI construction there are some rules that must be adhered to. Using too many colours at once is inappropriate and can distract rather than encourage. Tufte [1990] states that some applications of colour applied to a display screen have made what should be a straightforward tool into something that looks like a grim parody of a video game.

The following simple guidelines provide some insight in correct colour usage. (These are not concrete and will change with time as users’ tastes change.) GUI construction (or any software development for that matter) should:

• use colour conservatively – novice programmers are often far too eager to brighten up their displays often resulting in counterproductive situations;
• limit the number of colours used at one time – four colours is effective [Shneiderman, 1987];
• use colour as a coding technique – an important issue of recognition;
• be consistent – use the same colour coding rules throughout one program.

Towards a standard for GUIs

Screen output using bit-mapped images introduces the problem of interactive delays due to the large amount of communication that is incurred, especially with distributed systems. X Windows provides a partial solution, as well as a move towards a graphic interface standard. X is a windowing system for bit-mapped graphic displays. The X protocol is a well-defined set of graphics commands that supports an open, vendor-independent model which permits physically independent application and display processing [Socarra et al 1991; Cypser 1991]. It is emphasised that X is a protocol for communication of bit-maps, it is does not however mandate a particular type of window manager. The window manager is the contributing factor towards any GUI standard. Two commercial window managers, which abide to the X protocol, are OPEN LOOK from AT&T and SUN, and Motif from the Open Software Foundation. Both these window managers are establishing user interface guidelines [Nye 1990].

Although X Windows can be considered as a step towards standardisation, one must not forget that it is a networking protocol based on a client/server relation in an effort to decrease interactive delays in the situation where the application is remote from the display. In these systems, a window server runs on the host computer and the client is the application program. The point is that
the quantity of information that passes between applications and window server can be considerably less than the number of pixels on the screen [Coulouris and Dollimore 1988]. In other words, an application need not be running on the same system that actually supports the display [X Windows].

The latest version of the X Window System is commonly referred to as X11 (Version 11). The X consortium is an association of major computer manufacturers who plan to support the X standard [ibid].

3 Why Graphic User Interfaces?

Advantages

The advantages are numerous: recognition instead of recall; see-and-point instead of remember-and-type; icons instead of text; graphics; windows; ...

"Consistent visual communication is very powerful in delivering complex messages and opportunities simply, subtly and directly."

[Apple 1989]

For the novice user GUIs are a definite plus. However, the typical PC or Unix user may have to go through some mid-computer-life crisis. The fact is, that people are far more stimulated by graphics than by text and stimulation yields productivity. This all amounts to greater user effectiveness. Even traditional languages such as Pascal are being steered towards this exciting domain. Turbo Pascal now has an added facilities for windows programming [Linderholm 1991]. Now even a staunch advocate of the sequential world like Turbo Pascal can run in the exciting world of windows.

A GUI depicts a computer that operates via graphical messages – metaphors from the real world. For example, why should duplicating a file involve what can be described as a parody of a physics formula? Using a computer that has a well-crafted GUI allows a user to be effective and creative. Computers are meant to make life easier. GUIs help enforce this objective.

Disadvantages

The Apple Macintosh is a computer that includes a GUI as an integral part of the machine. Together, the computer and GUI form a machine of incredible power and ease of use. Arguing about its pitfalls would be to argue about its GUI. The success of the Mac captures its advantages – to harp on the disadvantages of its GUI would be to ridicule the machine all together, a rather futile thing to do.

On the other hand, there are those users, the vast majority in fact (due to the time factor), who are traditional PC, Unix or mainframe users. Perhaps they look with envy upon the new breed of computer users. Although the GUI environment visible on the Macintosh is now available to PC users in the form of Windows 3.0, and to Unix user in the form of X Windows, there are disadvantages. Both groups not only have to purchase their GUI software, they may also have to expand their hardware to take full advantage of all available features. Resource intensiveness is one of the more serious drawbacks of current GUIs. In time, however, this hurdle should be overcome.

