

Самарский Государственный Технический Университет

Кафедра “Информационные технологии”

УЧЕБНОЕ ПОСОБИЕ

**на тему: «Проектирование и дизайн пользовательского
интерфейса»
(материалы одноименной книги А.К.Гуляева и В.А.Машина)**

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ.....	4
1. ПОНЯТИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА И ТРЕБОВАНИЯ К НЕМУ	5
1.1. ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ: МОСТ МЕЖДУ ЧЕЛОВЕКОМ И КОМПЬЮТЕРОМ	5
1.2. ОСНОВНЫЕ ПРИНЦИПЫ РАЗРАБОТКИ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА	7
1.3. СТАНДАРТИЗАЦИЯ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА	11
2. ПРОЕКТИРОВАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА	13
2.1. ЖИЗНЕННЫЙ ЦИКЛ ПРОГРАММНОГО ПРОДУКТА	13
2.2. ЭТАПЫ ПРОЕКТИРОВАНИЯ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА	17
2.2.1. ВЫБОР СТРУКТУРЫ ДИАЛОГА	18
2.2.2. РАЗРАБОТКА СЦЕНАРИЯ ДИАЛОГА	24
2.2.3. ВИЗУАЛЬНЫЕ АТРИБУТЫ ОТОБРАЖАЕМОЙ ИНФОРМАЦИИ	28
3. ПРОЕКТИРОВАНИЕ ГРАФИЧЕСКОГО ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА	31
3.1. ОСОБЕННОСТИ ГРАФИЧЕСКОГО ИНТЕРФЕЙСА	31
3.2. ОБЪЕКТНЫЙ ПОДХОД К ПРОЕКТИРОВАНИЮ ИНТЕРФЕЙСА	33
3.3. КОМПОНЕНТЫ ГРАФИЧЕСКОГО ИНТЕРФЕЙСА	35
3.4. ВЗАИМОДЕЙСТВИЕ ПОЛЬЗОВАТЕЛЯ С ПРИЛОЖЕНИЕМ	40
3.5. ОБЩИЕ ПРАВИЛА ВЗАИМОДЕЙСТВИЯ С ОБЪЕКТАМИ	47
3.6. ОПЕРАЦИИ ПЕРЕСЫЛКИ И СОЗДАНИЯ ОБЪЕКТОВ	53
3.6.1. ОПЕРАЦИИ ПЕРЕСЫЛКИ	53
3.6.2. ОПЕРАЦИИ СОЗДАНИЯ НОВЫХ ОБЪЕКТОВ	64
3.6.3. ОПЕРАЦИИ СВЯЗЫВАНИЯ ОБЪЕКТОВ	65
4. ОКНА И ПИКТОГРАММЫ	66
4.1. ПРОЕКТИРОВАНИЕ ПИКТОГРАММ	67
4.2. ПЕРВИЧНЫЕ ОКНА	68
4.2.1. СТРУКТУРА ПЕРВИЧНОГО ОКНА	68
4.2.2. ОСНОВНЫЕ ОПЕРАЦИИ С ОКНАМИ	71
4.2.3. ИСПОЛЬЗОВАНИЕ ПОДОКОН	77
4.2.4. МНОГОДОКУМЕНТНЫЙ ИНТЕРФЕЙС (MDI)	79
4.2.5. ВЫБОР МОДЕЛИ ОКНА	87
4.3. ВТОРИЧНЫЕ ОКНА	88
4.3.1. ОСНОВНЫЕ СВОЙСТВА ВТОРИЧНЫХ ОКОН	88
4.3.2. ПАНЕЛИ СВОЙСТВ И КОНТРОЛЯ ПАРАМЕТРОВ	94
4.3.3. ДИАЛОГОВЫЕ ПАНЕЛИ	98
Интерпретация системой имени файла	103
4.3.4. ДРУГИЕ ТИПЫ ВТОРИЧНЫХ ОКОН	105
5. ПРОЕКТИРОВАНИЕ ЭЛЕМЕНТОВ УПРАВЛЕНИЯ	107
5.1. МЕНЮ	108
5.1.1. ГЛАВНОЕ МЕНЮ ОКНА И ВЫПАДАЮЩИЕ МЕНЮ	108
5.1.2. ВСПЛЫВАЮЩИЕ МЕНЮ	111
5.1.3. КАСКАДНЫЕ МЕНЮ	116
5.1.4. ЗАГОЛОВОК МЕНЮ	117
5.1.5. ПУНКТЫ МЕНЮ	117
5.2. КНОПКИ	121
5.2.1. КНОПКИ УПРАВЛЕНИЯ	121
5.2.2. ПЕРЕКЛЮЧАТЕЛИ	124
5.2.3. ФЛАЖКИ	125
5.3. СПИСКИ	126
5.3.1. СПИСОК ЕДИНИЧНОГО ВЫБОРА	128
5.3.2. ВЫПАДАЮЩИЙ СПИСОК	128
5.3.3. РАСШИРЕННЫЙ СПИСОК И МНОЖЕСТВЕННОГО ВЫБОРА	130
5.3.4. МОДИФИЦИРУЕМЫЙ СПИСОК	130
5.3.5. МОДИФИЦИРУЕМОЕ ДЕРЕВО	132
5.4. ТЕКСТОВЫЕ ОБЛАСТИ	133
5.4.1. ТЕКСТОВЫЕ ПОЛЯ	133
5.4.2. МНОГОСТРОЧНОЕ ТЕКСТОВОЕ ПОЛЕ	134
5.4.3. КОМБИНИРОВАННЫЙ СПИСОК	135
5.4.4. ВЫПАДАЮЩИЙ КОМБИНИРОВАННЫЙ СПИСОК	135
5.4.5. ДИСКРЕТНОЕ ТЕКСТОВОЕ ПОЛЕ	136
5.4.6. СТАТИЧЕСКИЕ ТЕКСТОВЫЕ ОБЛАСТИ	137
5.4.7. ПОЛЕ НАЗНАЧЕНИЯ ГОРЯЧИХ КЛАВИШ	137
5.5. ПАНЕЛЬ ИНСТРУМЕНТОВ И СТРОКА СОСТОЯНИЯ	138
5.6. ДРУГИЕ ЭЛЕМЕНТЫ ГРАФИЧЕСКОГО ИНТЕРФЕЙСА	141
5.6.1. ГРУППИРУЮЩИЙ БЛОК	142
5.6.2. ЗАГОЛОВКИ СТОЛБЦОВ	142
5.6.3. ЭТИКЕТКА ВКЛАДКИ	143
5.6.4. ПОЛОСЫ ПРОКРУТКИ	143
5.6.5. ПОЛЗУНКОВЫЙ РЕГУЛЯТОР	145
5.6.6. ИНДИКАТОР СОСТОЯНИЯ ПРОЦЕССА	146
5.6.7. ВСПЛЫВАЮЩАЯ ПОДСКАЗКА	146
5.6.8. КОЛЛЕКЦИИ	147
5.6.9. ОБЛАСТЬ СООБЩЕНИЙ	147
5.7. ВЫБОР ВИЗУАЛЬНЫХ АТРИБУТОВ ОТОБРАЖАЕМОЙ ИНФОРМАЦИИ	148
5.7.1. КОМПОЗИЦИЯ И ОРГАНИЗАЦИЯ	149
5.7.2. ЦВЕТ	150
5.7.3. ШРИФТ	152
5.7.4. «МНОГОМЕРНОСТЬ» ЭКРАНА	152
5.7.5. ПРОСТРАНСТВЕННОЕ РАЗМЕЩЕНИЕ ВИЗУАЛЬНЫХ ЭЛЕМЕНТОВ	152

5.7.6. ВИЗУАЛИЗАЦИЯ ВЫПОЛНЯЕМЫХ ОПЕРАЦИЙ.....	154
5.8. ТРИ СЛУЧАЯ ИЗ ЖИЗНИ GUI.....	156
6. ПРОЕКТИРОВАНИЕ СРЕДСТВ ПОДДЕРЖКИ ПОЛЬЗОВАТЕЛЯ.....	158
6.1. ОКНО СООБЩЕНИЕ.....	158
6.2. КОНТЕКСТНАЯ ПОМОЩЬ.....	162
6.2.1. КОМАНДА ЧТО ЭТО?.....	163
6.2.2. ВСПЛЫВАЮЩАЯ ПОДСКАЗКА.....	164
6.2.3. ВЫВОД СООБЩЕНИЙ В СТРОКЕ СОСТОЯНИЯ.....	165
6.2.4. КНОПКА СПРАВКА.....	166
6.3. ПРОБЛЕМНО-ОРИЕНТИРОВАННАЯ ПОМОЩЬ.....	166
6.4. СПРАВОЧНИК.....	168
6.5. МАСТЕРА.....	172
6.6. СРЕДСТВА ОБУЧЕНИЯ ПОЛЬЗОВАТЕЛЯ.....	175
6.7. СРЕДСТВА АДАПТАЦИИ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА.....	178
7. ПРОЕКТИРОВАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА: ОТ ОБЩЕГО К ЧАСТНОМУ.....	182
7.1. ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС WEB-ПРИЛОЖЕНИЙ.....	182
7.2. WEB-СТРАНИЦЫ И САЙТЫ.....	184
7.3. ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС СИСТЕМ РЕАЛЬНОГО ВРЕМЕНИ.....	193
8. СРЕДСТВА РЕАЛИЗАЦИИ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА.....	202
8.1. КЛАССИФИКАЦИЯ СРЕДСТВ РАЗРАБОТКИ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА.....	202
8.2. ИНСТРУМЕНТЫ РЕАЛИЗАЦИИ СРЕДСТВ ПОДДЕРЖКИ ПОЛЬЗОВАТЕЛЯ.....	208
8.3. СРЕДСТВА РАЗРАБОТКИ WEB-ДОКУМЕНТОВ.....	212
9. ПРОДОЛЖЕНИЕ СЛЕДУЕТ, ИЛИ ТЕНДЕНЦИИ И ПЕРСПЕКТИВЫ РАЗВИТИЯ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА.....	217
10. ПРИЛОЖЕНИЕ.....	222
11. ГЛОССАРИИ.....	229
ЛИТЕРАТУРА.....	239

ПРЕДИСЛОВИЕ

Еще не так давно (лет десять назад) программисты могли использовать весьма ограниченный арсенал средств взаимодействия пользователей с создаваемыми программами. Как правило, такое взаимодействие заключалось в обмене текстовыми сообщениями либо псевдографическими изображениями. При этом «промышленное» программное изделие в основном соответствовало действовавшему в то время государственным стандартам, которые предусматривали наличие соответствующей программной документации. С ее помощью пользователи программного изделия могли уяснить все особенности работы с ним. Более того, сами эти пользователи были если и не программистами, то по крайней мере людьми, достаточно хорошо знакомыми с вычислительной техникой.

В силу указанных причин проблемы, связанные с разработкой и освоением средств общения пользователей с программами, решались весьма просто и, как правило, достаточно стандартным образом.

За последние годы ситуация коренным образом изменилась.

Во-первых, с распространением персональных компьютеров невероятно возросло число пользователей ЭВМ, в том числе не имеющих даже начальных знаний в области вычислительной техники.

Во-вторых, значительно увеличилось и число программирующих пользователей ЭВМ, как имеющих соответствующую базовую подготовку, так и «самоучек»; в их распоряжении имеются мощные средства разработки, которые позволяют создавать программы с практически неограниченными интерактивными возможностями.

