



Object-Oriented Graphics

SIGGRAPH '88

Course #22

ATLANTA, AUGUST 1988

# **Object-Oriented Graphics**

**SIGGRAPH '88**

**Course #22**

**ATLANTA, AUGUST 2, 1988**

**Peter Wißkirchen**

**(Chair)**

**GMD, Gesellschaft für Mathematik und Datenverarbeitung  
(German National Research Center for Computer Science)**

**Erich L. Rome**

**GMD**

**May 10, 1988**

---

© GMD 1988

No part of this publication may be reproduced, stored in a retrieval system or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior permission of the GMD

---

**Address of the authors:**

**Dr. Peter Wisskirchen**  
Institute for Applied Information Technology  
GMD  
Postfach 1240  
D-5205 St. Augustin 1  
uucp: wissk@gmdzi

**Erich Rome**  
Institute for Applied Information Technology  
GMD  
Postfach 1240  
D-5205 St. Augustin 1  
uucp: eric@gmdzi

---

**Gesellschaft für Mathematik und Datenverarbeitung mbH (GMD)**  
(German National Research Center for Computer Science)  
Schloss Birlinghoven  
Postfach 1240  
D-5205 Sankt Augustin 1  
Tel. +49-2241-14-1

---

# Preface

## Acknowledgements

We thank our colleagues of the research group "man-machine communication" for various comments, especially Klaus Kansy for advice in GKS. We would like to express our gratitude to Dieter Bolz and Rüdiger Kolb for giving us an introduction into the MVC concept. We would also especially like to thank Rüdiger Kolb for elaborating and implementing the Smalltalk examples.

We thank Ursula Bernhard for translating chapter 1 and 2 of this paper and Kai Diestelmeier for correcting the English of the other parts.

We like to thank Symbolics GmbH of Eschborn (West Germany) for their permission to cite from their copyrighted material. The picture on the title page of chapter 3 is a modification of a figure from Symbolics' S-Geometry manual.

The abstract GKS program example on the title page of chapter 2 is taken from [Salmon87], p. 352, by the kind permission of Addison-Wesley GmbH, Bonn.

By the kind permission of Springer-Verlag, Heidelberg, [WISS86b] was reprinted in appendix A.3.

## Trademarks

Apple, Macintosh and MacDraw are trademarks of Apple Computer, Inc.

UNIX is a trademark of AT&T Bell Laboratories, Inc.

CLOS is a trademark of Lucid, Inc.

Symbolics, Symbolics-Lisp, Symbolics Common Lisp, Genera, Dynamic Windows, Document Examiner, S-RENDER, S-DYNAMICS and S-GEOMETRY are trademarks of Symbolics, Inc.

Zetalisp is a registered trademark of Symbolics, Inc.

Xerox, Smalltalk-80, Loops and Common Loops are trademarks of Xerox Corporation.

# Contents

<b>Preface</b>	<b>i</b>
Acknowledgements . . . . .	i
Trademarks . . . . .	i
<b>1 Introduction</b>	<b>1</b>
<b>2 Object-Oriented vs. Classical Approaches</b>	<b>3</b>
2.1 Introduction . . . . .	5
2.2 Object-Oriented Programming - Basic Concepts . . . . .	7
2.2.1 Objects and Methods . . . . .	9
2.2.2 Some Examples of Objects and Messages . . . . .	10
2.2.3 Communication Paths Between Objects . . . . .	14
2.2.4 Object-Oriented Programming - Inheritance . . . . .	17
2.2.5 Object-Oriented Programming Environments . . . . .	21
2.2.6 Summary of the Basic Concepts of Object-Oriented Systems	21
2.3 Inheritance in Application Frameworks . . . . .	23
2.3.1 The Conventional Toolbox . . . . .	23
2.3.2 Object-Oriented Application Frameworks . . . . .	23
2.4 Classical Graphics Systems . . . . .	27
2.4.1 What is to be Understood by Classical Systems? . . . . .	27
2.4.2 Integrating Graphics Kernel Systems into Programming En-	
vironments . . . . .	28
2.4.3 Control Structure . . . . .	29
2.4.4 Coordinate Transformations . . . . .	30
2.4.5 Output Primitives . . . . .	30
2.4.6 Segments . . . . .	30
2.4.7 Segment Hierarchies . . . . .	30
2.4.8 Assignment of Graphics Attributes . . . . .	31
2.4.9 Graphics Input . . . . .	31
2.5 Comparing Classical and Object-Oriented Graphics Systems . . . .	33
2.5.1 Assigning Names to Segments . . . . .	33
2.5.2 Graphics Segments in Different Modules . . . . .	36
2.5.3 Output Primitives as Objects . . . . .	40
2.5.4 Segment Editing . . . . .	42

2.5.5	Assignment of Attributes . . . . .	45
2.5.6	Symmetrical Communication Between Application and User Interface . . . . .	48
2.5.7	Specialization of a Graphics Kernel by Inheritance . . . . .	50
2.6	Modularization of Application and User Interface . . . . .	55
2.6.1	Application Model, Mapping and Basic Functions . . . . .	55
2.6.2	Separation Facilities in Object-Oriented Systems . . . . .	56
2.7	The Model-View-Controller Concept . . . . .	59
2.7.1	Triggering Messages via Dependence Relations . . . . .	59
2.7.2	Communication in the MVC Model . . . . .	60
2.7.3	Prefabricated Classes . . . . .	61
2.7.4	Establishing Communication . . . . .	61
2.7.5	Constructing a Model Class . . . . .	63
2.7.6	Constructing a Controller Class . . . . .	64
2.7.7	Constructing a View Class . . . . .	65
2.7.8	Example . . . . .	67
2.8	New Developments and Open Problems . . . . .	71
2.8.1	Objects and Part-of Hierarchies . . . . .	71
2.8.2	Graphical Constraints . . . . .	75
2.8.3	Graphics and Hybrid Knowledge Representation . . . . .	77
2.8.4	Computer Graphics as a Knowledge Representation Problem	81