Devoted PC users may have some difficulty adjusting to Windows since DOS operates by receiving a steady stream of ASCII characters rather than a bit-mapped image. BitBlt software, inspired by Smalltalk developers, made the bit-block transfer procedure possible [Perry, 1989]. Bit-mapped images make it easier to write programs to scroll a window, resize it, and drag windows from one location to another. Once again, resources can be a problem, since bit images consume considerable amounts of memory or disk space. In turn, it can be very slow to transmit these large bit images to the screen, leading to frustration and uncertainty on the part of the user.

Finally, the newer GUIs are not an integral part of the operating system (as is the case in the Macintosh), and rely on message passing between different layers of software. Developing applications is still error-prone in this environment, and "unrecoverable application error" is still a common problem for Windows 3.0 users.

Current Opinions

Abstraction has always been a concern of computer scientists. The GUI on the Apple Macintosh provides the user with an abstract view of the traditional PC computing environment – the desktop. The average user knows very little about the operating system of the computer and DOS has always been a scary concept for the majority of PC users. It is interesting to note that the Macintosh provides a GUI that seeks not to protect us from the rigours of the operating system – it is the operating system [Murphy 1991].

Why are DOS users apprehensive about GUIs? A big factor is that those users that are seriously involved with their DOS-like interfaces have fingers that are conditioned to type in certain phrases. Habits are hard to break!

Opponents of GUIs argue that text is streamlike, it flows; graphics on the other hand is chunklike. One user, on using a word processor, said,

"I type like the wind: give me Emacs and to hell with menus, buttons, and the mouse." [Sheldon et al 1991]

Sheldon [1991] states that we need both GUIs and prompt-driven interfaces. The fact that a user can pick up almost any Macintosh application without documentation is truly fantastic. But once you know the score, it
can be rather painful to be forced to point-and-click and point-and-click and point-and-click ....

"It's fun to lasso a bunch of files in Windows' file manager and then delete them. It's fun once. But after that, give me del *.obj. More important, give me ways to say del*.obj when date < value, every Thursday night at 11." [ibid]

The relationship of objects in a graphic user interface is of major consequence. Relationships must be logical and the interface easy to grasp. It is obvious that if a user interface (graphical or textual) is poorly designed it is liable to be rejected, irrespective of the facilities which it offers. An interface which has a poor design can allow the user to make potentially catastrophic errors which definitely contributes to that interface's acceptance [Sommerville 1985]. In this respect, the development of object-oriented design and programming has become integral to many GUI systems, for example Turbo Vision, which we mention in the next section.

4 The Galileo Project

Background

The Galileo Project aims to monitor and assess the extent to which exposure to GUIs fosters their emulation. Issues that Galileo considers are the quality of GUIs, level of graphics used, availability and cost of software and the progression of GUI impact over time.

The Galileo Project began at the University of Southampton in 1989, moved to the University of the Witwatersrand in 1991, and is now based at the University of Pretoria. This paper presents a summary of preliminary results: fuller results are in preparation.

Student assignments are an ideal place to investigate trends in GUIs. Students come into first year with a vast variety of backgrounds. Some will have been programming and using computers for ten years, some will never have used a computer. (This is as true in the UK as SA, though the proportions are different.) More importantly, some students will have been familiar with advanced user interfaces through using Archimedes, Macintosh or Amiga computers, whereas others will have been tied into DOS from an early age.

Phase 1 – Discovery

The first phase of the study took place with classes of 50 first year students in the UK in October 1989 and October 1990. The students were enrolled for a Principles of Programming course which concentrated on the use of graphical algorithms as a basis for learning to program recursively, and for appreciating the complexity of algorithms. The language used was Turbo Pascal 5.0. In each course, the students were set two assignments which each involved programming two versions of an algorithm couched in real world terms.