В-третьих, имеющиеся государственные стандарты на разработку программных продуктов безнадежно устарели и их влияние на технологию программирования практически равно нулю.

И, наконец, наличие такого уникального (и универсального) средства коммуникации, как Интернет, позволяет всем желающим выставлять на всеобщее обозрение плоды своего творчества, вне зависимости от их качества и предназначения; при этом каждый автор надеется на высокую оценку своего труда со стороны коллег и потенциальных пользователей.

Все это привело к тому, что на свет стало появляться все больше программных продуктов, которые на самом деле таковыми не являются. Назвать их «уродцами» язык не поворачивается, поскольку практически все они снабжены ярким и, на первый взгляд, весьма впечатляющим интерфейсом в виде многочисленных и разнообразных кнопок, пиктограмм, переключателей и т.п. Это, скорее, бабочки-однодневки, которые так резво машут своими пестрыми крылышками, что никак не удастся рассмотреть, что же они все-таки умеют делать.

С другой стороны, даже действительно полезные программы, но снабженные неудачным интерфейсом, остаются невостребованными. Более того, известны случаи, когда достаточно крупные проекты были отклонены заказчиком только из-за того, что разработчиком не были своевременно и качественно решены вопросы, связанные с пользовательским интерфейсом.

Важность грамотного проектирования и реализации пользовательского интерфейса осознана ведущими фирмами-производителями программного обеспечения уже давно. Об этом говорит тот факт, что проектирование пользовательского интерфейса рассматривается ими как отдельный процесс в рамках жизненного цикла любого программного продукта. Причем и сам этот процесс, и качество создаваемого интерфейса регламентируются целым рядом соответствующих стандартов. Те

программные продукты, которые им не удовлетворяют, практически не имеют шансов на «выживание».

В настоящее время на российском книжном рынке недостаточно литературы, посвященной разработке пользовательского интерфейса. Уже нельзя ориентироваться на устаревающие стандарты, а также устаревшие версии системного и прикладного программного обеспечения. Недостаток литературы по данному вопросу приводит к тому, что даже в среде профессиональных программистов отсутствует единая трактовка терминов и понятий, относящихся к пользовательскому интерфейсу.

Необходимо обобщить зарубежный и отечественный опыт в создании пользовательского интерфейса с учетом последних достижений в области графического интерфейса пользователя (GUI) с учетом особенностей применения наиболее распространенных средств визуального программирования — Visual C++, Visual Basic, Delphi.

1. ПОНЯТИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА И ТРЕБОВАНИЯ К НЕМУ

1.1. ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ: МОСТ МЕЖДУ ЧЕЛОВЕКОМ И КОМПЬЮТЕРОМ

Если водитель, садясь за руль незнакомого автомобиля, знает, куда вставить ключ зажигания, и где находятся педали газа и тормоза, то пользователи новых программных продуктов до последнего времени были лишены подобного преимущества.

Автомобилестроители опираются на многолетние традиции, главное содержание которых — «все на благо водителя». Тут и ремни безопасности, и зеркала заднего вида, и, наконец, кондиционер в салоне.

Разработчики же программ все еще ориентируются в основном на личный опыт и, в большинстве своем, относятся к пользователям, скорее, как профессиональный водитель к начинающему любителю: не только норовит прижать его к обочине, но еще и сопровождает это неприличным жестом.

И лишь в последние годы, когда число владельцев персональных компьютеров стало приближаться к числу автомобилистов, ситуация несколько изменилась.

Изменения заключаются в том, что создатели программных продуктов стали пытаться, во-первых, поставить себя на место потенциального пользователя, и, во-вторых, унифицировать «педали». Однако эти новые веяния затронули, прежде всего, ведущие фирмы-разработчики программного обеспечения (ПО), и практически не повлияли на стиль работы программистов-одиночек или небольших коллективов. Хотя указанные категории разработчиков ПО пользуются достаточно ограниченным набором инструментальных средств, создаваемые ими программные продукты весьма заметно различаются по организации взаимодействия с пользователем. При этом различия проявляются как на уровне внешнего оформления интерактивных компонентов приложения, так и на уровне принципов (взглядов разработчика), положенных в основу реализации этих компонентов.

Хорошо это или плохо?

С одной стороны, любая программа — это результат творчества ее создателя, в значительной степени отражающий его субъективные взгляды, предпочтения, художественный вкус и т.д.; и чем опытнее программист, тем ярче проявляется его индивидуальность в каждой новой программе.

С другой стороны, пользователь ПК — это не посетитель художественной выставки, и его меньше всего интересует личность программиста и то душевное состояние, в котором он находился, создавая тот или иной программный продукт. И даже если пользователь захочет получить эстетическое наслаждение и воспользуется для этого программой «По залам Эрмитажа», все его внимание будет сосредоточено на художественных достоинствах экспонатов музея, а не кнопок, позволяющих «переходить» из одного зала в другой.

Итак, что же такое пользовательский интерфейс: способ самовыражения разработчика или «возжи», с помощью которых пользователь управляет «лошадиными силами» своего компьютера?

А может это просто картинка, которую формирует программа в процессе своей работы под воздействием тех или иных (в том числе случайных) факторов?

И то, и другое, и третье! Сложность этого понятия породила множество определений, которые менялись в процессе развития самого пользовательского интерфейса. Обобщив все то, что было сказано и написано ранее о пользовательском интерфейсе, мы предлагаем следующее определение.

Пользовательский интерфейс — это совокупность информационной модели проблемной области, средств и способов взаимодействия пользователя с информационной моделью, а также компонентов, обеспечивающих формирование информационной модели в процессе работы программной системы.

Под информационной моделью понимается условное представление проблемной области, формируемое с помощью компьютерных (визуальных и звуковых) объектов, отражающих состав и взаимодействие реальных компонентов проблемной области.

Средства и способы взаимодействия с информационной моделью определяются составом аппаратного и программного обеспечения, имеющегося в распоряжении пользователя, и от характера решаемой задачи. Например, для пользователя, который хочет переписать файл с дискеты на жесткий диск, такими средствами являются устройства ввода-вывода (клавиатура, мышь и экран монитора) и два дисководов с дисками. А вот для пользователя, который пытается установить собственные значения параметров BIOS, перечень доступных средств существенно шире. Причем, большинство имеющихся ограничений, воспринимаются и разработчиками и пользователями как «осознанная необходимость», поскольку практически все они носят объективный характер.

Совсем другое дело — та часть интерфейса, которая относится к программным средствам.

Во-первых, для программы значительно сложнее сформулировать **объективные** требования по составу и компоновке органов управления; зачастую не только пользователи, но и сами разработчики не могут объяснить, почему программа имеет именно такие «рычаги» и «педали». Во-вторых, их перечень значительно шире, а состав изменяется во много раз **динамичнее**, чем состав аппаратных средств компьютера.

Весьма распространенной является ситуация, когда программы, равноценные по назначению и функциональным возможностям, оказываются совсем разными по организации взаимодействия с пользователем. При этом совсем не обязательно интерфейс какой-то из программ будет хуже, он просто будет **другим**. И если по какой-то причине знакомая программа окажется недоступной, освоение новой придется начинать практически с нуля.

Значительно большие потери может понести пользователь, которому предстоит либо выбрать одну из незнакомых программ, либо перейти на новую версию уже используемой программы.

В первом случае выбор может быть сделан в пользу менее функциональной и менее надежной программы, но обладающей более привлекательным (с субъективной точки зрения) интерфейсом.

Во втором же случае незнакомый интерфейс новой версии может оказаться психологическим барьером, не преодолев который пользователь так и не сможет воспользоваться преимуществами новой версии. Яркий пример такой ситуации — неожиданно медленный (для Microsoft) переход пользователей от Windows 3.* к Windows 9*.

Таким образом, эффективность работы пользователя определяется не только функциональными возможностями имеющихся в его распоряжении аппаратных и программных средств, но и доступностью для пользователя этих возможностей. В свою очередь, полнота использования потенциальных возможностей имеющихся ресурсов зависит от качества пользовательского интерфейса.

Если надо напечатать с помощью компьютера пригласительные билеты на юбилей, придется воспользоваться текстовым или графическим редактором. Все редакторы «общего пользования» позволяют выполнять примерно один и тот же перечень операций, но весь вопрос в том, как они это делают и каким представлял себе разработчик потенциального пользователя своего продукта. Редактор с неудачным интерфейсом может потребовать от пользователя знакомства с совершенно новыми для него терминами, такими как «лигатура» и «кегель», а после каждого неудачного действия заставит возобновить работу с самого начала. Работа с таким редактором может закончиться тем, что юбиляр предпочтет купить пригласительные билеты в магазине.

Главный вывод заключается в том, что **качество пользовательского интерфейса** является самостоятельной характеристикой программного продукта, сопоставимо по значимости с такими его показателями, как **надежность** и **эффективность** использования вычислительных ресурсов.

Важное следствие: разработчик приложения должен знать, что такое хороший интерфейс, и как его построить.

1.2. ОСНОВНЫЕ ПРИНЦИПЫ РАЗРАБОТКИ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

Создание качественного интерфейса требует значительно большего, чем просто соблюдение некоторых инструкций. Оно предполагает реализацию принципа «интересы пользователя превыше всего» и соответствующую методологию разработки всего программного продукта. В англоязычной литературе для описания такого подхода используется термин User-centered Design -UCD («Разработка, ориентированная на пользователя»). Эта технология, кроме всего прочего, предполагает как можно более раннее Проектирование интерфейса с последующим его развитием в процессе разработки самого программного продукта.

Основное достоинство хорошего интерфейса пользователя заключается в том, что ***пользователь всегда чувствует, что он управляет программным обеспечением, а не программное обеспечение управляет им.***

Для создания у пользователя такого ощущения «внутренней свободы» интерфейс должен обладать целым рядом свойств, рассмотренных ниже.

ЕСТЕСТВЕННОСТЬ ИНТЕРФЕЙСА

Естественный интерфейс — такой, который не вынуждает пользователя существенно изменять привычные для него способы решения задачи. Это, в частности, означает, что сообщения и результаты, выдаваемые приложением, не должны тре-

бовать дополнительных пояснений. Целесообразно также сохранить систему обозначений и терминологию, используемые в данной предметной области.

Использование знакомых пользователю понятий и образов (метафор) обеспечивает интуитивно понятный интерфейс при выполнении его заданий. Вместе с тем, используя метафоры, не надо ограничивать их машинную реализацию полной аналогией с одноименными объектами реального мира. Например, папка на Рабочем столе Windows может использоваться для хранения целого ряда других объектов.. Метафоры являются своего рода «мостиком», связывающим образы реального мира с теми действиями и объектами, которыми приходится манипулировать пользователю при его работе на компьютере; они обеспечивают «узнавание», а не «вспоминание». Пользователи запоминают действие, связанное со знакомым объектом, более легко, чем они запомнили бы имя команды, связанной с этим действием.

СОГЛАСОВАННОСТЬ ИНТЕРФЕЙСА

Согласованность позволяет пользователям переносить имеющиеся знания на новые задания, осваивать новые аспекты быстрее, и благодаря этому фокусировать внимание на решаемой задаче, а не тратить время на уяснение различий в использовании тех или иных элементов управления, команд и т.д. Обеспечивая **преемственность** полученных ранее знаний и навыков, согласованность делает интерфейс **узнаваемым и предсказуемым**.

