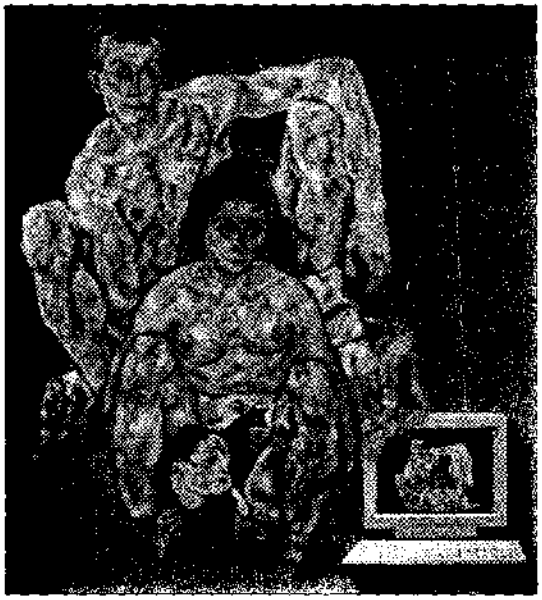
Thomas Grechenig Manfred Tscheligi (Eds.)

CCo 1- 733

Human Computer Interaction

Vienna Conference, VCHCI '93, Fin de Siècle
Vienna, Austria, September 20-22, 1993
Proceedings



Springer-Verlag
Berlin Heidelberg New York
London Paris Tokyo
Hong Kong Barcelona
Budapest

Series Editors

Gerhard Goos
Universität Karlsruhe
Postfach 69 80
Vincenz-Priessnitz-Straße 1
D-76131 Karlsruhe, Germany

Juris Hartmanis
Cornell University
Department of Computer Science
4130 Upson Hall
Ithaca, NY 14853, USA

Volume Editors

Thomas Grechenig
Department of Computer Science, Vienna University of Technology
Resselgasse 3/188, A-1040 Vienna, Austria

Manfred Tscheligi
Department of Applied Computer Science, University of Vienna
Lenaugasse 2/8, A-1080 Vienna, Austria

6341

CR Subject Classification (1991): D.2.2, H.1.2, H.5, I.2.7, I.7, K.3, K.4

ISBN 3-540-57312-7 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-57312-7 Springer-Verlag New York Berlin Heidelberg

Typesetting: Camera-ready by author
Printing and binding: Druckhaus Beltz, Hemsbach/Bergstr.
45/3140-543210 - Printed on acid-free paper

PREFACE

The scientific orientation of the VCHCI '93 corresponds to the mainstream HCI research spectrum which is represented by this year's prime events: INTERCHI '93 in Amsterdam and HCI '93 in Orlando.

The motto of the Viennese conference FIN DE SIÈCLE affiliates Vienna's intellectual tradition to the field's progressive development at the end of this century: Vienna in the FIN DE SIÈCLE was a place of intensity, disputes and rapid development in culture, politics and science. MODERNISM provoked both the moral-scientific tradition as well as the aesthetic-aristocratic ideal. By absorbing the fashionable poetic and plastic culture of all Europe in his language glowing darkly with purple and gold, the adolescent narcissus Hugo von Hofmannsthal became the idol of Vienna's culture-ravenous intelligentsia. Karl Kraus, the city's most acidulous moralist, poured contempt upon "THAT GEM-COLLECTOR" Hofmannsthal, who "FLEES LIFE AND LOVES THE THINGS WHICH BEAUTIFY IT" At that extraordinary time and location much of modern culture and thought was born out of a crisis of political and social disintegration. FREUD, MAHLER, SCHNITZLER, KLIMT were all working within a few steps from one another. ADLER, LOOS, SCHOENBERG at the same time discovered and developed their talents. KOKOSCHKA, SCHIELE and WITTGENSTEIN spent an inspiring youth then.

A century later, in today's fin de siècle, the VCHCI emphasizes on showing that HCI

- is more than AN AREA TO BEAUTIFY interaction with computers
- PROVOKES DISPUTES among its different contributing fields
- does not FLEE THE VITAL QUESTIONS for people using computers
- provides RADICALLY NEW opportunities for users

It is a pleasure to thank the invited speakers

William Buxton	Xerox PARC & University of Toronto
Tom Hewett	Drexel University

for accepting our invitation to give a keynote talk.
We are most grateful to the following cooperating organizations and their representatives:

ACM-SIGCHI	Robert Jacob, Tom Hewett
BCS	Victoria Bellotti
GI	Horst Oberquelle
IFIP TC 13	Brian Shackel
OCG	Günter Haring

Scientists and engineers from industry, academia, and major research institutes from 19 countries have contributed to the Vienna Conference on Human Computer Interaction 1993 (VCHCI '93). All full papers have been judged by at least three members of the Programme Committee on technical soundness, originality and innovativeness of work, importance and relevance to HCI research, clarity of presentation and methods as well as credibility of results. Only submittals of high scientific quality were accepted as papers. Posters have been selected on the basis of potential interest of VCHCI attendees. All contributions address the latest research and application in the human aspects of design and use of computing systems. The accepted contributions cover a large field of human computer interaction including design, evaluation, interactive architectures, cognitive models, workplace environment, as well as HCI application areas.

We wish to thank the authors and the members of the Programme Committee who so diligently contributed to the success of the conference and the direction of these proceedings.