For example, "The Sandwich Express" was a system for finding the best route for delivering sandwiches around the campus, depending on the day's orders. The travelling salesman algorithm had to be programmed in its optimal but unreasonable O(n!) form (which limited the orders to something like 7) and a non-optimal but reasonable version which was O(n^2) (and could handle orders from the whole campus). The students were asked to show on the screen the paths generated by both algorithms, and to compare their performance.

Other assignments used the convex hull algorithm, the minimal spanning tree, and the minimal polygon distance. The ideas for the assignments were novel, but were based on the exciting treatment of algorithms in Harel [1987].

When the first batch of assignments came in, we were astonished at the quality of GUIs produced by a group of the students. Their use of windows, menus and colour was excellent, and represented a wide variety of styles. Specifically, we found the following:

- a preference for dividing the screen into windows which were continuously resident, though some students used a hierarchy of overlaid windows quite effectively
- good use of colour for backgrounds (cyan, light grey) to distinguish windows
- a mature use of fonts (i.e. sticking to one)
- a help facility as standard
- cursor driven menu choice selection
- function keys as built-in alternatives to menu selections.

Considering that the students were programming in Pascal from scratch (that is, without Turbo Vision) and had a bare three to five weeks for the project, the achievement was remarkable.

When the second assignment was turned in by the same group of students, the reason why they were able achieve such a high standard became obvious: re-use. With small amendments, the students who had presented a GUI for the first assignment, presented the same GUI for the second! In discussions with the students, it became clear that in some cases, their GUIs had been developed at school, and had simply come with them to university.

Phase 2 – GUIs to the fore

A student who developed a GUI that he or she was comfortable with, and which interfaced easily to the programming language being used for assignments, was clearly at an advantage in the class. It was therefore decided in 1990 to devote some time to actually teaching students about GUIs and about the basic Pascal tools they would need to create one. Furthermore, tutorial
time was devoted towards discussing alternative GUIs in
groups and stressing that good ideas could and should be
copied and re-used.

The effective was dramatic. Compared to the first
year, far more students took the plunge and went for a
full colour graphical user interface. However, there was
a downside: some students became entangled in the GUI
development to the detriment of the algorithms they were
actually meant to be studying, and obtained poor marks
as a result.

Intermediate Results

The first output from these two phases was a suite of
some 16 demonstration programs, written by the
students, which could be used to give further intakes a
direct exposure to GUIs. The impact of telling students
that these programs were written by first years was
considerable. They were both encouraged and
challenged to emulate them. However, it was recognised
from the start that not all students would be able to
achieve the mastery over the additional Turbo Pascal
facilities which would be required to emulate these
examples. We therefore devised the following
classification of Turbo features as a guide to the amount
of effort required (Figure 3). Students were then told to
aim for a particular level, and stay there.

Interestingly, half the class then elected to stay at
Level 2, using only the Crt Unit and very basic windows
and cursor control. Nevertheless, the GUIs they
produced were easily better than the line-driven,
scrolling, DOS-oriented versions of yore. About a
quarter of the class used Level 3, which entailed doing
all text input-output with graphics procedures, but which
had the advantage that the screen could be regarded
almost as a desktop, with multiple windows. There were
also students who mixed levels and displayed their text
for menus and results via the Crt unit, while using the
line and shape drawing facilities of the Graph unit for
animating the algorithms. The disadvantage of this
approach is that the screens were continually swopping,
but the learning curve was not as steep.

Phase 3 – adapting to minimal facilities

In Figure 3, the Turbo Pascal features needed for each
level are listed. Turbo Pascal 5.5 and earlier only
supports the first three levels. Level 4 is available with
Turbo 6.0 onwards and level 5 has to be purchased as a
separate package.

Phase 3 of the project involved investigating the
effect of limiting students to the lower levels. In
September 1991, 100 first years at Wits were involved in
a similar course. This time, GUIs were strongly
emphasised right from the very first tutorial, and actual
instruction was given in their construction, with practice
sessions in the laboratories [Bishop 1992]. However, the
computer facilities were such that only level 2 could be
used in the laboratories (Turbo 4.0 with non-graphics
screens).