Согласованность важна для всех аспектов интерфейса, включая имена команд, визуальное представление информации и поведение интерактивных элементов. Для реализации свойства согласованности в создаваемом программном обеспечении, необходимо учитывать его различные аспекты.

Согласованность в пределах продукта

Одна и та же команда должна выполнять одни и те же функции, где бы она ни встретилась, причем одним и тем же образом. Например, если в одном диалоговом окне команда *Копировать* означает немедленное выполнение соответствующих действий, то в другом окне она не должна требовать от пользователя дополнительно указать расположение копируемой информации. Надо использовать одну и ту же команду, чтобы выполнить функции, которые кажутся подобными пользователю.

Согласованность в пределах рабочей среды

Поддерживая согласованность с интерфейсом, предоставляемым операционной системой, приложение может «опираться» на те знания и навыки пользователя, которые он получил ранее при работе с другими приложениями.

Согласованность в использовании метафор

Если поведение некоторого программного объекта выходит за рамки того, что обычно подразумевается под соответствующей ему метафорой, у пользователя могут возникнуть трудности при работе с таким объектом. Например, если для программного объекта *Корзина* определить операцию *Запуск*, то для уяснения ее смысла пользователю, скорее всего, потребуется посторонняя помощь.

ДРУЖЕСТВЕННОСТЬ ИНТЕРФЕЙСА (ПРИНЦИП «ПРОЩЕНИЯ» ПОЛЬЗОВАТЕЛЯ)

Пользователи обычно изучают особенности работы с новым программным продуктом методом проб и ошибок. Эффективный интерфейс должен принимать во внимание такой подход. На каждом этапе работы он должен разрешать только соот-

ветствующий набор действий и **предупреждать** пользователей о тех ситуациях, где они могут повредить системе или данным; еще лучше, если у пользователя существует возможность **отменить** или **исправить** выполненные действия.

Даже при наличии хорошо спроектированного интерфейса пользователи могут делать те или иные ошибки. Эти ошибки могут быть как «физического» типа (случайный выбор неправильной команды или данных) так и «логического» (принятие неправильного решения на выбор команды или данных). **Эффективный интерфейс должен позволять предотвращать ситуации, которые, вероятно закончатся ошибками.** Он также должен уметь **адаптироваться** к потенциальным ошибкам пользователя и облегчать ему процесс устранения последствий таких ошибок.

ПРИНЦИП «ОБРАТНОЙ СВЯЗИ»

Всегда обеспечивайте обратную связь для действий пользователя. Каждое действие пользователя должно получать **визуальное**, а иногда и **звуковое подтверждение** того, что программное обеспечение восприняло введенную команду; при этом вид реакции, по возможности, должен учитывать природу выполненного действия.

Обратная связь эффективна в том случае, если она реализуется своевременно, т.е. как можно ближе к точке последнего взаимодействия пользователя с системой. Когда компьютер обрабатывает поступившее задание, полезно предоставить пользователю информацию относительно состояния процесса, а также возможность прервать этот процесс в случае необходимости. Ничто так не смущает не очень опытного пользователя, как заблокированный экран, который никак не реагирует на его действия. Типичный пользователь способен вытерпеть только несколько секунд ожидания ответной реакции от своего электронного «собеседника».

ПРОСТОТА ИНТЕРФЕЙСА

Интерфейс должен быть простым. При этом имеется в виду не упрощенчество, а обеспечение **легкости** в его **изучении** и в **использовании**. Кроме того, он должен предоставлять **доступ** ко всему перечню функциональных возможностей, предусмотренных данным приложением. Реализация доступа к широким функциональным возможностям и обеспечение простоты работы противоречат друг другу. Разработка эффективного интерфейса призвана **сбалансировать** эти цели.

Один из возможных путей поддержания простоты — представление на экране информации, минимально необходимой для выполнения пользователем очередного шага задания. Многословные командные имена или сообщения, непродуманные или избыточные фразы затрудняют пользователю извлечение существенной информации.

Другой путь к созданию простого, но эффективного интерфейса — **размещение** и представление элементов на экране с учетом их **смыслового значения** и логической взаимосвязи. Это позволяет использовать в процессе работы ассоциативное мышление пользователя.

Можно помочь пользователям управлять сложностью отображаемой информации, используя **последовательное раскрытие** (диалоговых окон, разделов меню и т.д.). Последовательное раскрытие предполагает такую организацию информации, при которой в каждый момент времени на экране находится только та ее часть, которая необходима для выполнения очередного шага. Сокращая объем информации, представленной пользователю, уменьшают объем информации, подлежащей обработке. Примером такой организации является иерархическое (каскадное) меню, каждый уровень которого отображает только те пункты, которые соответствуют одному, выбранному пользователем, пункту более высокого уровня.

ГИБКОСТЬ ИНТЕРФЕЙСА

Гибкость интерфейса — это его способность учитывать уровень подготовки и производительность труда пользователя. Гибкость предполагает возможность изменения структуры диалога и/или входных данных. Концепция гибкого (*адаптивного*) интерфейса в настоящее время является одной из основных областей исследования взаимодействия человека и ЭВМ. Основная проблема состоит не в том, как организовать изменения в диалоге, а в том, какие признаки нужно использовать для определения необходимости внесения изменений и их сути.

ЭСТЕТИЧЕСКАЯ ПРИВЛЕКАТЕЛЬНОСТЬ

Проектирование визуальных компонентов является важнейшей составной частью разработки программного интерфейса. Корректное визуальное представление используемых объектов обеспечивает передачу весьма важной дополнительной информации о поведении и взаимодействии различных объектов. В то же время следует помнить, что каждый визуальный элемент, который появляется на экране, потенциально требует внимания пользователя, которое, как известно, не безгранично. Следует формировать на экране среду, которая не только содействовала бы **пониманию** пользователем представленной информации, но и позволяла бы **сосредоточиться** на наиболее важных ее аспектах.

Наибольших успехов в проектировании пользовательского интерфейса, обладающего перечисленными свойствами, к настоящему времени добились разработчики компьютерных игр.

Качество интерфейса сложно оценить количественными характеристиками, однако более или менее объективную его оценку можно получить на основе приведенных ниже частных показателей.

1. **Время**, необходимое определенному пользователю для достижения заданного уровня знаний и навыков по работе с приложением (например, непрофессиональный пользователь должен освоить команды работы с файлами не более чем за 4 часа).

2. **Сохранение** полученных рабочих **навыков** по истечении некоторого времени (например, после недельного перерыва пользователь должен выполнить определенную последовательность операций за заданное время).

3. **Скорость решения задачи** с помощью данного приложения; при этом должно оцениваться не быстроедействие системы и не скорость ввода данных с клавиатуры, а время, необходимое для достижения цели решаемой задачи. Исходя из этого, критерий оценки по данному показателю может быть сформулирован, например, так: пользователь должен обработать за час не менее 20 документов с ошибкой не более 1%.

4. Субъективная **удовлетворенность** пользователя при работе с системой (которая количественно может быть выражена в процентах или оценкой по n-бальной шкале).

Обобщая изложенное выше, можно кратко сформулировать те основные правила, соблюдение которых позволяет рассчитывать на создание эффективного пользовательского интерфейса.

- Интерфейс пользователя необходимо проектировать и разрабатывать как отдельный компонент создаваемого приложения.
- Необходимо учитывать возможности и особенности аппаратно-программных средств, на базе которых реализуется интерфейс.
- Целесообразно учитывать особенности и традиции той предметной области, к которой относится создаваемое приложение.

- Процесс разработки интерфейса должен носить итерационный характер, его обязательным элементом должно быть согласование полученных результатов с потенциальным пользователем.
- Средства и методы реализации интерфейса должны обеспечивать возможность его адаптации к потребностям и характеристикам пользователя.

1.3. СТАНДАРТИЗАЦИЯ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

Ведущие специалисты в области человеко-машинных компьютерных систем уже к середине 70-х годов осознали необходимость формирования единых подходов к реализации пользовательского интерфейса. Однако в силу ограниченных технических возможностей вычислительных систем многие из рассмотренных выше принципов воспринимались программистами-практиками как некие абстрактные пожелания.

Длительное время основной формой общения пользователя с компьютером оставался диалог в форме «вопрос-ответ». Но, возможно, именно потому, что компьютер выступал в роли собеседника, очень быстро возникла необходимость исследования психологических аспектов общения человека с компьютером. Результаты этих исследований заставили вспомнить об эргономике рабочего места. В настоящее время уже ни одна серьезная публикация, посвященная пользовательскому интерфейсу, не обходится без ссылок на результаты, полученные в таких областях знаний, как психология, эргономика, математическая лингвистика, кибернетика и т.д.

В качестве иллюстрации того, насколько серьезно относятся «законодатели моды» в области компьютерных технологий к проблемам интерфейса, можно отметить следующий факт. Американский Национальный институт стандартов (ANSI) имеет по данному направлению специальную консультативную группу — Комитет по стандартам интерфейса «человек-компьютер» (The Human-Computer Interface Standard Committee). Существуют подобные организации не только в США, но и в других странах; более того, имеются также международные исследовательские группы, работающие в этом направлении, например, Международный консультативный комитет по телеграфии и телефонии (International Telegraph and Telephone Consultation Committee), изучающий особенности интерактивных элементов интерфейса.

Многими из этих организаций или рабочих групп, в свое время, были подготовлены проекты документов по стандартизации пользовательских интерфейсов, содержащих принципы их проектирования и реализации. Так, в 1986 году было опубликовано «Руководство по разработке программного пользовательского интерфейса», содержащее 944 принципа, касающихся ввода и отображения данных, поддержки пользователя, защиты данных и т.д. Однако ни один из этих проектов не получил статуса официального документа, поскольку все они имели общий недостаток (тот же, что и первые исследования в этой области): в них не учитывались технологические возможности инструментальных средств, имевшихся в распоряжении разработчиков программного обеспечения.

Ситуация коренным образом изменилась в 1987 г., когда корпорация IBM объявила о намерении создать единую среду разработки приложений (Systems Application Architecture — SAA).

Данный проект предусматривает не только разработку единых принципов создания приложений, но и «материализацию» этих принципов на основе соответствующей технологической базы.

Целями проекта являются:

1. Повышение производительности труда программистов и конечных пользователей.
2. Облегчение эксплуатации и сопровождения программного обеспечения.

3. Повышение эффективности распределенной обработки информации.
4. Увеличение отдачи инвестиций в разработку информационных систем.

Проект SAA содержит 4 компонента:

- Соглашения по интерфейсу пользователя (Common User Access — **CUA**);
- Соглашения по программному интерфейсу (Common Programming Interface — **CPI**);
- Соглашения по разработке приложений (Common Applications — **CA**);
- Соглашения по коммуникациям (Common Communications Support — **CCS**).

В качестве технологической базы для реализации соглашений по пользовательскому интерфейсу было предложено конкретное инструментальное средство — Programming Toolkit для операционной системы OS/2. При его создании был учтен накопленный к тому времени опыт разработки интерфейсов, а также последние достижения в данной области, в первую очередь — появление графических интерфейсов.

Исследованиями и практической реализацией графических интерфейсов в то время уже занимались такие фирмы как Xerox, Apple, Digital Research и Microsoft. В результате их деятельности были определены основные концепции построения графических пользовательских интерфейсов:

- использование единой рабочей среды пользователя в виде так называемого Рабочего стола;
- объектно-ориентированный подход к описанию заданий пользователей;
- использование графических окон в качестве основной формы отображения данных;
- применение средств неклавиатурного ввода, основанного на выборе и указании с помощью манипулятора «мышь».