Thomas Grechenig
University of Technology Vienna

Manfred Tscheligi
University of Vienna

August 1993

PROGRAMME COMMITTEE

Beth Adelson	Rutgers University, Camden, USA
Bengt Ahlstrom	Royal Institute of Technology, Stockholm, SWE
Sandrine Balbo	LGI, Grenoble, FRA
Sebastiano Bagnara	University of Siena, ITA
David Benyon	Open University, Milton Keynes, UK
Meera M. Blattner	Lawrence Livermore National Laboratory, USA
Ahmet Cakir	Ergonomics Institute, Berlin, GER
Gilbert Cockton	University of Glasgow, UK
Prasun Dewan	Purdue University, West Lafayette, USA
Gitta Domik	University of Paderborn, GER
Sarah Douglas	University of Oregon, Eugene, USA
Wolfgang Dzida	GMD, Sankt Augustin, GER
Scott Elrod	Xerox PARC, Palo Alto, USA
Tom Erickson	Apple Computer, Cupertino, USA
Giorgio P. Faconti	CNR, Pisa, ITA
James Foley	Georgia Institute of Technology, Atlanta, USA
Andrew U. Frank	University of Technology Vienna, AUT
William W. Gaver	Xerox EuroPARC, Cambridge, UK
Peter Gorny	University of Oldenburg, GER
Richard A. Guedj	Institut National des Telecomm. Evry, FRA
Nuno M. Guimaraes	INESC, Lisbon, POR
Judy Hammond	University of Technology Sydney, AUS
Tom Hewett	Drexel University, Philadelphia, USA
James D. Hollan	Bellcore, Morristown, USA
Bradley Hartfield	University of Hamburg, GER
Ulrich Hoppe	GMD-IPSI, Darmstadt, GER
Robert J.K. Jacob	Naval Research Laboratory, Washington DC, USA
Peter Johnson	University of London, UK
Clare-Marie Karat	IBM United States, Greenwich, USA
John Karat	IBM Watson Res. Center, Yorktown Heights, USA
Wendy A. Kellogg	IBM Watson Res. Center, Yorktown Heights, USA
Werner Kuhn	Vienna University of Technology, AUT
David Kurlander	Microsoft Research, Redmond, USA
Clayton Lewis	University of Colorado, Boulder, USA
Jonas Lowgren	Linkoping University, SWE
Allan MacLean	Xerox EuroPARC, Cambridge, UK
David Maulsby	University of Calgary, CAN
James R. Miller	James R. Miller, Apple Computer, Cupertino, USA
Michael J. Muller	US WEST Advanced Techn., Boulder, USA
Dianne Murray	University of Surrey, UK
Robert Neches	University of South. Calif., Marina del Rey, USA
Gary M. Olson	University of Michigan, USA
Paolo Paolini	Politecnico di Milano, ITA
Franz Penz	Rutherford Appleton Laboratory, Chilton, UK
Christian Rathke	University of Stuttgart, GER
Matthias Rauterberg	ETH Zurich, CH
Harald Reiterer	GMD, St. Augustin, GER

Mary Beth Rosson	IBM Watson Res. Center, Yorktown Heights, USA
Gavriel Salvendy	Purdue University, West Lafayette, USA
Dominique L. Scapin	INRIA, Le Chesnay, FRA
Helmut Schauer	University of Zurich, CH
Franz Schiele	GMD-IPSI, Darmstadt, GER
Chris Schmandt	MIT Media Laboratory, Cambridge, USA
Chris Shaw	University of Alberta, CAN
John Stasko	Georgia Institute of Technology, Atlanta, USA
Tom Stewart	System Concepts, London, UK
Martha R. Szczur	NASA/Goddard Space Flight Center, Greenbelt, USA
Pedro Szekely	University of Southern California, Marina del Rey, USA
Michael Tauber	University of Paderborn, GER
Jo Tombaugh	Carleton University, Ottawa, CAN
Thomas S. Tullis	Canon Information Systems, Costa Mesa, USA
Andrew Turk	University of Melbourne, AUS
Gerrit van der Veer	Free University, Amsterdam, NLD
Ina Wagner	Vienna University of Technology, AUT
Yvonne Waern	Linkoping University, SWE
Pierre Wellner	Xerox EuroPARC, Cambridge, UK
Alan Wexelblat	MIT Media Laboratory, Cambridge, USA
Juergen Ziegler	Fraunhofer-Institut IAO, Stuttgart, GER

CONFERENCE CHAIRS

Thomas Grechenig	Vienna University of Technology, AUT
Manfred Tscheligi	University of Vienna, AUT

ORGANIZING CHAIR

Monika Fahrnberger	Vienna University of Technology, AUT

The VCHCI '93 is an in co-operation conference with ACM-SIGCHI. It is supported by BCS, GI, OCG and IFIP TC 13

TABLE OF CONTENTS

[illegible]

SUPPORTING THE DEVELOPERS

[illegible]

Integrating Interactive 3D-Graphics into an Object-Oriented Application Framework

Dominik Eichelberg and Philipp Ackermann
University of Zurich, Department of Computer Science, Multi-Media-Lab
Winterthurerstrasse 190, CH-8057 Zürich, Switzerland
e-mail: eichel@ifi.unizh.ch, ackerman@ifi.unizh.ch

Abstract. This paper describes the integration of 3D graphics into the visual 2D part of an application framework. Most object-oriented application frameworks are built to ease the development of interactive graphical applications that use direct manipulation techniques. This study is based on the object-oriented application framework ET++ that provides predefined visual classes for text, 2D graphics display, and standard user interface components. We extended ET++ to support 3D graphics in a general way. It is now possible to integrate 3D graphics objects which may be placed wherever other visual objects can go, i.e. in scrollable views, as building blocks in dialogs or graphics editors, as characters in text editors, as items in lists or pop-up menus, etc. As a consequence 2D and 3D graphics are dealt in a uniform way. Interaction techniques on 3D graphics objects and 3D user interface components are supported.