Nevertheless, the students took to the idea straight
away and in class assessment feedback rated “the use of
windows” and “controlling the screen” as the most
exciting aspects of an otherwise traditional Data
Structures course. The message here was that GUIs
should be introduced no matter how primitive the
facilities, and that the management concepts (menus,
help, uniformity) are actually independent of any
graphical whizz-bangery.

At the same time, an independent study was
conducted on the design of a GUI specifically for general
algorithm animation (AL). Although Turbo 6 was used,
it was decided to restrict usage to level 3. The resulting
GUI was evaluated by a set of test users and the results
are contained in Apteker [1991]. Of note are the
following:

<table>
<thead>
<tr>
<th>Level</th>
<th>Title</th>
<th>Features needed</th>
<th>Effort to Master</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Text</td>
<td>none</td>
<td>none</td>
<td>Traditional Pascal programming, simple and 90% portable</td>
</tr>
<tr>
<td>2</td>
<td>Screen</td>
<td>Crt unit</td>
<td>days</td>
<td>Character based cursor-control plus colour</td>
</tr>
<tr>
<td>3</td>
<td>Graphics</td>
<td>Graph unit</td>
<td>2-3 weeks</td>
<td>Line and pixel graphics, colour and shading, text i/o via special procedures</td>
</tr>
<tr>
<td>4</td>
<td>Object</td>
<td>Turbo Vision</td>
<td>3-4 months</td>
<td>Object-oriented, event-driven programming with full pull-down menus, overlaid windows, buttons and scroll bars</td>
</tr>
<tr>
<td>5</td>
<td>Windows</td>
<td>Turbo for Windows 3-6 months</td>
<td>3-6 months</td>
<td>As for 4, but integrated into Windows 3. No graphics as yet.</td>
</tr>
</tbody>
</table>

Figure 3. Levels of GUI programming in Turbo Pascal
• users want mouse support;
• sound does not really assist in a program development interface – it is distracting;
• careful partitioning of the screen is required when text such as program text is to be displayed.

AL used several windows partitioned vertically, so that full lines of Pascal could be displayed, but as a result could only show a small portion of a program at a time. It is clear that the goal of keeping all windows visible all the time has its drawbacks. These and other issues raised by AL are being further explored.

Phase 4 – Windows 3 or not

The current phase of Galileo focuses on the latter programming levels. Once students have been exposed to object-oriented programming in Turbo Pascal or C++, the obvious course is to make use of a platform of GUI software such as Turbo Vision. Fifty third year students at the University of Pretoria have embarked on group projects which for the most part will involve a GUI. Many of these will use Turbo Vision, and the cost in terms of effort and effectiveness of the final products will be studied.

Among the students there is a distinct division between those who feel the GUI should be based on Turbo Vision, and those who felt that it should be written for Windows 3. The errors present in Windows 3 are a drawback, but the ultimate market value of applications developed for it is undoubtable.

5 Conclusions

Computer technology is continually progressing, and with it the price of microcomputer equipment is decreasing. This provides for an increasing user community that requires new ways of interacting with these often intimidating technological advances. Graphic user interfaces serve to provide users with a clear view of complex computational activities.

This paper has highlighted the benefits of GUls. User effectiveness is the key to GUI acceptance. GUls stress intuition and logic. The see-and-point scenario evident in a GUI is far more appealing than the remember-and-type scenario in the prompt-driven world of DOS. A GUI provides for random access of pictures (icons). There is no longer a need to read through a display screen of text to ascertain a fact – GUls allow a user to establish a fact with a single glance. People recognise objects more easily than they recall text. But GUls do not only service the needs of the novice computer user, they inspire computer science allowing for further technological advancements.

The ingredients of a GUI are now familiar. Windows, icons, menus and a mouse are the features that should be part of any GUI found in today’s computer market. We emphasise that a GUI is not attempting to transform a valuable tool, the computer, into a grim resemblance of an arcade game. GUls can be entertaining but this is not their intention. GUls are attempting to create an environment where a user feels comfortable in front of a computer. A user must feel in control, and must be able to communicate with a computer effectively, in a language that is easily understood – the language of pictures. GUls are alternatives, not substitutes, for command-driven interfaces. GUls do not eliminate text totally, but rather substitute where possible by icons – some things simply cannot be represented graphically.