В силу различных причин фирма IBM при реализации проекта SAA наиболее тесно сотрудничала с фирмой Microsoft, в результате чего была создана графическая оболочка Microsoft Windows IBM Top View. PI хотя впоследствии пути двух гигантов компьютерного бизнеса несколько разошлись, основные положения проекта SAA живы и успешно развиваются: корпорацией IBM — применительно к OS/2, а фирмой Microsoft — в рамках семейства OC Windows.

В марте 1997 года фирма Microsoft выпустила пакет Visual Studio 97, в который вошли все созданные ею инструментальные средства разработки приложений, а также средства автоматизации сопровождения программных продуктов (Visual SourceSafe). Это событие можно рассматривать как очередной шаг в направлении практической реализации идей проекта SAA.

И хотя требования и спецификации, изложенные в CUA, пока так и не стали международным стандартом де-юре, ориентация огромного числа производителей ПО на интерфейс MS Windows позволяет считать их таковыми де-факто.

Справедливости ради следует отметить, что для UNIX-систем, в определенном смысле являющихся конкурентом Windows, существует аналогичный «почти стандарт», представленный архитектурой XWindow.

Итак, стремление к стандартизации пользовательского интерфейса налицо, и оно обусловлено не только коммерческими интересами ведущих производителей ПО. Вместе с тем, единого официально утвержденного стандарта пока нет, но уже сейчас хотелось бы знать, по каким параметрам может оцениваться «степень стандартизации» пользовательского интерфейса.

На наш взгляд, стандартизованный интерфейс (именно стандартизованный, а не стандартный) должен отвечать двум основным требованиям:

- обладать перечисленными в предыдущем разделе свойствами (естественности, согласованности и т.д.);
- быть узнаваемым (или предсказуемым, что в данном случае одно и то же).

Второе требование, в свою очередь, предполагает, что интерфейс содержит только стандартные базовые элементы; каждый такой элемент должен иметь «узаконенное» название и определенный перечень свойств. Например, нельзя называть меню «списком» и при этом использовать его для вывода результатов расчетов.

На первый взгляд может показаться, что стандартизация интерфейса ведет к убогому однообразию внешнего облика программных продуктов. Но ведь и Моцарт, и автор «Мурки» пользовались одними и теми же семью нотами... А программисты, знакомые с алгоритмизацией, знают, что любой, сколь угодно сложный алгоритм содержит всего три-четыре базовые алгоритмические конструкции. Так что и при создании стандартизованного интерфейса результат будет зависеть в первую очередь от «композитора» — разработчика.

Несмотря на широкое распространение графического интерфейса, он не является единственно возможным или необходимым вариантом организации взаимодействия пользователя с приложением. Поэтому и в проектах документов по стандартизации интерфейса рассматривается целый ряд вопросов, относящихся к общей методике проектирования и реализации пользовательского интерфейса.

2. ПРОЕКТИРОВАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

2.1. ЖИЗНЕННЫЙ ЦИКЛ ПРОГРАММНОГО ПРОДУКТА

По истечении почти двух десятилетий активного использования компьютерных технологий Министерство обороны США пришло к выводу, что основной причиной неудач крупных информационных проектов является методология их реализации (точнее, ее отсутствие). От неприятных сюрпризов не могли спасти ни опытные менеджеры, ни исчерпывающее тестирование, ни применение CASE-средств. Каждая новая обнаруженная ошибка заставляла почти полностью пересматривать проект. И даже, если в конце концов удавалось получить приемлемый результат, при поступлении следующего заказа картина повторялась. Осознание указанного обстоятельства заставило Министерство обороны США сделать шаг в направлении стандартизации процессов разработки программных систем. Первый стандарт был утвержден в 1985 году, а в 1994-1996 годах был разработан и принят новый, значительно более детальный и глубокий стандарт — **MIL-STD-498**. Он представляет собой комплект из трех документов общим объемом около 600 страниц и устанавливает терминологию, процессы, задачи и объекты, используемые при разработке и сопровождении проектов программных систем. Основные положения этого военного стандарта согласованы с международным стандартом **ISO/IEC 12207:1995 («Процессы жизненного цикла программных средств»)**, но вместе с тем являются более конкретными и приближенными к практике.

В нашей стране попытка стандартизации процессов создания программных систем вылилась в создание комплекта документов под общим названием **«Единая система программной документации»**, который увидел свет еще в 1977 году (да, были времена, когда мы оказывались впереди планеты всей не только в области балета) и периодически корректировался вплоть до последнего времени.

При весьма заметном различии в деталях все известные стандарты используют в качестве исходного положения понятие **жизненного цикла** программного продукта (или системы).

Под жизненным циклом понимается последовательность процессов, действий и задач, которые осуществляются в ходе разработки, эксплуатации (использования) и сопровождения программного продукта в течение всей его жизни, от определения требований до завершения использования.

Практически все стандарты (даже военные) предусматривают возможность их адаптации к особенностям конкретного проекта при условии соблюдения основных требований к технологии и показателям качества продукта. Например, на этапе формирования требований к системе должны учитываться:

- область применения системы;
- требования пользователя (заказчика) к функциональным возможностям системы, к уровню ее безопасности и защищенности;
- эргономические требования и требования к уровню квалификации пользователей;
- степень документированности системы;
- организация сопровождения и т.д.

Необходимо подчеркнуть, что положения стандартов остаются справедливыми вне зависимости от предназначения, уровня сложности и состава коллектива разработчиков программного продукта, то есть даже в тех случаях, когда речь идет о небольшой программной утилите, а коллектив разработчиков состоит из одного человека.

ПРОЕКТИРОВАНИЕ

Начальный этап в разработке программного продукта (приложения) является наиболее критичным, поскольку на этой фазе определяется общая концепция создаваемого продукта. Если проект в своей основе неудовлетворителен, впоследствии трудно будет что-либо кардинально изменить в лучшую сторону.

Эта часть процесса разработки включает не только определение цели и характеристик приложения, но и понимание того, кто является его потенциальными пользователями — их задачи, намерения, цели. Это предполагает учет таких показателей, как например, возраст пользователей, их пол, экспертные знания, уровень опыта, физические ограничения, специальные потребности и т.д. Продумайте структуру приложения и метафоры, которые могут быть применены при ее реализации. Решению указанной проблемы способствует наблюдение за работой пользователей при выполнении ими задач в данной предметной области.

Представьте проект разработки в письменной форме; это не только обеспечивает важную контрольную точку в процессе создания приложения и средство взаимодействия с пользователями, но часто помогает сделать проект более конкретным и выявить нерешенные вопросы.

ПРОТОТИПИРОВАНИЕ

После того, как определены основные концепции проекта, разрабатывается прототип создаваемого приложения, отражающий некоторые основные аспекты его функционирования. Это может быть сделано с помощью ручки и бумаги; в зависимости от уровня вашей подготовки и сложности приложения его прототип может быть представлен либо в виде иллюстраций интерфейса (внешне они могут напоминать картинки из комиксов), либо в виде специальных схем (в частности, в виде сетей Петри). На последующей стадии может быть создана модель (или макет)

проектируемого приложения — действующее программное обеспечение, использующее либо специальные средства макетирования (например, какую-либо CASE-систему), либо обычные инструментальные средства программирования.

Прототип играет важную роль по многим причинам. Во-первых, он предоставляет хорошую возможность для обсуждения создаваемого приложения как внутри группы разработчиков, так и с потенциальными пользователями. Во-вторых, он может помочь вам определить характер потока заданий и лучше представить себе то, чем вы, собственно, занимаетесь. Это особенно полезно в начале процесса разработки.

Форма представления прототипа зависит от цели разработки. Действующие прототипы обычно наиболее полно позволяют оценить качество механизма взаимодействия пользователя с разрабатываемым приложением, т.е. качество интерфейса.

ИСПЫТАНИЕ ПРОГРАММНОГО ПРОДУКТА

UCD-технология предполагает достаточно активное привлечение пользователя к процессу разработки. Совместное с потенциальным пользователем испытание создаваемого приложения обеспечивает получение весьма ценной дополнительной информации и является, как правило, залогом последующей успешной реализации продукта. Необходимо подчеркнуть, что испытание программного продукта принципиально отличается от его отладки.

Первое и наиболее важное отличие обусловлено различными целями этих двух процессов: отладка имеет целью выявление дефектов (ошибок) программирования, в то время как в ходе испытаний вы оцениваете, насколько полно разработанное приложения (в частности, его интерфейс) отвечает потребностям и ожиданиям пользователя. Конечно, имеющиеся программные ошибки могут иногда повлиять на качество интерфейса, но это скорее исключение, чем правило.

Второе принципиальное отличие состоит в том, что отладку выполняет непосредственно его разработчик, а основным действующим лицом при проведении испытаний является потенциальный пользователь (заказчик). Благодаря этому в ходе испытаний могут быть выявлены не только технические погрешности, но и концептуальные проблемы в предлагаемом продукте.

Кроме того, испытания могут проводиться для двух или более альтернативных вариантов реализации создаваемого приложения с целью выявления наиболее удачного именно с точки зрения пользователя и решаемых им задач.

В качестве результатов испытаний большое значение имеют не только количественные (объективные) данные о работе приложения в тех или иных ситуациях, но и «качественная» информация, отражающая субъективное восприятие пользователем предлагаемого варианта приложения, его удовлетворенность, а также перечень проблем, которые на его взгляд могут иметь место при реальной эксплуатации программного продукта.

ПОВТОРНОЕ ВЫПОЛНЕНИЕ ЭТАПОВ РАЗРАБОТКИ

Поскольку испытания часто обнаруживают те или иные слабости проекта, или, по крайней мере, обеспечивают получение дополнительной информации, которую вы захотите использовать, почти всегда оказывается необходимым возврат к одному из предыдущих этапов разработки (а иногда и в начальную точку) и проведение повторных испытаний. Так может продолжаться до тех пор, пока и разработчик, и потенциальные пользователи не будут полностью удовлетворены полученными результатами.

ОЦЕНКА ПОТРЕБИТЕЛЬСКИХ СВОЙСТВ ПРИЛОЖЕНИЯ В ПРОЦЕССЕ РАЗРАБОТКИ

Как было указано выше, основная цель испытаний — определить, насколько полно разработанное приложение (в первую очередь, его интерфейс) отвечает потребностям и ожиданиям пользователя. В связи с этим основным направлением испытаний приложения является оценка его «потребительских свойств» (Usability). Такая оценка должна проводиться, начиная с самых ранних этапов разработки. Основой для проведения оценки должны служить данные о том, как пользователи обычно выполняют ту работу, которую призвано автоматизировать создаваемое приложение. По мере того, как разработка продвигается, оценка потребительских свойств приложения должна постоянно уточняться. Чем чаще и корректнее будет проводиться оценка, тем выше будет качество разработки.

ТЕХНИКА ПРОВЕДЕНИЯ ИСПЫТАНИЙ ПОТРЕБИТЕЛЬСКИХ СВОЙСТВ ПРИЛОЖЕНИЯ

Проведение испытаний потребительских свойств приложения требует привлечения значительных дополнительных сил и средств, в том числе специалистов тестовых лабораторий, имеющих в своем распоряжении соответствующее оборудование для регистрации результатов испытаний. Тем не менее, даже обычные «офисные инструменты» (магнитофон, секундомер и записная книжка) могут принести существенную пользу. Используемые тесты не должны быть «всеохватывающими». Значительно более полезны быстрые итеративные тесты, ориентированные на исследование конкретных проблем; практика показывает, что при таком подходе 6-10 участников испытаний могут идентифицировать 80 — 90 процентов основных проблем разработки.