Keywords: interactive 3D graphics, user interface, C++, object-oriented application framework, ET++

1. Introduction

Graphical user interfaces with direct manipulative interaction techniques based on 2D graphics are widely used in today's computer applications and have replaced command language and menu driven programs. Integrating new media such as 3D graphics, audio, and video into human computer interaction can enrich the user's abbilities to process complex context. The implementation of such interfaces often requires great efforts. The realization of modern user interfaces tend to consume a great part of a project time. The development of object-oriented application frameworks was very successful in reducing the complexity of 2D user interface programming. Examples are MacApp™ [Schmucker89], Interviews [Linton89], ET++ [Weinand89], and NeXTStep™ [NeXT92]. The purpose of our study is to investigate whether these object-oriented techniques are also suitable for multimedia extensions of the user interface. These extensions increase the accessibility and usability of computer applications that process complex data. In order to do these explorations the object-oriented application framework ET++ [Gamma88, Gamma92, Weinand89, Weinand92] is used. ET++ was chosen because it is highly portable, it integrates many features in a seamless fashion, and its source code is available in the public domain. As a first step of the multimedia extension the 2D graphics model of ET++ was extended to 3D graphics. In this paper we explain our motivation to integrate 2D and 3D graphics, introduce the object-oriented framework ET++, and present our approach to extend it with interactive 3D graphics.

2. Integrating 2D and 3D Graphics into the User Interface

The interactive direct manipulation paradigm of today's user interfaces is strongly coupled with the presentation and visualization of data objects. The information which is modeled and manipulated in applications becomes more and more complex as the processing power of workstations increases. New media such as interactive 3D graphics, animations, audio, and video are introduced in multimedia systems to allow richer presentation of complex data. It is extremely costly to implement user interfaces based on direct manipulation techniques for these new media. It takes a good deal of thought and care to create user interface components that combine visualization and manipulation of these media into useful and reusable software.

Not long ago 3D graphics was a field for specialists. They often were programmers and users of their programs in one person. They were not interested in software ergonomics, instead they concentrated their work on optimizing graphics performance. On the other hand there is little help in 3D graphics libraries for 2D interaction elements like pop-up and pull-down menus, buttons, sliders, scroll-bars, and so on. As a consequence, ad hoc solutions with no or separated 2D user interface components were developed. Today, 3D graphics is available on low cost workstations and users are not computer specialists that are satisfied with an ad hoc user interface. This means that 2D user interface components must be merged seamlessly with 3D graphics. In doing so it is even possible to add new 3D interaction components into the user interface. Realizing the interaction elements as a set of cooperative classes in an object-oriented framework will increase the software reusabilty.

3. The Object-Oriented Application Framework ET++

ET++ is an object-oriented class library integrating user interface building blocks, basic data structures, support for object input/output, printing, and high level application framework components. It eases the development of interactive textual and graphical applications with direct manipulation and high semantic feedback. ET++ is implemented with the programming language C++ and uses abstract classes as a portability layer and interface protocol to concrete window and operating systems (Figure 1). It is available for UNIX and VMS and runs either with SunView or with X11 window system.

In the domain of user interaction ET++ supports direct manipulation, efficient flicker-free screen update, multi-undoable commands, and a flexible mechanism to compose visual objects with declarative layout specification and automatic object positioning. Standard interaction components like pop-up and pull-down menus, buttons, scroll-bars, etc. are included as predefined classes that can be extended through inheritance. The application framework part of ET++ consists of abstract classes for the following concepts: application, manager, document, view, and command. These abstract classes define a generic application with predefined functionality that will be inherited by the respective subclasses of a concrete application. According to [Schmucker89] an application framework is a set of interconnected objects that provides the basic functionality of a working application, but which can be easily specialized (through subclassing and inheritance) into individual applications. An application framework not only allows the reuse of code as a class library, but also the reuse of design structures, because the dependencies between objects are preimplemented through predefined object composition possibilities, event dispatching mechanism and message flow control. The following functionality will automatically be part of any ET++ application:

- creating, saving, and opening of documents with a file dialog
- handling of several documents in different windows by one application
- scrolling of views through scroll-bars with auto-scrolling feature
- automatic layout management of visual components
- mouse and keyboard event handling
- multi-undoable commands
- clipboard cut/copy/paste, drag and drop
- flicker-free drawing with double-buffering
- device-independent printing
- change propagation
- program inspection by a programming environment based on run-time class information

Any extension of ET++ must be aware of this functionality and avoid conflicts with the sophisticated generic behavior of the application framework. In order to develop 3D graphics applications with ET++ to use its interface components and event handling mechanism, it is necessary to integrate 3D concepts into the application framework. The closer they fit into the philosophy of the framework, the better they will be usable by the programmer.

