ETH Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Institut für Computersysteme

Institut für Computersysteme
ETH-Zentrum
CH-8092 Zürich

Fax +41 1 261 5389
Tel. +41 1 254 7311
e-mail oberon@inf.ethz.ch

July 30, 1993/Hg

OBERON

Dear Madam/Sir:

Thank you very much for your interest in Oberon. The enclosed pamphlet informs you about the system in general, as well as about special implementations that are available for commercial environments. Implementations in diskette form can be ordered for sFr 50.– each (which is approximately 35 US$).

In case of DOS-Oberon, two versions are available: Standard and System 3/ Gadgets. The latter is an object-oriented evolution of the standard system with a graphical user interface toolkit. For more information on Gadgets, see the last page of the enclosed pamphlet.

If you are interested in getting one of the available implementations, please fill out one of the enclosed order forms (DOS-Oberon or general) and return or fax it back to us. Alternatively, you can also order directly via e-mail to oberon@inf.ethz.ch .

Sincerely yours

H. Gülgün, Secretary

Oberon™

by Institute for Computer Systems, ETH Zurich

Oberon, the successor of *Pascal* and *Modula-2*, is both a programming language and a novel programming environment designed by N.Wirth and J.Gutknecht at ETH Zurich. The Oberon System is available without fee from ETH. Currently there are implementations for *Apple Macintosh II*, *Digital Equipment DECstation*, *IBM PC (MS-DOS)*, *IBM RS/6000*, and *Sun SPARCstation*. Implementations for *Commodore Amiga*, *Digital Equipment Alpha*, *Hewlett-Packard PA-RISC*, and *MS-Windows* are under development. These implementations are completely source-code compatible with each other and share the same document architecture. This sheet shows some of the characteristics of the Oberon System and describes how to obtain it.

System

- Single-process multitasking
- Automatic garbage collection
- Commands: procedures that can be called like programs
- Dynamic loading (adding modules to a running program)
- Text as a built-in abstract data type
- Tools for text and graphics editing, and for program development

Compiler

- Generates native code, no separate linking necessary
- Very fast compilation
- Can compile directly from edit window

Language

- Pascal-like syntax
- Strong type checking
- Modules with type-checked interfaces and separate compilation
- Type extension, which provides for object-oriented programming
- Type-bound procedures (methods) in Oberon-2
- Support for run-time type tests
- Persistent objects
- Compatibility between all numeric types (mixed expressions)
- String operations
- Support for system programming

Literature

The standard software distribution contains sufficient basic on-line documentation to enable first-time users to get started with Oberon. However, for serious work we recommend the following books published by Addison-Wesley and Springer:

N. Wirth and M. Reiser: Programming in Oberon. Steps beyond Pascal and Modula.
Addison Wesley, 1992, ISBN 0-201-56543-9.
Tutorial for the Oberon programming language and concise language reference.

M. Reiser: The Oberon System. User Guide and Programmer's Manual.
Addison Wesley, 1991, ISBN 0-201-54422-9.
User manual for the programming environment and reference for the standard module library.

N. Wirth and J. Gutknecht: Project Oberon. The Design of an Operating System and Compiler.
Addison Wesley, 1992, ISBN 0-201-54428-8.
Program listings with explanations for the whole system, including the compiler for NS32000.

H. Mössenböck: Object-Oriented Programming in Oberon-2.
Springer, 1993, to be published.
Principles and applications of object-oriented programming with examples in Oberon-2.

How to get Oberon

The various Oberon implementations can be obtained via anonymous internet file transfer *ftp* (at no charge) or on floppy disks (for a fee of 50 Swiss Francs, which is about 35 U.S. Dollars). We accept payment via Eurocard/Mastercard or VISA. To order by credit card, specify your credit card number, expiration date, and your name exactly as it appears on the card.

FTP: neptune.inf.ethz.ch (129.132.101.33) Directory: Oberon

For further information please contact our secretary at:
Institut für Computersysteme ETH-Zentrum, CH-8092 Zürich, Switzerland
Telephone +41 (1) 254 7311, Facsimile +41 (1) 262 3973, E-Mail oberon@inf.ethz.ch

What are Commands ?