Позвольте пользователям, участвующим в испытании, в течение разумного времени попытаться самостоятельно справиться с возникшими затруднениями. В тех случаях, когда они все-таки окажутся в тупиковой ситуации, которая потребует вмешательства разработчика, не торопитесь давать конкретный совет. Дайте сначала общие рекомендации по решению возникших проблем. Для более трудных ситуаций, вероятно, целесообразно приостановить тест и внести соответствующие поправки в работу пользователей.

Попросите участников испытаний, чтобы они вслух комментировали свою работу, возникающие проблемы и высказывали свои предложения по их устранению. В некоторых случаях по окончании испытаний полезно провести анкетирование участников, с тем чтобы они оценили продукт или задания, которые они выполняли.

Запишите результаты теста, используя портативный магнитофон, или, еще лучше, видеокамеру. Поскольку даже наилучший наблюдатель способен пропустить детали, последующий просмотр результатов может оказать неоценимую услугу. Кроме того, достаточно рискованно принимать решения, базирующиеся на выводах одного человека. Запись данных позволяет просматривать и оценивать результаты всей группе разработчиков.

АЛЬТЕРНАТИВНЫЙ ПОДХОД К ПРОВЕДЕНИЮ ИСПЫТАНИЙ ПРИЛОЖЕНИЯ

Наряду с рассмотренным выше подходом, ряд проблем, возникающих в процессе создания программного продукта, может быть решен на основе методики «коллективной генерации идей». Как правило, для ее реализации формируется специальная группа из числа разработчиков (focus group — группа концентрации). Она бывает особенно полезна для генерации начальных идей или для предварительной оценки идей, возникающих в процессе разработки. Для работы такой группы требуется председатель, который направлял бы дискуссию об аспектах выполнения того или иного шага разработки, позволяя вместе с тем участникам свободно выражать их мнения.

Можно также использовать технику «сквозного контроля» (walkthroughs), при которой вы «проводите» пользователя по определенному, заранее продуманному сценарию работы с приложением и запрашиваете его впечатления по мере продвижения к конечной цели.

На разработку продукта оказывают влияние многие дополнительные факторы. Например, маркетинговые соображения могут потребовать сокращения сроков разработки, или сравнительные оценки с аналогичными продуктами могут заставить реализовать в создаваемом ПО дополнительные характеристики. При этом следует помнить, что сокращение сроков и внесение дополнительных функциональных возможностей могут повлиять на качество продукта. Нет простого уравнения, позволяющего определить, каким образом вносимые изменения повлияют на первоначальный проект приложения. Однако важную роль в такой оценке могут сыграть следующие соображения:

- Каждая дополнительная характеристика потенциально влияет на поведение, сложность, устойчивость, эксплуатацию и издержки по сопровождению создаваемого ПО.
- После выхода в свет официальной версии продукта труднее устранить проблемы, оставшиеся нерешенными на стадии разработки, поскольку пользователи могут приспособиться, или даже «подчиниться» имеющимся недостаткам вашего ПО.
- Простота пользовательского интерфейса — это далеко не то же самое, что его упрощенчество. Чтобы сделать нечто простым в использовании, часто требуется приложить много сил и создать весьма сложное изделие с точки зрения его внутренней организации.
- Доработки программного кода с целью расширения функциональных возможностей ПО совсем не обязательно будут иметь пропорциональный положительный эффект сточки зрения интерфейса пользователя. Например, если основная задача пользователя состоит в выборе единственного объекта, то предоставляя ему возможность одновременного выбора нескольких объектов, вы только усложняете ему работу.

2.2. ЭТАПЫ ПРОЕКТИРОВАНИЯ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

Проводившиеся в свое время исследования психологических аспектов общения человека с компьютером показали, что следует всячески стремиться к дегуманизации этого общения, то есть пользователь не должен воспринимать компьютер как полноценного собеседника. Тем не менее обмен информацией между пользователем и компьютером (точнее, его программным обеспечением) по всем формальным признакам соответствует понятию «диалог» в общепринятом смысле. Вероятно, читатель даже без помощи Толкового словаря сможет перечислить основные правила, которые должны соблюдаться, чтобы диалог оказался конструктивным:

во-первых, участники диалога должны понимать язык друг друга;

во-вторых, они не должны говорить одновременно;

в-третьих, очередное высказывание должно учитывать как общий контекст диалога, так и последнюю информацию, полученную от собеседника.

Если собеседники обсуждают вопросы, относящиеся к какой-либо специальной области, они должны придерживаться единой терминологии; если же один из них пытается что-то объяснить другому, ему следует сначала пояснить основные термины и понятия.

К этому следует также добавить, что применение дополнительных выразительных средств способствует лучшему взаимопониманию. Иногда одна удачная иллюстрация заменяет десятки слов. Например, на вопрос "Как пройти в библиотеку?" лучше всего отвечать, имея под рукой карту города.

Теперь давайте вспомним, чего не любят наши собеседники.

Прежде всего — излишней фамильярности. Кроме того, мало кому нравится, когда во время разговора его дергают за рукав, откручивают пуговицу или используют еще какие-то оригинальные способы привлечения внимания. Очень краткие ответы и слишком большие паузы могут сбить собеседника с толку, а злоупотребление специальными терминами или жаргонизмами вообще может привести к преждевременному завершению беседы.

Приведенные выше рассуждения носят весьма общий характер и применимы практически к любому диалогу, независимо от того, в каких отношениях состоят собеседники и с какой целью ведется диалог. Однако эти факторы существенно влияют на *структуру* диалога, то есть на форму общения. Например, если встретились два приятеля, то диалог напоминает игру в теннис: инициатива попеременно переходит от одного собеседника к другому; если вы пришли в ресторан, то ваше общение с официантом ограничивается выбором блюд из предложенного меню, а при оформлении загранпаспорта чиновник предложит заполнить одну-две анкеты, а затем просмотрит их, при необходимости уточняя те или иные моменты.

Таким образом, при проектировании пользовательского интерфейса необходимо определить:

- структуру диалога;
- возможный сценарий развития диалога;
- содержание управляющих сообщений и данных, которыми могут обмениваться человек и приложение (семантику сообщений);
- визуальные атрибуты отображаемой информации (синтаксис сообщений).

2.2.1. ВЫБОР СТРУКТУРЫ ДИАЛОГА

Выбор структуры диалога — это первый из этапов, который должен быть выполнен при разработке интерфейса.

Рассмотренные ниже четыре варианта структуры диалога являются разновидностями структуры типа «вопрос — ответ», тем не менее каждая из них имеет свои особенности и наиболее удобна для определенного класса задач.

ДИАЛОГ ТИПА «ВОПРОС - ОТВЕТ»

Структура диалога типа «вопрос-ответ» (Q&A) основана на аналогии с обычным интервью. Система берет на себя роль интервьюера и получает информацию от пользователя в виде ответов на вопросы. Это наиболее известная структура диалога; все диалоги, управляемые компьютером, в той или иной степени состоят из вопросов, на которые пользователь отвечает. Однако в структуре Q&A этот процесс выражен

явно. В каждой точке диалога система выводит в качестве подсказки один вопрос, на который пользователь дает один ответ. В зависимости от полученного ответа система может решить, какой следующий вопрос задавать. Структура Q&A предоставляет естественный механизм ввода как управляющих сообщений (команд), так и данных. Никаких ограничений на диапазон или тип входных данных, которые могут обрабатываться, не накладывается. Существуют системы, ответы в которых даются на естественном языке, но чаще используются предложения из одного слова с ограниченной грамматикой.

Диалог в виде вопросов и ответов в достаточной степени обеспечивает поддержку пользователя, так как даже краткий наводящий вопрос при разумном построении может быть самопоясняющим. Структура Q&A не гарантирует минимального объема ввода, оцениваемого по количеству нажатий клавиш, однако при подходящем подборе сокращений можно уменьшить любую избыточность. Вместе с тем, структура Q&A обладает одним существенным недостатком. Даже если ввод происходит достаточно быстро, для человека, который уже знает, какие вопросы задает система и какие ответы нужно на них давать, отвечать на всю серию вопросов довольно утомительно.

С появлением графического интерфейса структура Q&A несколько устарела, тем не менее у нее имеются определенные достоинства. Эта структура может удовлетворить требования различных пользователей и типов данных. В частности, такая структура особенно уместна при реализации диалога с множеством «ответвлений», т.е. в тех случаях, когда на каждый вопрос предусматривается большое число ответов, каждый из которых влияет на то, какой вопрос будет задан следующим. По этой причине структура Q&A часто используется в экспертных системах.

ДИАЛОГ НА ОСНОВЕ МЕНЮ

Меню является, пожалуй, наиболее популярным вариантом организации запросов на ввод данных во время диалога, управляемого компьютером.

Существует несколько основных форматов представления меню на экране:

- список объектов, выбираемых прямым указанием, либо указанием номера (или мнемонического кода);
- меню в виде блока данных;
- меню в виде строки данных;
- меню в виде пиктограмм.

Меню в виде строки данных может появляться вверху или внизу экрана и часто остается в этой позиции на протяжении всего диалога. Таким образом, посредством меню удобно отображать возможные варианты данных для ввода, доступных в любое время работы с системой. Если в системе имеется достаточно большое разнообразие вариантов действий, организуется иерархическая структура из соответствующих меню. Дополнительные меню в виде блоков данных «всплывают» на экране в позиции, определяемой текущим положением указателя, либо «выпадают» непосредственно из строки меню верхнего уровня. Эти меню исчезают после выбора варианта.

Меню в виде пиктограмм представляет собой множество объектов выбора, разбросанных по всему экрану; часто объекты выбора содержат графическое представление вариантов работы.

Пользователь диалогового меню может выбрать нужный пункт, вводя текстовую строку, которая идентифицирует этот пункт, указывая на него непосредственно или

просматривая список и выбирая из него. Система может выводить пункты меню последовательно, при этом пользователь выбирает нужный ему пункт нажатием клавиши.

Меню можно с равным успехом применять для ввода как управляющих сообщений, так и данных. Приемлемая структура меню зависит от его размера и организации, от способа выбора пунктов меню и реальной потребности пользователя в поддержке со стороны меню.

Структура типа меню является наиболее естественным механизмом для работы с устройствами указания и выбора: меню представляет собой изображение тех объектов, которые выбираются пользователем. Если диалог состоит исключительно из меню, можно реализовать последовательный интерфейс, при котором пользователь применяет только устройства для указания; однако такое постоянство редко достижимо на практике. Следует также заметить, что, хотя работа с этими устройствами не требует профессионального владения клавиатурой, для подготовленного пользователя это не самый быстрый способ выбора из меню. Вместо указания пользователь может сообщить о своем выборе вводом соответствующего идентификатора.

Меню — это наиболее удобная структура диалога для неподготовленных пользователей; жесткая очередность открытия и иерархическая вложенность меню может вызывать раздражение профессионала, замедлять его работу. Традиционная структура меню недостаточно гибка и не в полной мере согласуется с методами адаптации диалога, такими, например, как опережающий ввод, с помощью которого можно ускорить темп работы подготовленного пользователя. Более подробно вопросы организации и визуального представления меню рассмотрены в разделе «Проектирование элементов управления».