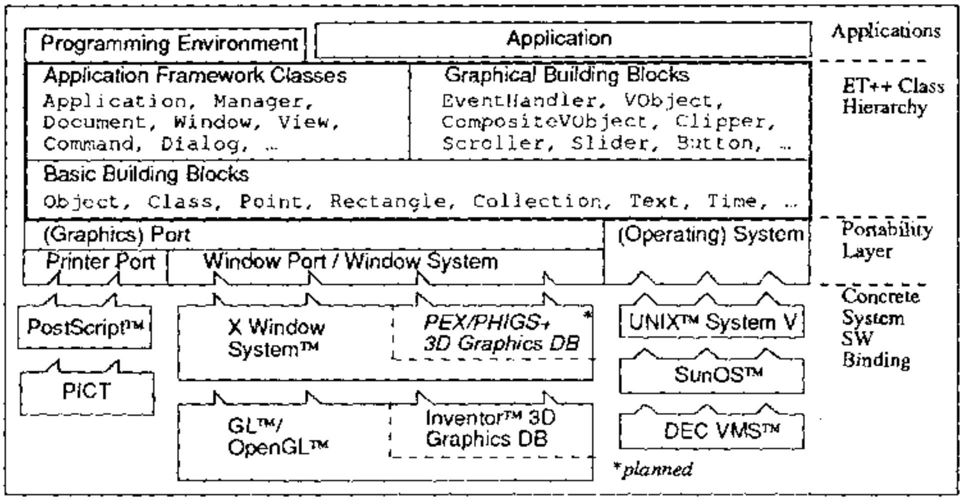

Programming Environment
Application
Applications
Application Framework Classes
Application, Manager, Document, Window, View, Command, Dialog, ...
Graphical Building Blocks
EventHandler, VObject, CompositeVObject, Clipper, Scroller, Slider, Button, ...
ET++ Class Hierarchy
Basic Building Blocks
Object, Class, Point, Rectangle, Collection, Text, Time, ...
(Graphics) Port
(Operating) System
Printer Port
Window Port / Window System
Portability Layer
PostScript™
PICT
X Window System™
PEX/PHIGS+ 3D Graphics DB *
GL™/ OpenGL™
Inventor™ 3D Graphics DB
UNIX™ System V
SunOS™
DEC VMS™
Concrete System SW Binding
*planned

Fig. 1. The system architecture of ET++.

4. Concepts of Integrating 2D and 3D Graphics

Support for new media, such as 3D graphics, audio, and video, is added to workstation hardware so quickly that monolithic software is obsolete within a short time. The experience with ET++ has shown that good object-oriented design can obtain extensibility even for large and complex software systems. The extensibility and adaptability of ET++ is guaranteed by a portability layer (Figure 1) which consists of abstract classes defining method protocols. Beside code reuse, inheritance is used to propagate common method protocols in order to ensure flexible object composition. Concrete behaviour is realized in subclasses of the abstract classes. There exist concrete classes for specialized operating systems, window systems, and printers.

The ET++ class hierarchy only operates with the abstract classes of the portability layer and is therefore completely independent of a specific system environment. Dynamic binding is used to couple an abstract interface to a concrete realization. This architecture makes it easy to create adaptor classes for other system resources such as 3D graphics. The two following possibilities to integrate 3D graphics in the object-oriented application framework ET++ were investigated:

a) Creating a New Window for Each 3D View
Window systems such as the X11 Window System [Scheifler90] allow the allocation of windows not only for standard windows with their own borders, but also for many separate graphical objects in the main window. In the Xt toolkit these subwindows are called *widgets* and form a window hierarchy. By defining a widget for each 3D view it is possible to render 3D graphics in a separate window without disturbing the 2D part of the framework. There are some disadvantages inherent to such 3D interface objects:

- 3D object views are restricted to be rectangular on the 2D display since most window systems do not provide arbitrarily shaped windows.
- A 3D object cannot be covered by a visual object which is not a window.
- A 3D object cannot intersect 3D objects in a different widget or window.
- In ET++, standard visual objects are not windows, and define their own methods for clipping, layout management, event handling, and cut/copy/paste. Problems arise in implementing these methods for visual objects being windows (e.g. the event distribution mechanism of ET++ is disturbed by windows).

b) Combining 2D and 3D Rendering in the Same Window
Rendering 2D and 3D graphics in the same window allows a flexible combination of the two rendering modes. In this approach 3D objects can overlay or be overlaid by 2D objects and intersect other 3D objects.

The visible part of an ET++ application is hierarchically composed of visible objects called *VObjects*. In ET++, a *VObject* is not a window. Therefore, the second approach of combining 2D and 3D graphics in the same window is preferred. By deriving a new visible 3D object class *V3DObject* from *VObject*, the integration of 3D graphics in the framework is seamless. The class *V3DObject* provides all necessary functionality to work with the 2D part of ET++, e.g. it has methods for drawing, clipping and manipulating (resizing) the 2D content rectangle. In addition to the *VObject*, it defines parameters required for 3D visualization, such as camera position, zooming, twist, light source, etc. It also implements direct manipulation operations for these parameters. The abstract class *V3DObject* must be overwritten to implement a concrete 3D view.

5. Implementation

For the reasons mentioned above, the second approach combining 2D and 3D graphics in the same window was implemented on Silicon Graphics IRIS workstations using the Graphics Library (GL) [SGI91]. GL was chosen because it supports *immediate-mode* rendering. Immediate-mode rendering displays graphical primitives directly into the framebuffer of the screen, where *retained-mode* rendering collects graphical primitives in a database (display list) and redisplays the database every time it is changed. Other 3D graphics libraries like PHIGS+/PEX strongly recommend the use of retained mode rendering. Since most of the 2D graphics libraries support only immediate-mode rendering and the 2D rendering of ET++ is based on this drawing model, an immediate-mode drawing approach was chosen for the 3D graphics, too. This allows ET++ to have full control over all drawings which is important for animations in double buffer mode. It is generally agreed that GL will become an open standard for 3D graphics under the name OpenGL™. OpenGL is announced as a 3D extension for the X Window System and Windows NT on many hardware platforms [Foley92].