In Oberon, the notion of *Program* is represented by its two distinct aspects of textual unit for compilation, called a *Module*, and of unit of requestable action, called a *Command*. Each command invocation is an atomic action in the dialog between the operator and the computer. To activate a command, one simply points at a text of the form *Module.Procedure* (e.g. Edit.Open) and clicks the middle mouse button. The global system state represents the command's parameter such as text immediately following the command, the most recent text selection, or even a marked viewer displaying a complex data structure. In addition to producing output into a file or writing directly to the screen as is done in conventional operating systems, commands in Oberon generate output in the form of (displayed) *data structures in memory*, thereby allowing for efficient processing by further command invocations.

What is Dynamic Loading ?

Unlike traditional systems, Oberon makes no distinction between *application* and *operating system*. By contrast, applications in the form of modules *extend the system* when they are loaded. Whenever a command is activated, Oberon looks for the referenced module. If it is not found in main memory, it is loaded from disk and the code of the command is executed. Upon further activations of commands of the same module, it is already present and need not be loaded again. Therefore, only those modules are loaded which are needed during the current session. An additional advantage of this scheme is that the functionality of a set of modules can be *extended even at run-time* – say a text editor with a print module – without changing these running modules. This feature is used extensively throughout the system.

What is Extensibility ?

Through the consistent application of *object-oriented techniques* in the design of Oberon, the system is extensible in several ways. Not only can new modules and therefore procedures be added to a running system through dynamic loading, but also data can be extended at run-time through the notion of *type extension*. New data types can *rely on existing algorithms* to perform common tasks. For instance, a new frame class displaying a hardware design need not concern itself with the management of such frames on the screen. It simply *reuses already existing procedures* for that task.

What is Integrated Text ?

By integrating the data structure *Text* into the system's core, commands can be activated and parameters for commands can be composed wherever text is being displayed on the screen. Output produced by commands is editable immediately and can be processed by other commands. The user rarely has to type in a command on the keyboard because it is probably somewhere on the screen already. The textual nature of commands allow menus to be customized in the form of *tool texts*, which can be edited freely.

What are Tiled Viewers ?

Instead of overlapping windows (*viewers*) mirroring documents piled up on a desk, Oberon uses a *tiled viewer system* to arrange its windows. The display area is divided up into *tracks* – a user track on the left and a system track on the right side. These can be divided up further by viewers which cover the whole track's width. The result is a less cluttered and clearly arranged screen.

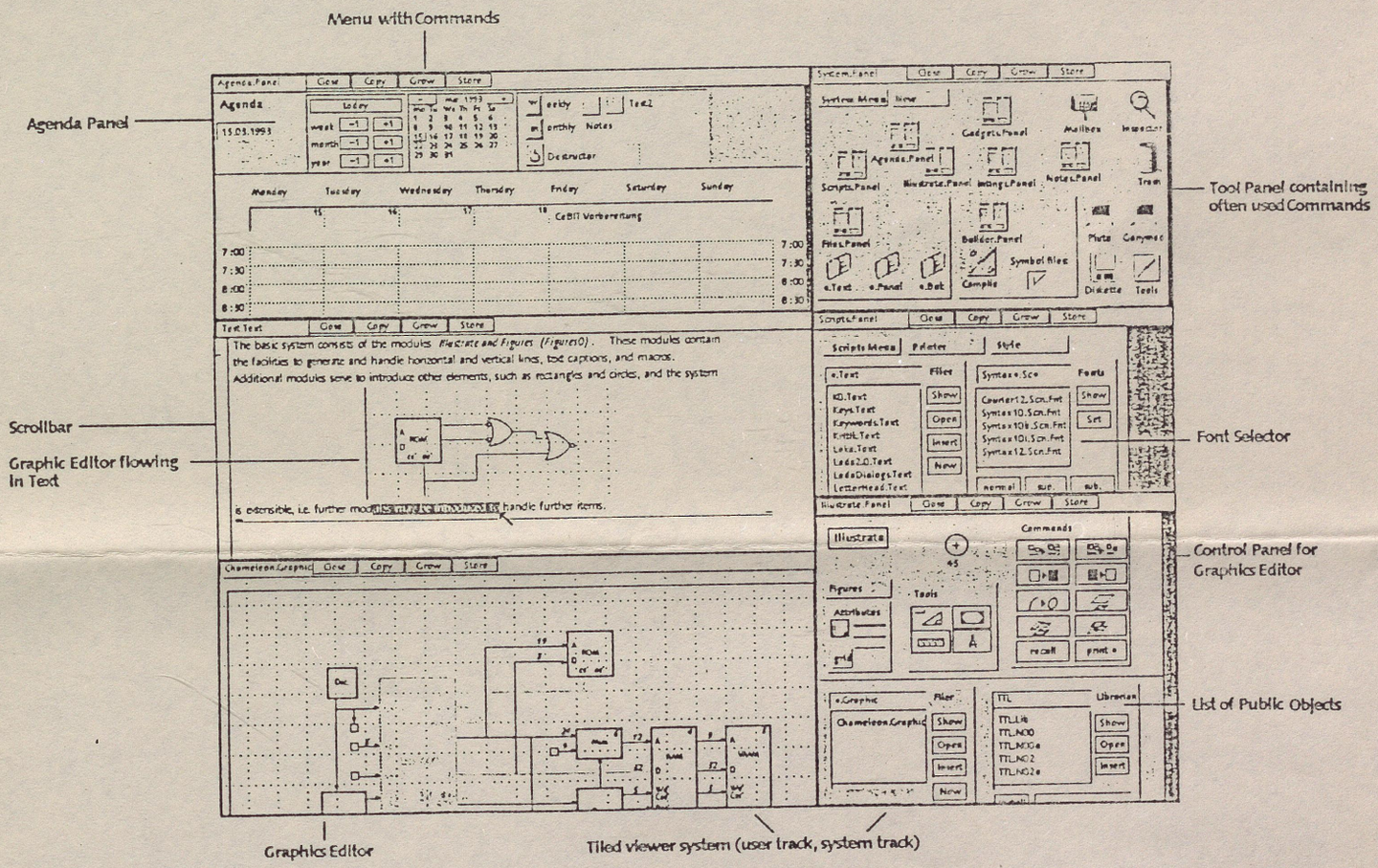
What is Garbage Collection ?