ДИАЛОГ НА ОСНОВЕ ЭКРАННЫХ ФОРМ

Как структура типа «вопрос — ответ», так и структура типа меню предполагают обработку на каждом шаге диалога единственного ответа. Диалог на основе экранных форм допускает обработку на одном шаге диалога нескольких ответов. На практике формы используются в основном там, где учет какой-либо деятельности требует ввода достаточно стандартного набора данных. Человек, заполняющий форму, может выбирать последовательность ответов, временно пропускать некоторый вопрос, возвращаться назад для коррекции предыдущего ответа и даже «порвать бланк» и начать заполнять новый. Он работает с формой до тех пор, пока не заполнит ее полностью и не передаст системе. Программная система может проверять каждый ответ непосредственно после ввода или выждать и вывести список ошибок только после заполнения формы целиком. В некоторых системах информация, вводимая пользователем, становится доступной только после нажатия клавиши «ввод» по окончании заполнения формы. Вопрос о том, надо ли проверять ответ непосредственно или отложить проверку до окончания ввода всех ответов, решить непросто: сообщения об ошибках, выводимые непосредственно после ответа, могут отвлечь внимание, но могут оказать и положительное влияние. Вообще в тех случаях, когда информация для ввода выбирается из некоторого целостного документа, проверку лучше отложить до конца заполнения формы, чтобы не прерывать процесс ввода; если же такой целостности нет, то проверку следует выполнять сразу после ввода ответа (после заполнения очередного поля).

Если встретилась какая-либо ошибка, приложение не должно заново выводить пустую форму; выводится форма с предыдущими ответами и допущенными

ошибками. Новый «бланк» выдается лишь в случае соответствующего запроса пользователя.

Таким образом, эту структуру уместно применять там, где источником данных служит существующая входная («бумажная») форма документа.

Не обязательно, чтобы внешний вид этих форм совпадал (это даже может ухудшить восприятие данных на экране), но все вводимые элементы данных должны располагаться в том же относительном порядке и иметь такой же формат, что и в исходном документе.

Часто все необходимые единицы ввода нельзя отобразить одновременно в пределах одного экрана (или окна), и их необходимо разделить на группы, которые отображаются на последовательности экранов (окон). Важно, чтобы это разбиение сохраняло логические связи и не приводило к разделению связанных частей документа.

Структура диалога на основе экранной формы обеспечивает высокий уровень поддержки пользователя: для каждого вопроса формы могут быть предусмотрены сообщения об ошибках и справочная информация. Пользователю можно также оказать помощь, включив некоторые элементы формата ответа в вопрос или в поле ответа. Эта структура позволяет повысить скорость ввода данных по сравнению со структурой типа «вопрос — ответ» и манипулировать более широким диапазоном входных данных, нежели меню; кроме того, с ней могут работать пользователи любой квалификации. Поскольку эта структура имеет последовательную, а не древовидную организацию, она в меньшей степени подходит для работы в режиме выбора вариантов. Еще одной областью применения экранных форм является задание параметров запросов в базах данных. Этот механизм иногда называют запросом по образцу (*Query by Example*).

Одним из типов заполнения форм являются также многовариантные меню. В таких меню пользователю предоставляется список вариантов и он не ограничен возможностью единственного выбора; можно указать несколько вариантов.

ДИАЛОГ НА ОСНОВЕ КОМАНДНОГО ЯЗЫКА

Структура диалога на основе командного языка столь же распространена, что и структура типа меню. Она очень часто используется в операционных системах и располагается на другом конце спектра структур диалога по отношению к структуре типа меню. Исторически это первая из реализованных структур диалога.

При такой организации диалога программная система не выводит ничего, кроме постоянной подсказки (приглашения на ввод команды), которая означает готовность системы к работе. Каждую команду вводят с новой строки и обычно заканчивают нажатием клавиши «ввод». Ответственность за правильность задаваемых команд ложится на пользователя. Система информирует о невозможности выполнения неверной команды, не поясняя, как правило, причин.

Подобно меню, диалог на базе команд удобен для ввода управляющих сообщений, однако он обеспечивает более широкие возможности выбора в любой точке диалога и не требует иерархической организации обслуживающих его программ.

Программная система может поддерживать достаточно большое количество команд, но на практике следует ограничивать их число, чтобы не перегружать память пользователя. Структура на базе командного языка не отличается хорошей поддержкой пользователя и пригодна в основном для подготовленных специалистов. Перед использованием такой системы необходимо пройти курс обучения и в дальнейшем изучать особенности работы по документации, а не на практике. Более того,

поскольку системе неизвестно, что намеревается делать пользователь, трудно предложить какую-либо реальную помощь в процессе работы, кроме выдачи справок достаточно общего характера.

Поскольку данная структура предполагает большой объем запоминаемого материала, имена команд следует выбирать так, чтобы они несли смысловую нагрузку и легко запоминались. Разработчик должен стремиться избегать излишней Функциональности, происходящей от желания создать собственную команду для каждой функции, выполняемой системой, то есть не стоит создавать множество разнообразных команд с часто перекрывающимися функциями. Такие «благие» намерения нередко приводят к появлению большого количества ключевых слов для обозначения команд и синтаксических правил, многие из которых редко используются и лишь осложняют работу.

Диалог должен управлять данными. В интерфейсах на основе языков команд это обычно достигается с помощью составных командных строк, где ключевое слово для обозначения команды (что делать) предшествует списку параметров (входным данным). Параметры в списке можно задавать в одной из двух форм — в позиционной или ключевой. В первом случае назначение параметра определяется по его месту в командной строке. В случае ключевых параметров каждое значение предваряется определенным идентификатором, который определяет его назначение.

Позиционные параметры уменьшают объем вводимой информации, но их существенным недостатком является то, что вводимые значения должны указываться в строго определенном порядке, нарушение которого плохо диагностируется системой и может повлечь серьезные последствия. Задание позиционных параметров осложняется, если их список достаточно велик. Этот недостаток стремятся компенсировать, разрешая опускать неизменяемые параметры, вводя два разделителя друг за другом.

Ключевые параметры уменьшают нагрузку на память пользователя в том отношении, что отпадает необходимость в запоминании порядка их следования; кроме того, можно опускать необязательные параметры. С другой стороны, в этом случае пользователю необходимо запомнить множество ключевых слов, а разработчику — подобрать для них «осмысленные» имена. Этот подход также требует большего времени работы системы, чтобы распознать ключевые слова, заданные в произвольном порядке.

Многие командные языки поддерживают **макросы**, которые расширяют функциональные возможности диалога без увеличения количества команд. Макрос содержит несколько отдельных командных строк. При обращении к нему в процессе диалога отдельные строки макроса выполняются одна за другой, как если бы они вводились с клавиатуры. Это существенно сокращает диалоговые сеансы, содержанию часто повторяющиеся последовательности команд, которые, собственно, и оформляются в виде макросов.

Структура на основе языка команд по своим возможностям самая быстрая и гибкая из всех структур диалога. Большинство пользовательских интерфейсов на базе «естественного» языка реализуется с помощью языков команд с очень большим набором ключевых слов. Подготовленный пользователь испытывает удовольствие от ощущения того, что он управляет системой, а не наоборот. Однако эта структура не обеспечивает пользователя поддержкой, поэтому даже подготовленные пользователи считают, что очень сложно использовать все заложенные в ней возможности. Большинство же пользователей хорошо знакомы только с весьма ограниченным набором средств, с которыми они работают регулярно.

Изложенное можно представить в форме так называемой «таблицы выбора» [2] (рис. 2.1).

Критерии	Выбор пользова- теля	Тип диалога			
		меню	воп рос — отве т	язык кома нд	заполнен ие экранны х форм
Цель: Запрос Вычисления Сложный выбор ввод данных ввод данных (большой объем)		 + + + 	 + + + + +	 + + + + +	 + + +
Тип пользователя: Программист Непрограммист Имеет опыт работы нет опыта работы		 + +	 + +	 + + *	 + + *
Время обучения: очень малое менее 1 дня более 1 дня		 + +	 + +	 ** +	 ** +
Результат оценки					

* — использование этого типа диалога данной категорией пользователей требует наличия системы помощи;

** — использование средств системы возможно только в ограниченном объеме.

Рис. 2.1. Вариант таблицы выбора

Наряду с указанными в таблице, в неё могут быть включены и другие критерии выбора (наличие инструментальных средств разработки интерфейса, характер пользователя, ограничения по имеющимся ресурсам и т.д.).

Выбор наиболее подходящей структуры диалога на основе таблицы выполняется следующим образом.

1. Закрыть графы «Тип диалога».
2. В графе «Выбор пользователя» пометить критерии, относящиеся к рассматриваемому применению.
3. Для каждого типа диалога подсчитать число случаев, когда помечены соответствующие пункты и в графе «выбор пользователя», и в графе «тип диалога».
4. Подсчитать число совпадений для каждого типа диалога.

Основная ценность таблицы состоит в том, что ее можно использовать как исходный вариант выбора типа диалога, либо как средство окончательной проверки соответствия выбранного типа диалога рассматриваемым критериям.

Если предполагается, что одни пункты более важны, чем другие, можно брать их с разными весовыми коэффициентами. Можно также указать, какие пункты должны рассматриваться как выполняемые безусловно; типы диалога, не соответствующие хотя бы одному из таких пунктов, должны немедленно отвергаться, сколько бы очков они ни набрали по остальным пунктам.

2.2.2. РАЗРАБОТКА СЦЕНАРИЯ ДИАЛОГА

Развитие диалога во времени можно рассматривать как последовательность переходов системы из одного состояния в другое. Очевидно, что ни одно из этих состояний не должно быть «тупиковым», т.е. пользователь должен иметь возможность перейти из любого текущего состояния диалога в требуемое (за один или несколько шагов). Для этого в ходе разработки интерфейса необходимо определить все возможные состояния диалога и пути перехода из одного состояния в другое. Другими словами, необходимо разработать *сценарий диалога*.

Целями разработки сценария диалога являются:

- выявление и устранение возможных тупиковых ситуаций в ходе развития диалога;
- выбор рациональных путей перехода из одного состояния диалога в другое (из текущего в требуемое);
- выявление неоднозначных ситуаций, требующих оказания дополнительной помощи пользователю.

Сложность разработки сценария определяется в основном двумя факторами:

функциональными возможностями создаваемого приложения (т.е. числом и сложностью реализуемых функций обработки информации) и степенью неопределенности возможных действий пользователя.

В свою очередь, степень неопределенности действий пользователя зависит от выбранной структуры диалога. Наибольшей детерминированностью обладает диалог на основе меню, наименьшей — диалог типа «вопрос-ответ», управляемый пользователем.

Из сказанного следует, что сценарий диалога можно упростить, снизив степень неопределенности действий пользователя. Возможными способами решения этой задачи являются:

использование смешанной структуры диалога (применение меню с целью «ограничения свободы» пользователя там, где это возможно);

применение входного контроля вводимой информации (команд и данных).

Дополнительные возможности по снижению неопределенности действий пользователя предоставляет объектно-ориентированный подход к разработке интерфейса, при котором для каждого объекта заранее устанавливается перечень свойств и допустимых операций. Наиболее эффективен такой подход при создании графического интерфейса; более подробно эти вопросы обсуждаются в разделе «Особенности графического интерфейса».