In the current version of GL it is not allowed to mix GL rendering with drawing by an additional 2D graphics library in the same window. For this reason all 2D graphics methods of the ET++ application framework had to be implemented using GL. All drawing in ET++ uses methods defined in the abstract class *Port*. The drawing methods of *Port* are specialized for window systems in a subclass called *WindowPort*, and for printing in a subclass *PrinterPort* (see Figure 2). The methods of *WindowPort* are specialized for a specific window system in a subclass of *WindowPort*. These virtual C++ methods are bound dynamically at run-time so that the actual and specialized window or printer port instance can be assigned to a global port variable. There exists for example a *XWindowPort* for the X Window System. This system architecture of ET++ makes it possible to implement a new class *GLPort* inherited from *WindowPort* which implements all drawing routines of ET++ with GL. We use GL in mixed-mode, which means that the input event handling and window management is done by the X Window System whereas the content of the window is rendered by GL. Beside standard drawing routines for line, rectangle, oval, polygon, etc. there exist *Font* and *Bitmap* classes to handle character and image drawing. In the GLPort they are all implemented with routines from the GL graphics library.

For the 3D extension of ET++, the class *Port* was extended by abstract 3D graphics methods which are implemented in concrete terms in the *GLPort*. All 3D rendering is based on 3D methods of the extended *Port* class. This will in the future allow to implement other derivations of the extended class *Port*, e.g. a *PEXPort*.

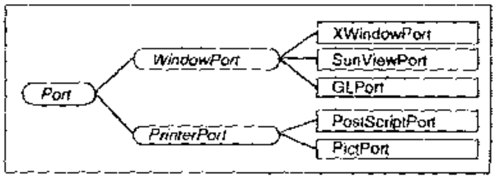


Fig. 2. The inheritance hierarchy of the class *Port* which functions as a graphics portability layer.

The class *V3DObject* is a subclass of the abstract class *VObject* which defines the protocols for drawing, automatic layout management, and event handling of all visual objects in ET++. *V3DObject* overwrites the method *DrawAll* of the class *VObject*. The framework calls this method every time a *VObject* should draw itself. In this method, *V3DObject* initialises the 3D drawing mode (e.g. activates the Z-buffer for hidden surface removal), sets the clipping region to the content rectangle, and places the camera and possibly defined light sources. Then it calls the method *Draw*, which should be overwritten in a subclass of *V3DObject* to implement a specific graphical appearance. The method *Draw* uses 3D methods of the class *Port* in order to define 3D graphics. *V3DObject* provides direct manipulation interaction by mouse movements on the *V3DObject* for camera and light positioning. By double-clicking the middle mouse button on a *V3DObject*, a camera/light control window opens which allows the interactive manipulation of the camera and light positions in front, top, side and perspective views in the coordinate system (Figure 3). The change propagation mechanism will automatically update all dependent views of the camera/light control. This functionality is implemented in *V3DObject* and is inherited by all its subclasses.

6. Sample Applications

To prove the flexibility of the *V3DObject* approach several reusable classes and applications were written using the new 3D features of ET++. The application shown in Figure 3 visualizes a bone that is sampled by a computer tomograph. The application has various interaction components created with the 2D part of ET++. These sliders, buttons, and menus interact with the 3D model. The view of the 3D data is created by a derivation of the class *V3DObject*. The user can rotate the bone by pressing the middle mouse button and dragging the mouse. These movements are commands and are therefore undoable. By double-clicking the 3D view, a separate camera/light control window appears where the position of the camera and the light can be changed interactivaly. All these features are inherited from the class *V3DObject*. For this application the 3D view only provides the drawing of the polygons. This is done by overwriting the method *Draw* of the super class *V3DObject*.

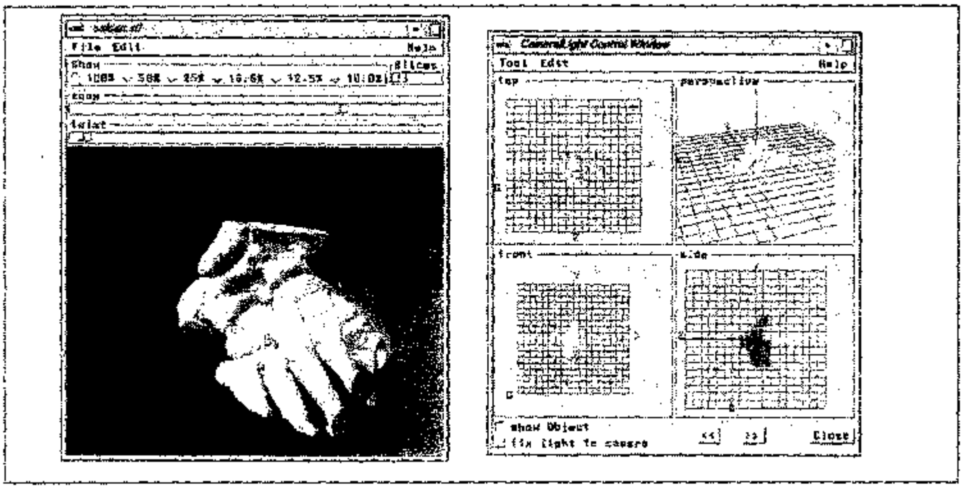


Fig .3. An application that visualizes medical data and displays a camera/light editor in a separate window.

