Figure 1.1 The layers of NeXTSTEP 1
Figure 2.1 In procedural programming, the functions are kept separate from the data they operate on 8
Figure 2.2 An object is a collection of data (instance variables) along with the functions (methods) that can access the data 9
Figure 2.3 Accessing an instance variable through one of the methods 9
Figure 2.4 Once a class is defined, it can be used to create instances 11
Figure 2.5 A subclass inherits the methods and instance variables defined in the superclass 14
Figure 2.6 A message expression has two parts 18
Figure 2.7 Polymorphism allows identically named methods in more than one class 19
Figure 2.8 The higher classes are more generalized than the lower classes 21
Figure 2.9 The interface file declares the instance variables and methods which the class contains 31
Figure 2.10 The implementation file contains the code for the methods defined in the class' interface file 34
Figure 2.11 The class object has two message dispatch tables that contain the addresses of the instance and class methods 37
Figure 2.12 Objective-C searches for the appropriate method in the dispatch table 39
Figure 2.13 Dynamic binding defers associating the receiver to the method until runtime 40
Figure 2.14 **self** is the current object and **super** is the superclass 43
Figure 2.15 Objective-C starts at the **Object** class to allocate the memory for a newly created instance 45
Figure 2.16 The contents of **theWindow** and **name** are completely external to **theDocument 48**
Figure 2.17 An object should free the memory block an instance variable is pointing to before freeing the instance variable 50
Figure 3.1 Object-oriented design is composed of activities rather than steps 55
Figure 3.2 **AbstractShape** has no collaborators 64
Figure 3.3 The **Rectangle** class has no collaborators 65
Figure 3.4 The **Square** class has no collaborators 65
Figure 3.5 The **List** class has no collaborators 66
Figure 3.6 In a message diagram, the classes are depicted as boxes, and the lines and their directions depict the messages 69
Figure 3.7 The **Object** class is always the root class of an Objective-C application 70
Figure 4.1 The major classes in the Application Kit 100
Figure 4.2 The Common Classes hierarchy 102
Figure 4.3 The CRC card for the **Menu** class 106
Figure 4.4 The CRC card for the **Window** class 107
Figure 4.5 The CRC card for the **Application** class 107

Figure 4.6 The message diagram for **AppKitDemo** 111
Figure 4.7 **AppKitDemo** in execution 114
Figure 4.8 **AppKitDemo** with a **Quit** option 119
Figure 5.1 The NeXTSTEP development cycle 125
Figure 5.2 Creating a new project with ProjectBuilder 127
Figure 5.3 The Files accessory view of ProjectBuilder 127
Figure 5.4 The Attributes accessory view of ProjectBuilder 129
Figure 5.5 The Build accessory view 129
Figure 5.6 The initial screen in InterfaceBuilder 130
Figure 5.7 The File Window contains the resources 131
Figure 5.8 The Palettes Window contains more than one palette 132
Figure 5.9 InterfaceBuilder automatically creates menu options for the Main Menu 132
Figure 5.10 Adding a menu option to the Main Menu 133
Figure 5.11 Adding a panel to the application 134
Figure 5.12 Placing a textfield in the Info panel 134
Figure 5.13 Displaying the Attributes Inspector for the Info panel 135
Figure 5.14 Editing the title of a panel 136
Figure 5.15 Editing the text of a textfield 137
Figure 5.16 Editing the text of a menu option 137
Figure 5.17 Making a connection to the Main Window 139
Figure 5.18 Displaying an already existing connection. 140
Figure 5.19 Switching to **Test Mode** in InterfaceBuilder 141
Figure 5.20 Building an application in ProjectBuilder 144
Figure 5.21 The components of an application 147
Figure 6.1 The framework for a NeXTSTEP application 151
Figure 6.2 Clicking on the **Hide** menu option sends a **hide:** message to **NXApp** (the target) 153
Figure 6.3 Some sample buttons 155
Figure 6.4 **ControlDemo** during execution 158
Figure 6.5 The components of **ControlDemo** 159
Figure 6.6 Some sample sliders 161
Figure 6.7 The granularity of a slider 162
Figure 6.8 **ControlDemo** with a slider and a button 164
Figure 6.9 Some sample textfields 165
Figure 6.10 A converter application with two textfields 166