Сокращая число возможных состояний диалога, разработчик вместе с тем должен помнить о необходимости отражения в его сценарии работы средств поддержки пользователя, что, несомненно, делает сценарий более сложным.

Способ описания сценария диалога зависит от степени его сложности. Существующие методы описания сценариев можно разделить на две большие группы:

неформальные и формальные методы.

Главное достоинство формальных методов состоит в том, что они позволяют автоматизировать как проектирование диалога, так и его модификацию (адаптацию) в соответствии с характеристиками пользователя.

В настоящее время наиболее широко используются формальные методы описания сценариев на основе сетей Петри и их расширений, а также на основе систем представления знаний (фреймовые модели и продукционные системы).

Независимо от способа описания сценария его основной структурной единицей является *шаг диалога*, соответствующий одному акту взаимодействия пользователя с системой. Схематично шаг диалога можно представить так, как показано на рис. 2.2.

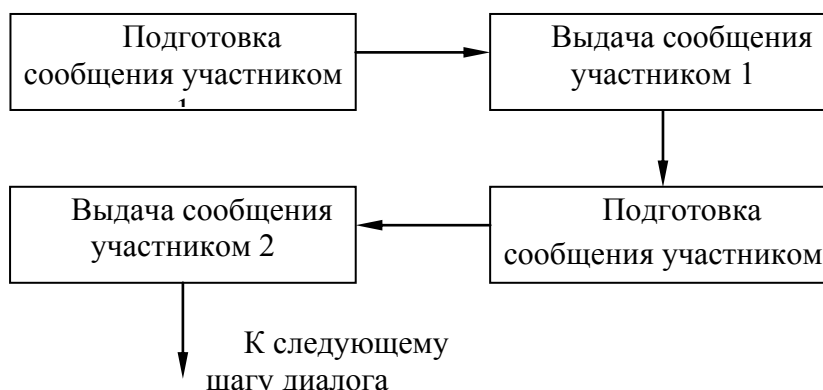


Рис. 2.2. Шаг диалога

Сценарий диалога позволяет описать процесс взаимодействия пользователя с приложением на уровне решаемой им прикладной задачи. Однако для программной реализации интерфейса такое описание носит слишком общий характер. Поэтому на этапе реализации необходимо перейти на уровень описания соответствующих процессов с помощью используемых инструментальных средств разработки приложения.

ТЕМП ВЕДЕНИЯ ДИАЛОГА

Процесс общения пользователя с компьютером связан с рядом существенных объективных и субъективных ограничений и должен соответствовать психофизиологическим возможностям человека: способности по приему и переработке информации, объемам сенсорной и кратковременной памяти, умению концентрировать внимание на наиболее важной информации, способности воспроизводить информацию из долговременной памяти и т.п.

В связи с этим при разработке сценария диалога должны учитываться такие психофизиологические особенности потенциальных пользователей, как моторные навыки, время реакции, восприимчивость цветовой гаммы и т.д. В данном разделе рассмотрим более подробно требования к одной из важнейших характеристик интерфейса — к обеспечиваемому темпу ведения диалога. Остальные из перечисленных выше требований будут рассмотрены в последующих разделах книги.

Темп ведения диалога зависит от характеристик аппаратных и программных средств ЭВМ, а также от специфики решаемых задач. Требование соответствия темпа ведения диалога психологическим особенностям человека выдвигает ограничения на значения этих характеристик не только «сверху», но и «снизу». Поясним это утверждение.

Время ответа (отклика) системы определяется как интервал между событием и реакцией системы на него. Данная характеристика интерфейса определяет задержку в работе пользователя при переходе к выполнению следующего шага задания.

Важность учета темпа ведения диалога была осознана еще в 60-х годах, когда появились первые интерактивные системы. Медленный ответ системы не соответствует психологическим потребностям пользователя, что приводит к снижению эффективности его деятельности. Слишком быстрый ответ также может создать неблагоприятное представление о системе.

Требования к времени ответа зависят от того, что ожидает пользователь от работы системы, и от того, как взаимодействие с системой влияет на выполнение его заданий. Исследования показали [3], что если время ответа меньше ожидаемого, точность выбора операции из меню увеличивается с увеличением времени ответа системы. Это связано с тем, что излишне быстрый ответ системы как бы «подгоняет» пользователя, заставляет его суетиться в стремлении «не отставать» от более расторопного партнера по общению. Замечено, что начинающий пользователь боится работать с системой, если, потратив несколько минут на ввод, он моментально получает от нее ответ с сообщением об ошибке.

Время ответа должно соответствовать естественному ритму работы пользователей. В обычном разговоре люди ожидают ответа около 2 секунд и ждут того же при работе с компьютером. Время ожидания зависит от их состояния и намерений. На представления пользователя оказывает сильное влияние также его предшествующий опыт работы с системой.

Обычно человек может одновременно запомнить сведения о пяти-девяти предметах. Считается также, что хранение данных в кратковременной памяти ограничено по времени: около 2 секунд для речевой информации и 30 секунд для сенсорной. Поэтому люди имеют склонность разбивать свою деятельность на этапы, соответствующие порциям информации, которые они могут хранить одновременно в памяти. Завершение очередного этапа называется *клаузой*. Задержки, препятствующие наступлению клаузы, очень вредны и неприятны, так как содержимое кратковременной памяти требует постоянного обновления и легко стирается под влиянием внешних факторов. Зато после клаузы подобные задержки вполне приемлемы и даже необходимы. Завершение задачи, ведущее к отдыху, называют *закрытием*. В этот момент исчезает необходимость дальнейшего хранения информации и человек получает существенное психологическое облегчение. Так как пользователи интуитивно стремятся к *закрытию* в своей работе, следует делить диалоги на фрагменты, чтобы пользователь мог «вовремя» забывать промежуточную информацию.

Пользователи, особенно новички, обычно предпочитают много мелких операций одной большой операции, так как в этом случае они могут не только лучше контролировать общее продвижение решения и обеспечить его удовлетворительный ход, но и отвлечься от деталей работы на предыдущих этапах.

Имеющиеся результаты исследований позволили выработать следующие рекомендации по допустимому времени ответа интерактивной системы:

0,1... 0,2 с — для подтверждения физических действий (нажатие клавиши, работа со световым пером, «мышью»);

0,5... 1,0 с — для ответа на простые команды (например, от момента ввода команды, выбора альтернативы из меню до появления нового изображения на экране);

1... 2 с — при ведении связного диалога (когда пользователь воспринимает серию взаимосвязанных вопросов как одну порцию информации для формирования одного или нескольких ответов, задержка между следующими друг за другом вопросами не должна превышать указанную длительность);

2... 4 с — для ответа на сложный запрос, состоящий в заполнении некоторой формы. Если задержка не влияет на другую работу пользователя, связанную с первой, могут быть приемлемы задержки до 10 с;

более 10 с — при работе в мультизадачном режиме, когда пользователь воспринимает данную задачу как фоновый процесс. Принято считать, что если пользователь не получает ответ в течение 20 с, то это не интерактивная система. В

таком случае пользователь может «забыть» о задании, заняться решением другой задачи и возвращаться к нему тогда, когда ему будет удобно. При этом программа должна сообщать пользователю, что задержка ответа не является следствием выхода системы из строя (например, путем регулярного обновления строки состояния системы или ведения протокола выполнения задания пользователя).

МЕТОДЫ РАЗРАБОТКИ ГИБКОГО ИНТЕРФЕЙСА

Предварительный анализ (хотя бы и на качественном уровне) возможного сценария диалога позволяет избежать многих проблем на этапе реализации приложения. Однако в случае если приложение может использоваться группой пользователей, имеющих различную степень подготовки, ряд вопросов остается нерешенным. Поэтому крайне желательно, чтобы в ходе диалога обеспечивалась достаточная гибкость. Она должна заключаться в способности приложения адаптироваться (пользователем или автоматически) к любому возможному уровню подготовки пользователя.

Существуют три вида адаптации: фиксированная, полная и косметическая.

При **фиксированной** адаптации пользователь явно выбирает уровень диалоговой поддержки. Простейший вариант такой адаптации основан на использовании правила двух уровней, согласно которому система обеспечивает два вида диалога:

- подробный (для начинающего пользователя);
- краткий (для подготовленного пользователя).

Правило двух уровней может быть расширено до правила N уровней диалога. Однако такой подход имеет несколько недостатков:

- 1) не учитывается тот факт, что навыки накапливаются постепенно;
- 2) пользователь может хорошо знать одну часть системы и совсем не знать другую;
- 3) пользователь сам определяет уровень своей подготовки, что снижает объективность оценки.

При **полной** адаптации диалоговая система стремится построить модель пользователя, которая по мере обучения последнего и определяет стиль диалога в зависимости от этих изменений. При этом одной из основных проблем является распознавание характеристик пользователя. Для ее решения необходимо определить, что использовать в качестве таких характеристик: время, затрачиваемое пользователем на ответ, количество его обращений за помощью или характер ошибок и тип запрашиваемой помощи.

В настоящее время полная (автоматическая) адаптация практически ни в одной диалоговой системе не реализована.

Косметическая адаптация призвана обеспечить гибкость диалога без учета поведения пользователя, но и без однозначного выбора им конкретного стиля диалога.

Такая адаптация может быть достигнута за счет применения следующих методов:

- использование умолчаний;
- использование сокращений;
- опережающий ввод ответов;
- многоуровневая помощь;
- многоязычность.

Использование умолчаний. Сущность умолчания состоит в том, что система использует некоторое изначально заданное значение какого-либо параметра, пока пользователь не изменит его. В этом случае имеют место два аспекта адаптации системы:

во-первых, начинающий пользователь имеет возможность использовать большинство параметров системы по умолчанию,

во-вторых, система может запоминать значения, либо заданные при последнем сеансе работы (например, имя редактируемого файла), либо наиболее часто используемые.

Для удобства начинающих пользователей значения, используемые по умолчанию, могут выводиться на экран вместе с соответствующим вопросом системы, например: «Дата регистрации документа? [текущая]».

Самый распространенный способ принятия значений по умолчанию — это нулевой ввод, т.е. простое нажатие клавиши «Ввод» в качестве ответа на вопрос системы. Если используется командный язык, то пользователь просто пропускает параметр, используемый по умолчанию.

Использование сокращений предполагает, что пользователь вместо полного имени команды может вводить ее любое допустимое сокращенное обозначение. На первый взгляд может показаться, что сокращенный ввод более удобен для начинающего пользователя. Но это не совсем так. Чтобы пользователь мог, не задумываясь, заменить команду корректным сокращением, он должен достаточно хорошо представлять имеющийся набор команд, усвоить «лексику» системы. Например, если в системе имеются команды *Copy* и *Compare*, то начинающему проще набрать полное имя, чем выбрать корректный вариант сокращения.

Одной из модификаций этого подхода является **опережающий ввод символов**, при котором система, «узнав» по первым символам команду, «дописывает» ее сама. Примером может служить интерфейс системы GPSS/PC, в которой при вводе начальных символов команды на экран выводится ее полное имя, а курсор автоматически перемещается в нужную позицию для ввода параметров этой команды. Разумеется, пользователь, особенно начинающий, испытывает чувство «глубокой признательности» такой системе за «посильную помощь».

Идея **опережающего ввода ответов** заключается в том, что пользователь имеет возможность на очередном шаге диалога вводить не один ответ, а цепочку последовательных ответов, упреждая возможные вопросы системы.