The flexibility of objects inherited from *V3DObject* is shown in Figure 4. Since a menu-item in ET++ must be a *VObject* and a *V3DObject* is derivated from *VObject*, it is possible to integrate 3D graphics in menus. Obviously it can also be part of a dialog box. Until now *V3DObject* have not been overlaid by other visual objects. This ability is shown in the extension of the application *draw*, which in its 2D version is shipped with ET++ as a sample application. In Figure 5, 2D and 3D graphics objects are integrated in one view and can be positioned, sized, and arranged in any order. It is even possible to insert 3D objects in text as shown in Figure 6. While editing, the *V3DObject* will flow with the text, and cut/copy/paste operations are

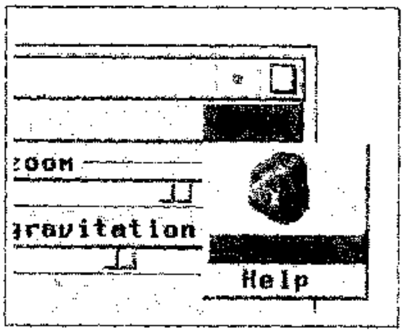


Fig .4. A 3D menu item.

possible. The extent of the 3D object can be expanded or reduced within the text by mouse control while the text layout adopts automatically. The text, including the 3D graphics, can be saved to and loaded from files.

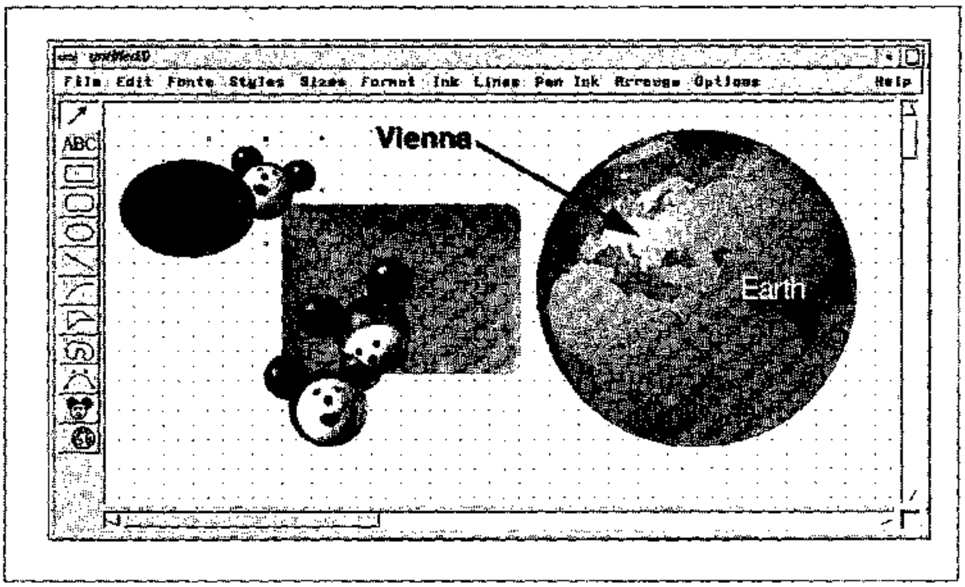
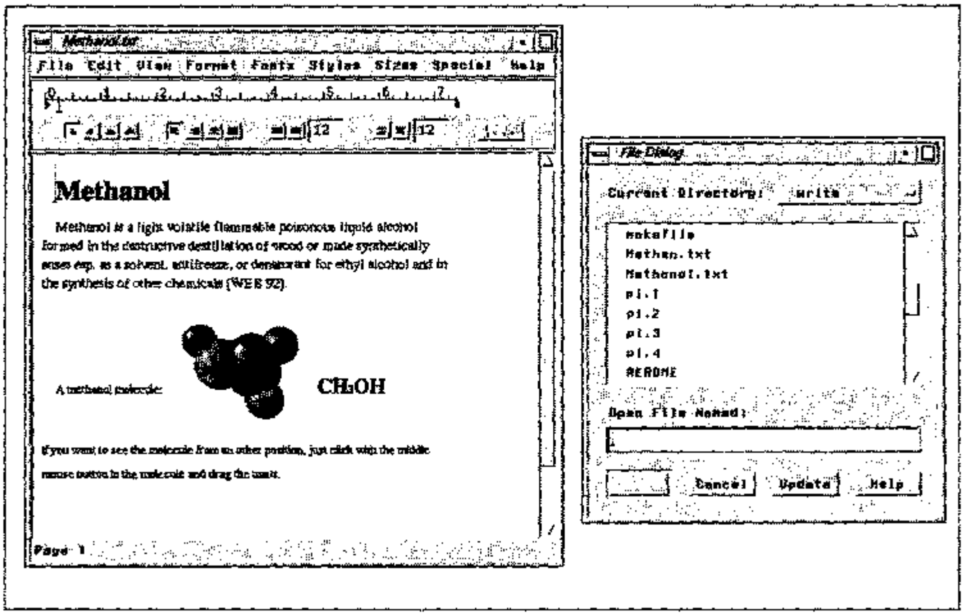

untitled0
File Edit Fonts Styles Sizes Format Ink Lines Pen Ink Arrange Options Help
ABC
Vienna
Earth

Fig. 5. *V3DObjects* can overlay or be overlaid by other *VObjects.* *V3DObjects* can intersect other *V3DObjects.*



Methanol

Methanol is a light volatile flammable poisonous liquid alcohol formed in the destructive distillation of wood or made synthetically uses esp. as a solvent, antifreeze, or denaturant for ethyl alcohol and in the synthesis of other chemicals [WEB 92].