The easier it is to understand an application the easier it will be to get the job done. When considering GUI technology it is important to remember that the majority of computer users are not “experts”. The novice user represents the masses, the masses represent the consumer market, and the consumers encourage technological advances. GUls encourage computer usage, and consequently an increase in the consumer market. This ultimately leads to an improvement in computer technology.

Within the context of the new South Africa, therefore, GUls have a definite role to play. Not only do they transcend language barriers, but they can also open up technology to people whose literary skills would otherwise hamper them.

Computer science is often discussed in a mathematical context. But using a computer is not a science. Psychology plays a large role in designing an acceptable interface. Duplicating a file through DOS involves the use of a command language that looks to a novice like a physics formula. Using a mouse to click on an icon is a more natural approach to computing. Abstraction has always been a goal of computer scientists. GUls help achieve this goal.

The Galileo project aims to produce quantitative evidence of the effort and effectiveness of creating cheap but good GUls within a variety of hardware and software configurations. As a byproduct, it hopes to raise the awareness of GUls among today’s students and tomorrow’s computer managers.

References


Appendix – Major GUIs Pros and Cons

Every GUI has its good and bad points. Here is a summary of the major GUIs found in today’s circle of computer users [Barron et al 1991]. Major GUIs include:

- Windows 3.0
- Macintosh
- Open Look
- Ensemble
- NextStep
- Motif
- OS/2 Presentation Manager
- Amiga Workbench

These GUIs support certain features and lack others. The success of each individual GUI stems from its capabilities and most of all, from the number of applications that are available. Here is a list of those features pertaining to the above mentioned GUIs:

- GUI is an integral part of the machine (Macintosh, Amiga Workbench, NextStep)
- numerous applications available (Macintosh, Windows 3.0, Amiga Workbench)
- multitasking capabilities (Open Look, Ensemble, NextStep, Motif, OS/2 Presentation Manager, Amiga WorkBench, Windows 3.0)
- network capability (Macintosh, Open Look, NextStep, Motif)
- applications are stable and compatible with each other (Macintosh, NextStep, OS/2 Presentation Manager)

There are some other considerations however. In particular, Windows 3.0 requires a considerable investment in hardware and memory to take full advantage of its features. The problem with GUIs such as Motif and the Amiga Workbench is that there are no mandatory rules for developers, resulting in inconsistent applications. In contrast, Apple maintains strict guidelines for software developers leading to applications that are stable and robust.
Notes for Contributors

The prime purpose of the journal is to publish original research papers in the fields of Computer Science and Information Systems, as well as shorter technical research papers. However, non-refereed review and exploratory articles of interest to the journal's readers will be considered for publication under sections marked as Communications or Viewpoints. While English is the preferred language of the journal, papers in Afrikaans will also be accepted. Typed manuscripts for review should be submitted in triplicate to the editor.

Form of Manuscript

Manuscripts for review should be prepared according to the following guidelines:

- Use wide margins and 1\frac{1}{4} or double spacing.
- The first page should include:
  - title (as brief as possible);
  - author's initials and surname;
  - author's affiliation and address;
  - an abstract of less than 200 words;
  - an appropriate keyword list;
  - a list of relevant Computing Review Categories.
- Tables and figures should be numbered and titled. Figures should be submitted as original line drawings/printouts, and not photocopies.
- References should be listed at the end of the text in alphabetic order of the (first) author's surname, and should be cited in the text in square brackets [1, 2, 3]. References should take the form shown at the end of these notes.