Garbage collection is the process of *reusing memory* that is no longer referenced. It is a prerequisite for truly *extensible design*, in which programmers of software modules cannot know all clients and possible extensions of their data structures and therefore cannot be responsible for the deallocation of memory. In Oberon, the garbage collector is invoked *between commands*, when temporary data structures are no longer referenced. Thereby, the process can be kept simple and efficient. In contrast to other systems, one seldom notices the presence of Oberon's garbage collector.

What is the Gadgets System ?

The Gadgets System is an experimental graphical user interface management system. It allows the *run-time construction of user interfaces* from basic building blocks called *gadgets*. The system comes standard with a set of often used gadgets. They range from simple ones like buttons, checkboxes, and scrollbars, to more complicated ones like gadget and drawing editors. The user combines gadgets at run-time, using a process called *assembly* or *composition*, to build more complicated gadgets from simpler ones. They act as functionally independent units that can be used in any environment, allowing them to be integrated in each other, or into texts, drawing systems, page layout systems etc. *Gadgets are completely embedded in all environments and can be edited in place.* User interfaces can be constructed separately from the application, and in many cases allow the construction of graphical interfaces for existing text-based Oberon applications. One consequence of such a model is that the distinction between the application and the document disappears. The user may also freely modify user interfaces, possibly *customizing* them to personal preferences, or to extract useful components from other applications. With time, it is expected that many more gadgets will become available to be used in applications.

In addition, programmers may add their own functionality to the system, by building new types of gadgets, or extending existing ones using *object-oriented techniques*. Users or programmers can take control of their own or of other gadgets using either the Oberon programming language or a simple script facility. The system provides a powerful imaging model that helps to reduce the burden of programming graphical applications. Experience with student projects show that a student can get to know the system and build meaningful graphical applications in less than one semester.



ETH Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Institut für Computersysteme

Institut für Computersysteme
ETH-Zentrum
CH-8092 Zürich

Tel. +41 1 254 7311
Fax +41 1 261 5389
e-mail sekcs@inf.ethz.ch

OBERON

ORDER FORM

Name:

Address:

Telephone Number:

Computer Type:

Cheque
or
Credit Card (Euro/Master or VISA):

No.: Expiry date:

Date: Signature:

ETH Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Institut für Computersysteme

Institut für Computersysteme
ETH–Zentrum
CH–8092 Zürich

Tel. +41 1 254 7311
Fax +41 1 261 5389
e–mail sekcs@inf.ethz.ch

OBERON

ORDER FORM

Name:

Address:

Telephone Number:

Current available DOS–Oberon Systems:

Standard System: ☐

System 3/Gadgets: ☐

Cheque
or

Credit Card (Euro/Master or VISA):

No.: Expiry date:

Date: Signature:

Oberon™

From the Creators of Pascal and Modula-2

Institute for Computer Systems

ETH Zurich