A methanol molecule: CH_3OH

If you want to see the molecule from an other position, just click with the middle mouse button in the molecule and drag the mouse.

Fig 6. A*V3DObject* in a text editor.

All 3D graphics that are seen in Figures 3 to 6 are not images but active objects that can be manipulated in an interactive way with mouse events upon the *V3DObject* or in a separate camera/light control window.

7. Three-Dimensional User Interface Components

With the presented 3D extention of ET++ it is possible to realize 3D user interface components. The main problem of user interaction upon 3D data is the use of two degrees-of-freedom devices such as mouse or tablet to manipulate three degrees-of-freedom data. The functions of mouse movements on 3D views are currently switched with modifier keys and pressing one of the three mouse buttons. This is not an elegant solution because of its different states and modi. Depending on the state of the modifier keys and pressed mouse button it means selecting, selecting subelement, moving, stretching, rotating, camera moving, zooming, twisting, etc.

To address this problem, a 3D user interface component was realized that represents a joystick. Although the mouse movements are still two-dimensional, the visual feedback gives a three-dimensional feeling of the interaction. The joystick is selected by positioning the mouse cursor over the knob of the joystick and pressing the left mouse button. Moving the mouse while holding the mouse button will deflect the joystick and will continuosly generate control values. Angle and orientation of the stick relating to the neutral position are computed and considered in the data manipulation and feedback visualization. Releasing the mouse button causes the joystick to move back to its neutral position and to stop generating control values.

The application *etgeoid* visualizes the earth and shows the effect of varying gravitation (Figure 7). The control values of joystick interaction are mapped to camera position changes. Angle and distance of the joystick deflection determine spin orientation and turning speed of the camera rotation. We considered two display possibilities of the joystick: the view from top allows to adapt the mouse movements to the deflection of the joystick (Figure 8, left side). But in its neutral position, it is hardly recognized as a joystick. A perspective view of the joystick allows this recognition in the original state, but makes the reaction of the joystick to mouse movements less intuitive (Figure 8, right side). Until now, no decision was made which joystick view is better suited to interaction and therefore, the user has the posibility to switch the view by pressing the middle mouse button.

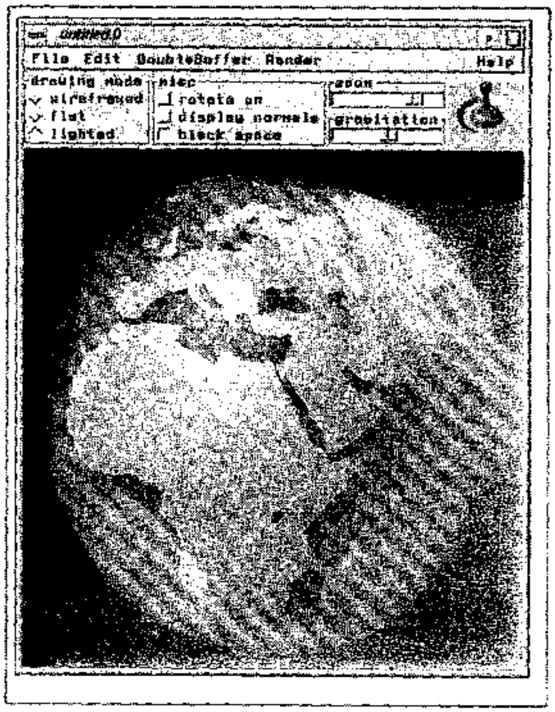


Fig. 7. Visualization of the geoid. A joystick changes the camera view point.