<b>3</b>	<b>Object-Oriented Graphics in Lisp Environments</b>	<b>83</b>
3.1	Introduction . . . . .	84
3.2	The Development of Object-Oriented Lisp Programming . . . . .	85
3.3	The New Flavor System . . . . .	86
3.3.1	What's a Flavor? . . . . .	87
3.3.2	Inheritance . . . . .	93
3.3.3	Method Combination . . . . .	96
3.4	Object-Oriented User Interface . . . . .	103
3.4.1	The Lisp Machine Window System . . . . .	103
3.4.2	Static Windows . . . . .	103
3.4.3	Dynamic Windows . . . . .	107
3.4.4	Graphic I/O Primitives . . . . .	115
3.4.5	The Presentation Type System . . . . .	120
3.4.6	Advanced Object-Oriented Graphics . . . . .	128
3.5	Programming with Flavors . . . . .	133
3.5.1	The Object-Centered View . . . . .	133
3.5.2	Object-Oriented Graphics Programming . . . . .	135
3.6	A Simple Object-Oriented Tree Editor . . . . .	137
3.6.1	The Task . . . . .	137
3.6.2	System Layers and Inheritance . . . . .	138
3.6.3	Graphic Objects . . . . .	142
3.6.4	Tree Layout . . . . .	144
3.6.5	Using TED with Other Applications . . . . .	145
3.7	Conclusion . . . . .	146
<b>A</b>	<b>Appendix</b>	<b>147</b>
A.1	A Very Short Introduction to Lisp . . . . .	147
A.2	Flavor Ordering Program Example . . . . .	149
A.3	Article Reprint . . . . .	152
A.4	Smalltalk Example Simple-Counter . . . . .	159
A.5	Smalltalk Example Graphics-Counter . . . . .	164
A.6	References . . . . .	170
	<b>Index</b>	<b>174</b>

# List of Figures

2.1	The object-oriented user interface of MacDraw. . . . .	8
2.2	Graphical representation of an object. . . . .	10
2.3	Message passing to a with b as return value. . . . .	11
2.4	Structure of a class. . . . .	13
2.5	Message passing to a class. . . . .	14
2.6	Message to self. . . . .	15
2.7	Symmetrical communication. . . . .	16
2.8	The class Partner. . . . .	17
2.9	Inheritance tree. . . . .	18
2.10	Searching for inherited messages. . . . .	18
2.11	Message to super. . . . .	19
2.12	Inheritance graph. . . . .	20
2.13	Addition of code by subclassing. . . . .	24
2.14	Fundamental concept of application frameworks. . . . .	25
2.15	GKS layer model. . . . .	28
2.16	Limited control structure of GKS. . . . .	29
2.17	Objects are not accessible by the application programmer. . . . .	35
2.18	Segments as instances of class Segment. . . . .	36
2.19	The PHIGS model. . . . .	43
2.20	Editing subclasses of Segment - an open problem. . . . .	45
2.21	Assignment of attributes. . . . .	47
2.22	Classical input model. . . . .	49
2.23	Object-oriented input model. . . . .	50
2.24	Specialization of a geometrical object. . . . .	51
2.25	Different groups of methods for class Cylinder. . . . .	52
2.26	Problems with hierarchical inheritance. . . . .	53
2.27	Mixture of three different types of code in the application. . . . .	56
2.28	Six communication paths may be senseful. . . . .	60
2.29	The MVC classes. . . . .	62
2.30	The MVC instances. . . . .	62
2.31	Schematic structure of MyModel. . . . .	64
2.32	Structure of MyController. . . . .	64
2.33	Structure of MyView. . . . .	66
2.34	Example of a part-of hierarchy (acyclic graph). . . . .	72



2.35	Direct access (flattening the hierarchy).	73
2.36	Information hiding and access of parts.	74
2.37	Constraint net.	75
2.38	Semantic network.	78
2.39	Hybrid knowledge representation.	80
2.40	Graphical data structure as semantic network.	82
3.1	Development of Lisp and its object-oriented extensions.	86
3.2	Traversal of a flavor "tree".	94
3.3	Pseudo-tree with duplicate flavor occurrence.	95
3.4	Flavor components of <code>tv:window</code> .	105
3.5	Static window example.	107
3.6	Flavor components of <code>dw:dynamic-window</code> .	108
3.7	Output history and viewport of a dynamic window.	109
3.8	Dynamic window with scroll bars and label.	112
3.9	Dynamic window with scroll bars and label, version 2.	113
3.10	Dynamic window with scroll bars and label, version 3.	113
3.11	Dynamic window instance and related instances.	114
3.12	Graphic primitives of static windows.	116
3.13	New graphic primitives.	119
3.14	Presentation of an integer as a circle.	124
3.15	Flavors dependent on <code>dw::presentation</code> .	125
3.16	Presentation flavor instances for presented integer 7.	126
3.17	Presentations as link between internal and external representation.	127
3.18	S-Geometry, a programmable 3D-graphics editor.	129
3.19	Some of S-Geometry's flavors.	132
3.20	Protocols for Output Streams.	135
3.21	Tree editor display example.	138
3.22	Flavors of the Tree Editor application.	139
3.23	Blinkers highlighting a path in a tree.	142
3.24	Changed node representation and blinker area.	143
3.25	Node and external representation Flavors.	144