Figure 6.11 **ControlDemo** with a button, a slider, and a textfield 167
Figure 6.12 The slider is the textfield's target and the textfield is the slider's target 168
Figure 6.13 A sample form 169
Figure 6.14 **ControlDemo** with all the controls 171
Figure 6.15 **ControlDemo** with the button as the form's target 172
Figure 6.16 The **windowWillClose:** method in action 180
Figure 6.17 **Money** with six textfields and a button 182
Figure 6.18 The CRC card for the **MoneyConverter** class 184
Figure 6.19 The message diagram for **Money** **185**
Figure 6.20 The hierarchy graph for **Money** **185**
Figure 6.21 Adding fields to a form 186
Figure 6.22 Labeling the form with the appropriate fields 187
Figure 6.23 Subclassing **Object** to create **MoneyConverter** **188**
Figure 6.24 Adding the **moneyForm** outlet to the **Converter** class 189
Figure 6.25 Adding the **convert:** method 190
Figure 6.26 Instantiating the **MoneyConverter** class 191
Figure 6.27 Connecting the objects in **Money.nib** **191**
Figure 6.28 Connecting to the form instead of to the formcell 192
Figure 6.29 Adding an arrow icon to the button 194
Figure 6.30 Generating the template files with the **Unparse** command 194
Figure 6.31 Setting the moneyconverter as the delegate of the Main Window 198
Figure 6.32 Determining the size of a window with the Size Inspector 199
Figure 6.33 Parsing in a class updates the outlets and actions for the class in InterfaceBuilder 200
Figure 6.34 Setting the moneyconverter as the delegate of the application object 201
Figure 6.35 Using the autosizing features in the Size Inspector 204
Figure 6.36 Setting the autosizing characteristics of the button and the form 205
Figure 7.1. Coordinates in the base system and screen system 210
Figure 7.2. A view's location inside its window 213
Figure 7.3. Drawing order of views in a window 215
Figure 7.4. A view's frame rectangle can be outside of its superview's 216
Figure 7.5. Drawing a shape with PostScript 224
Figure 7.6. Execution of a typical PostScript command 225
Figure 7.7. Execution of **square_outline.ps** 226
Figure 7.8. Execution of **black_square.ps** 227

Figure 7.9. Execution of **circle.ps** **228**
Figure 7.10. Rotating the axes 230
Figure 7.11. Producing a shadow effect in PostScript 231
Figure 7.12. How **pswraps** fits in the program structure 235
Figure 7.13. A sample wraps function 236
Figure 7.14. An example of instance drawing 238
Figure 7.15. Correct and incorrect use of instance drawing 241
Figure 7.16. Updating a view in response to the user's actions 244
Figure 7.17. A preliminary interface for **Shapes** **245**
Figure 7.18. A more refined interface for **Shapes** **246**
Figure 7.19. The CRC card for the **ShapeView** class 247
Figure 7.20. The CRC card for the **SquareView** class 247
Figure 7.21. The CRC card for the **CircleView** class 248
Figure 7.22. The message diagram for **Shapes** **251**
Figure 7.23. The hierarchy graph for **Shapes** **252**
Figure 7.24. Instantiating the **SquareView** class 254
Figure 7.25. Creating a matrix of two sliders 255
Figure 7.26. The user interface with the sliders labeled 256
Figure 7.27. Grouping objects with a box 257
Figure 7.28. The user interface with the controls defined 259
Figure 7.29. Making the connections in **Shapes.nib** **260**
Figure 8.1 The precedence order for building the registration table 285
Figure 8.2 Using **NXUpdateDefault()** to update a default's value in the registration table 288
Figure 8.3 The views in a Preferences panel is controlled by a popuplist 289
Figure 8.4 InterfaceBuilder doesn't allow connections between **.nib** files since this would violate encapsulation 291
Figure 8.5 An object can appear as an instance in one **.nib** file and as the **File's Owner** in another **.nib** file 291
Figure 8.6 A switchbutton is more appropriate than a textfield for options that only have two possible values 298
Figure 8.7 The CRC card for the **PrefsController** class 304
Figure 8.8 The CRC card for the **SwitchView** class 304
Figure 8.9 The updated CRC card for the **MoneyConverter** class 305
Figure 8.10 The message diagram for **Money** **308**
Figure 8.11 The updated class hierarchy graph for **Money** **309**