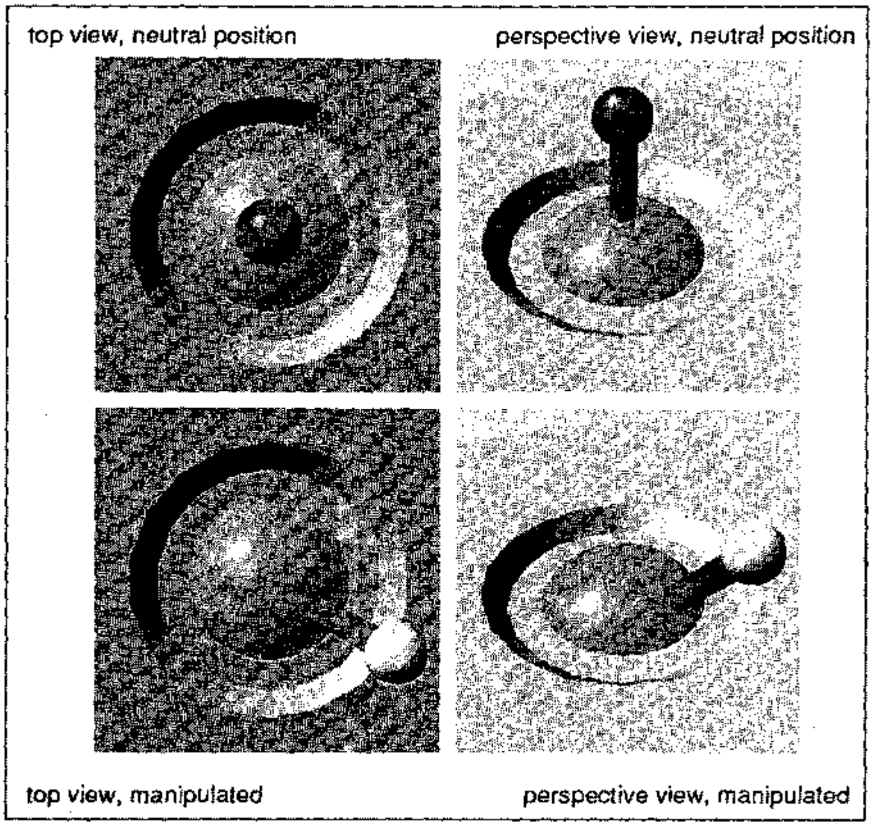

top view, neutral position
perspective view, neutral position
top view, manipulated
perspective view, manipulated

Fig. 8. Different views of the *Joystick* user interface component.

8. Conclusions and Future Work

The 3D graphics extension of the object-oriented application framework ET++ is seamlessly integrated into the 2D graphics part of the user interface. ET++ gives the opportunity to develop applications which combine interactive 2D and 3D graphics in a general way. Therefore, it can be used as an experimental development environment for new user interface components in order to design intuitive 3D visualization and interaction elements. The presented 3D extension of ET++ still lacks full 3D graphics support, e.g until now there exist no complete 3D modelling facilities and the problem of printing 3D graphics has not yet been addressed. In spite of this, we are satisfied of having proved the applicability of integrating 2D and 3D graphics seamlessly by implementing all drawing methods on a 3D graphics library. The performance of interactive 2D and 3D graphics is reasonably good on workstations such as the Silicon Graphics Indigo. Moreover, we want to emphasize that this kind of 3D graphics extension is extremely open because it will always be possible to enhance its implementation without interfering with the functionality of the application framework. Reusable classes were developed which are based on the 3D extension of the ET++ graphics port.

Future work includes refining of the presented 3D graphics classes, improving the modelling classes, and adding animation mechanisms in order to define time dynamic behavior. Currently we are developing classes for time synchronization, which handle different media such as 2D and 3D graphics, camera, audio, and MIDI objects with the same method protocol. Media presentations are regarded as hierarchical compositions of time objects that define serial or parallel synchronization of the inserted media objects.

9. Availability

The object-oriented application framework ET++ is public domain and distributed by anonymous ftp at iamsun.unibe.ch. The distribution includes all C++ source code and some sample applications. The usual educational and non-profit restrictions apply. The 3D graphics extension of ET++ which is discussed in this paper will be added to the public domain in a future ET++ release.

10. Acknowledgements

Thanks are due to André Weinand and Erich Gamma, which are the principal designers and developers of ET++. Peter Stucki and Martin Dürst gave helpful comments on a draft of this paper. Thanks go to Urs Meyer, who has developed an early non-object-oriented version of the geoid application [Meyer89].

References

[Foley92] James Foley (chairman): *3D Graphics Standards Debate: PEX versus OpenGL*; Proceedings of SIGGRAPH '92, in: Computer Graphics, 26, 2 (July 1992), ACM SIGGRAPH, New York, 1992, pp. 408-409.

[Gamma88] E. Gamma, A. Weinand, R. Marty: *ET++ – An Object-Oriented Application Framework in C++*; in Proc. EUUG, pp. 159-174, Cascais, Portugal, 3-7 Oktober 1988.

[Gamma92] E. Gamma: *Objektorientierte Software-Entwicklung am Beispiel von ET++*; Springer Verlag, Berlin 1992.

[Linton89] Mark A. Linton, John M. Vlissides, Paul R. Calder: *Composing User Interfaces with Interviews*; in IEEE Computer Vol. 22(2):8-22, Februar 1989.

[Meyer89] Urs Meyer: *Modellbildung und Animation eines Geoids*; in: M. Paul (Hrsg.), Proceedings GI-19. Jahrestagung I, Computergestützter Arbeitsplatz, pp. 452-459, München, Oktober 1989, Spinger Verlag, Berlin 1989.

[NeXT92] NeXT Computer Inc.: *NeXTStep Developer's Library*; Addison-Wesley, Reading MA, 1992.

[Scheifler90] R. Scheifler, J. Gettys: *X Window System: the complete reference to X lib, X protocol, ICCCM, XLFD*; 2nd Edition, Digital Equipment Corporation, 1990.

[Schmucker89] Kurt J. Schmucker: *Object Oriented Programming for the Macintosh*; Hayden, Hasbrouck Heights, New Jersey 1989.

[SGI91] *Graphics Library Programming Guide*; Silicon Graphics Computer Systems, Mountain View, California, 1991.

[Weinand89] A. Weinand, E. Gamma, R. Marty: *Design and Implementation of ET++, a Seamless Object–Oriented Application Framework*; Structured Programming, Vol. 10, No. 2, June 1989, pp. 63-87.

[Weinand92] André Weinand: *Objektorientierte Architektur für graphische Benutzeroberflächen – Realisierung der portablen Fenstersystemschnittstelle von ET++*; Springer Verlag, Berlin 1992.