Manuscripts accepted for publication should comply with the above guidelines (except for the spacing requirements), and may be provided in one of the following formats (listed in order of preference):

1. As (a) \LaTeX file(s), either on a diskette, or via e-mail/ftp – a \LaTeX style file is available from the production editor;
2. In camera-ready format – a detailed page specification is available from the production editor;
3. As an ASCII file accompanied by a hard-copy showing formatting intentions:
   - Tables and figures should be on separate sheets of paper, clearly numbered on the back and ready for cutting and pasting. Figure titles should appear in the text where the figures are to be placed.
   - Mathematical and other symbols may be either handwritten or typed. Greek letters and unusual symbols should be identified in the margin, if they are not clear in the text.

Further instructions on how to reduce page charges can be obtained from the production editor.

Charges

Charges per final page will be levied on papers accepted for publication. They will be scaled to reflect scanning, typesetting, reproduction and other costs. Currently, the minimum rate is R20-00 per final page for \LaTeX or camera-ready contributions and the maximum is R100-00 per page for contributions in typed format.

These charges may be waived upon request of the author and at the discretion of the editor.

Proofs

Proofs of accepted papers in categories 3 and 4 above will be sent to the author to ensure that typesetting is correct, and not for addition of new material or major amendments to the text. Corrected proofs should be returned to the production editor within three days.

Note that, in the case of camera-ready submissions, it is the author's responsibility to ensure that such submissions are error-free. However, the editor may recommend minor typesetting changes to be made before publication.

Letters and Communications

Letters to the editor are welcomed. They should be signed, and should be limited to less than about 500 words.

Announcements and communications of interest to the readership will be considered for publication in a separate section of the journal. Communications may also reflect minor research contributions. However, such communications will not be refereed and will not be deemed as fully-fledged publications for state subsidy purposes.

Book reviews

Contributions in this regard will be welcomed. Views and opinions expressed in such reviews should, however, be regarded as those of the reviewer alone.

Advertisement

Placement of advertisements at R1000-00 per full page per issue and R500-00 per half page per issue will be considered. These charges exclude specialized production costs which will be borne by the advertiser. Enquiries should be directed to the editor.

References

Contents

Forward .......................................................... i
Referees ................................................................ ii

Machine Translation from African Languages to English
DG Kourie, WJ vd Heever and GD Oosthuizen .................. 1

Automatically Linking Words and Concepts in an Afrikaans Dictionary
PZ Theron and I Cloete ............................................ 9

A Lattice-Theoretic Model for Relational Database Security
A Melton and S Shenoi ....................................... 15

Network Partitions in Distributed Databases
HL Viktor and MH Rennhackkamp ......................... 22

A Model for Object-Oriented Databases
MM Brand and PT Wood ................................. 27

Quantifier Elimination in Second Order Predicate Logic
D Gabbay and HJ Ohlbach ............................. 35

Animating Neural Network Training
E van der Poel and I Cloete .............................. 44

HiLOG - a Higher Order Logic Programming Language
RA Paterson-Jones and PT Wood ................. 53

ESML - A Validation Language for Concurrent Systems
PJA de Villiers and WC Visser .................. 59

Semantic Constructs for a Persistent Programming Language
SB Sparg and S Berman .................................... 65

The Multiserver Station with Dynamic Concurrency Constraints
CF Kriel and AE Krzesinski .................. 75

Mechanizing Execution Sequence Semantics in HOL
G Tredoux .................................................. 81

Statenets - an Alternative Modelling Mechanism for Performance Analysis
L Lewis ................................................... 87

N Pendock .............................................. 95

Galileo: Experimenting with Graphical User Interfaces
R Apteker and JM Bishop ................................... 99

Placing Processes in a Transputer-based Linda Programming Environment

Accessing Subroutine Libraries on a Network
PH Greenwood and PH Nash ..................... 117

A Multi-Tasking Operating System Above MS-DOS
R Foss, GM Rehmet and RC Watkins ....... 122

Using Information Systems Methodology to Design an Instructional System
BC O’Donovan ........................................ 126

Managing Methods Creatively
G McLeod ........................................... 131

A General Building Block for Distributed System Management
P Putter and JD Roos ................................ 141