Figure 8.12 Adding header files to a project by dragging them from the Workspace 311
Figure 8.13 Setting the class of the **File's Owner 311**
Figure 8.14 Editing the entries in a popuplist 312
Figure 8.15 The Preferences panel with the switchview 313
Figure 8.16 Connecting all the objects in **Prefs.nib 313**
Figure 8.17 Enabling the **Preferences** menu option 315
Figure 8.18 Connecting the objects in **Money.nib** 316
Figure 8.19 Centering a view in its superview's coordinate system 329
Figure 8.20 Drawing a bezeled line 330
Figure 9.1 A regular coordinate system vs. a flipped coordinate system 338
Figure 9.2 The key window and the main window may or may not be the same window 343
Figure 9.3 The search order when a target is not explicitly set 345
Figure 9.4 A text object can grow beyond the boundaries of its superview 346
Figure 9.5 The components of a scrollview 347
Figure 9.6 A typical savepanel 351
Figure 9.7 A typical openpanel 357
Figure 9.8 Retrieving the selected filename(s) 360
Figure 9.9 A preliminary interface for **Words 368**
Figure 9.10 The CRC card for the **Document** class 369
Figure 9.11 The CRC card for the **TextController** class 370
Figure 9.12 The CRC card for the **SavePanel** class 370
Figure 9.13 The CRC card for the **OpenPanel** class 371
Figure 9.14 The CRC card for the **Window** class 371
Figure 9.15 The message diagram for **Words 374**
Figure 9.16 The hierarchy graph for the custom classes in **Words 374**
Figure 9.17 Adding a **Command-w** keyboard alternative to the **Close** menu option 375
Figure 9.18 Making the connections in **Words.nib 379**
Figure 9.19 Adding a scrollview to **Document.nib 381**
Figure 9.20 Connecting the objects in **Document.nib 382**
Figure 9.21 Double-clicking on a file from the Workspace sets off a complex series of events 403
Figure 9.22 Adding an icon and changing the extension for an application's documents 405
Figure 9.23 Changing the application's icon 406
Figure 9.24 Setting **Words** as the default application for **.word** documents 407
Figure 10.1 The on-line help for the keyboard in the Preferences application 412

Figure 10.2 Enabling **Developer Mode** in Edit 413
Figure 10.3 Inserting a link in a document 414
Figure 10.4 Displaying and editing a link 415
Figure 10.5 Inserting a marker in a file 416
Figure 10.6 The **New.rtf** file after it has been modified 420
Figure 10.7 The **New.rtf** file with the links and marker 421
Figure 10.8 The **Open.rtf** file with the links and marker 422
Figure 10.9 The **Save.rtf** file with the links and marker 423
Figure 10.10 Attaching help to the **New** menu option 424
Figure A.1 The NeXTSTEP login window 431
Figure A.2 The initial screen after logging in 432
Figure A.3 NeXTSTEP windows and icons 433
Figure A.4 Displaying a submenu 434
Figure A.5 Miniaturizing a window 435
Figure A.6 Display the contents of a folder and selecting a file 436
Figure A.7 Scrolling through the **README.rtf** file 437
Figure A.8 The title bar contains the filename and the path 438
Figure A.9 To close a window, click on the close button, located in the upper right hand of the title bar 439
Figure A.10 Double-click a word to select it 440
Figure A.11 Copying and pasting the text 440
Figure A.12 To get our attention, NeXTSTEP places the alert panel above the other windows 441
Figure A.13 Edit displays a savepanel to ask us where to save the file 442
Figure A.14 To recycle a folder, drag the folder to the Recycler icon 443
Figure A.15 Displaying the contents of the Recycler 443
Figure A.16 Renaming the folder from **NewFolder** to **JunkContainer 444**
Figure A.17 Recovering the **junk** folder from the Recycler 445
Figure B.1 Using the implicit Expansion Dictionary 448
Figure B.2 Adding an entry to the Expansion Dictionary 450
Figure B.3 Using an entry from the Expansion Dictionary 450
Figure B.4 Edit displays only the name of the method when the method is contracted 453
Figure B.5 Indexing an unindexed target in Librarian 455
Figure B.6 The initial screen of HeaderViewer (the Browser view) 458
Figure B.7 The Find Control Options panel of HeaderViewer 461

Figure B.8 The List view of HeaderViewer 462
Figure B.9 Appending the current directory in Terminal by dragging from the Workspace 469
Figure D.1 Searching an entire folder for a given string 503
Figure D.2 Including debugging information in ProjectBuilder 505
Figure D.3 Clicking on the **Debug** button in ProjectBuilder produces a gdb shell window 506
Figure D.4 The Gdb panel in Edit 507
Figure D.5 Click on **self** to display its contents 515
Figure D.6 The value of **popUpButton**'s and **switchView**'s **window** outlets should be identical since they refer to the same window 516
Figure D.7 Comparing the value of **switchView**'s **accessoryView** outlet against **truncateSwitch** **517**
Figure D.8 Checking the values of the **rate** array in the moneyconverter 520