Один из методов обеспечения **многоуровневой помощи** состоит в том, что сначала на экран выводится сообщение начального уровня, а затем пользователь может уточнить полученную информацию, используя переход на более низкий уровень по ключевому слову. На таком принципе основана работа многих современных Help-систем, обучающих гипертекстовых систем.

Сущность **многоязычности** интерфейса состоит в том, что структура и семантика диалоговых сообщений, которые выдает и получает пользователь, должны отвечать нормам родного языка пользователя и не зависеть от того, на каком языке разработаны инструментальные средства, которые он использует.

Возможный подход к реализации многоязычности — создание средств реакции системы на действия пользователя (сообщения-запросы, подсказки, сообщения об ошибках) отдельно от синтаксиса языка программирования (инструментальных средств).

2.2.3. ВИЗУАЛЬНЫЕ АТРИБУТЫ ОТОБРАЖАЕМОЙ ИНФОРМАЦИИ

К визуальным атрибутам отображаемой информации относятся:

- взаимное расположение и размер отображаемых объектов;
- цветовая палитра;
- средства привлечения внимания пользователя. Проектирование размещения данных на экране предполагает выполнение следующих действий:

- 1) Определение состава информации, которая должна появляться на экране;

- 2) Выбор формата представления этой информации;
- 3) Определение взаимного расположения данных (или объектов) на экране;
- 4) Выбор средств привлечения внимания пользователя;
- 5) Разработка макета размещения данных на экране;
- 6) Оценка эффективности размещения информации.

Процесс проектирования повторяется до тех пор, пока разработчик и потенциальные пользователи не будут удовлетворены.

Общие принципы расположения информации на экране должны обеспечивать для пользователя:

возможность просмотра экрана в логической последовательности;
 простоту выбора нужной информации;
 возможность идентификации связанных групп информации;
 различимость исключительных ситуаций (сообщений об ошибках или предупреждений);

возможность определить, какое действие со стороны пользователя требуется (и требуется ли вообще) для продолжения выполнения задания.

Вопрос о том, какая информация подлежит отображению, решается в зависимости от специфики выполняемого пользователем задания. Здесь существенную роль играет правильное разбиение задания на операции (этапы), не требующие одновременного присутствия большого объема данных на экране. Это условие вытекает из такой психофизиологической особенности человека, как ограниченность его кратковременной памяти, способной хранить одновременно не более пяти — девяти объектов. Если вся информация исходного документа не помещается на одном экране, некоторые элементы данных могут повторяться на других экранах для сохранения целостности и последовательности обработки. Как правило, повторяемая информация не должна менять своего расположения на всех шагах выполнения задания.

Если в выделении логических групп есть сомнения, необходим тщательный учет пожеланий заказчика или предоставление ему возможности самостоятельного формирования таких групп.

Свойство естественности интерфейса предполагает, что информация отображается на экране в виде, пригодном для непосредственного использования. Не следует заставлять пользователя дополнительно обрабатывать эту информацию, например, уточнять по справочникам значения кодов, производить какие-либо преобразования, пересчеты и т.п. Формат для вывода даты, времени и других подобных стандартизованных данных должен быть общепринятым, а не индивидуальным для данной системы. Общепринятая система сочетания больших и малых букв в тексте улучшает его восприятие.

Очень серьезным вопросом, во многом определяющим качество восприятия информации, является рациональное размещение данных на экране. Требуемая плотность расположения данных — понятие субъективное. Она зависит от конкретного пользователя и решаемой задачи. Однако существуют некоторые правила, регулирующие плотность расположения данных на экране (или в пределах окна):

- оставлять пустым приблизительно половину экрана (окна);
- оставлять пустую строку после каждой пятой строки таблицы;

оставлять четыре-пять пробелов между столбцами таблицы. Фрагменты текста должны располагаться на экране так, чтобы взгляд пользователя сам перемещался в нужном направлении. Содержимое полей должно не «прижиматься» к краю экрана, а располагаться около его горизонтальных или вертикальных осей. Меню, содержащее относительно небольшой объем информации, должно смещаться в левую верхнюю часть экрана. Чтобы подчеркнуть симметрию, содержимое и наименования полей,

относящихся к одной группе, должны выравниваться по вертикали. По возможности необходимо выравнивать все логически связанные группы данных.

Другой набор рекомендаций определяется факторами, связанными с право-левой асимметрией головного мозга человека. Известно, что левое и правое полушария по-разному участвуют в восприятии и переработке информации. В частности, при запоминании слов ведущую роль играет левое полушарие, а при запоминании образов более активно правое. Информация с правой части экрана поступает непосредственно в левое полушарие, а с левой части — в правое (естественно, при бинокулярном зрении оператора). В связи с этим можно рекомендовать текстовые сообщения группировать справа, а изображения — слева. У некоторых людей это распределение функций полушарий противоположно, у женщин асимметрия выражена слабее, чем у мужчин. Этот факт еще раз подтверждает необходимость индивидуализации характера отображения информации. Учет право-левой асимметрии памяти имеет существенное значение, если интервалы следования сообщений не превышают 10с. Поэтому приведенные рекомендации следует в первую очередь учитывать в интерфейсах программ, работающих *в режиме реального времени*.

Рациональное размещение данных на экране является наиболее важным, но не единственным методом обеспечения удобства и естественности пользовательского интерфейса. Современные мониторы предоставляют в распоряжение разработчика различные методы выделения выводимой информации на экране.

Выделение информации — это использование таких атрибутов, которые позволяют привлечь внимание пользователя к некоторой области экрана. В качестве подобных атрибутов могут выступать: цвет символов, цвет фона, уровень яркости, мерцание и применение различных шрифтов для выводимых символов. Часто для выделения информации используют подчеркивание, вывод в инверсном виде, различные рамки и «тени». Эффект применения этих атрибутов различен, а их сочетание — часто непредсказуемо и зависит от индивидуальных особенностей пользователей. В литературе по этому поводу приведено значительное число рекомендаций, основной смысл которых сводится к следующему положению: следует стараться использовать минимально необходимое число атрибутов (чтобы привлечь внимание человека, достаточно лишь легонько его «коснуться»).

Существуют ли объективные критерии оценки плотности заполнения экрана, сбалансированности данных и других показателей качества форматирования экрана? Проблема заключается в том, что на любого человека, который смотрит на экран, оказывает влияние содержание информации, затрудняя такую оценку. Один из возможных подходов к решению этой проблемы — отделить содержание от формы. Для этого применяются два метода — прямоугольников и выделенных точек.

При использовании **метода прямоугольников** после разбиения экрана на поля каждое из них заполняется произвольным текстом и отделяется от других по всему периметру, по крайней мере, одним пробелом. Через центр экрана мысленно проводятся оси, позволяющие оценить сбалансированность размещения полей.

Метод выделенных точек позволяет определить число и размещение областей экрана, к которым будет привлечено внимание пользователя (из-за повышенной яркости, цвета или мерцания символов). Для этого каждая область, требующая повышенного внимания, моделируется группой символов, отличных от пробела.

Рассмотренные методы позволяют устранить грубые ошибки в форматировании экрана, однако лучший способ оценить его качество — дать возможность потенциальному пользователю поработать с системой.

Некоторые технические аспекты использования визуальных атрибутов отображаемой информации рассматриваются в следующей главе.

3. ПРОЕКТИРОВАНИЕ ГРАФИЧЕСКОГО ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

3.1. ОСОБЕННОСТИ ГРАФИЧЕСКОГО ИНТЕРФЕЙСА

В основу разработки практически любого графического пользовательского интерфейса (GUI — Graphical User Interface) положены три метафоры: «рабочий стол», «работаешь с тем, что видишь», «видишь, что получил».

Метафора «рабочий стол», в частности, означает следующее.

Для человека, сидящего за рабочим столом, доступны как определенные источники информации, так и средства обработки этой информации. При этом на рабочем столе могут находиться документы, представленные в различной форме: текст, таблицы, графики, рисунки и т.д., относящиеся либо к различным задачам, либо к разным аспектам одной и той же задачи. В любом случае человек имеет возможность просмотреть любой из этих документов, сделать пометки или выборку из них, сравнить содержащиеся в них данные.

Другие две метафоры являются развитием идеи «рабочего стола».

В каждый момент времени сидящий за столом может работать только с теми документами, которые он видит перед собой. Если необходимый документ в данный момент отсутствует на столе, его предварительно требуется достать из ящика стола, из папки или из портфеля.

Выполняя какие-то действия над документами, человек, естественно, тут же видит результат своей деятельности.

В рамках графического интерфейса пользователя все три метафоры получили достаточно адекватное воплощение: пространство экрана монитора — это **рабочий стол** пользователя, необходимые для решения задачи объекты представлены на нем в виде соответствующих графических образов (**пиктограмм и окон**), а чтобы изменить рабочую среду, пользователю достаточно изменить состав объектов, представленных на рабочем столе; при этом все необходимые действия выполняются не с помощью команд, а путем **прямого манипулирования** объектами (точнее, их образами).

Прямое манипулирование объектами обладает следующими достоинствами:

- Обеспечивает визуальный контроль за выполняемыми операциями.
- Позволяет легко восстановить предшествующее состояние «рабочего стола».
- Позволяет решать различные задачи, используя ограниченный набор стандартных операций (открытие/закрытие окна, буксировка объекта, изменение атрибутов окна или объекта и т.п.).

Еще одна важная особенность современных графических интерфейсов — это **многоооконность**.

Многооконая технология обеспечивает пользователю доступ к большему объему информации, чем при использовании одного экрана. Кроме того, имея через окна доступ к нескольким источникам информации одновременно, пользователь может объединять имеющуюся в них информацию. Например, изображения, полученные с помощью графического редактора, можно включить в текстовый документ.

С помощью нескольких окон пользователь может также одновременно анализировать информацию, представленную на разных уровнях детализации. Наличие на экране нескольких окон или пиктограмм позволяет «расширить» кратковременную память пользователя.

Таким образом, графический интерфейс расширяет пространство обзора и облегчает работу пользователя. Вместе с тем, само по себе предоставление пользова-

телю графического интерфейса еще не гарантирует повышения эффективности его работы. Это обусловлено тем, что метафора «рабочий стол» далеко не всегда означает «аккуратный рабочий стол». Если «рабочий стол» плохо организован, существует опасность, что пользователь будет тратить больше времени на работу с «папками», чем на решение стоящих перед ним задач.

Прямое манипулирование также не всегда удобно, прежде всего, для опытного пользователя. Часто некоторую последовательность действий можно выполнить быстрее с помощью одной команды (или макроса), чем посредством серии манипуляций мышью.

Прямое манипулирование снижает также гибкость диалога, поскольку изначально графический интерфейс базируется на структуре меню.

С появлением инструментов визуального программирования, таких как Visual C, Visual Basic, Power Builder, Delphi, создание графического пользовательского интерфейса перестало быть прерогативой лишь немногих специалистов. Но одновременно с этим возникла проблема, способная свести на нет все преимущества быстрой разработки GUI. Эта проблема заключается в появлении большого количества плохих графических интерфейсов: не согласованных, не дружественных пользователю, громоздких, слабофункциональных, не помогающих, а мешающих пользователю решать стоящие перед ним задачи. Изложению некоторых рекомендаций, позволяющих повысить качество разработки GUI, как раз и посвящена данная глава.

3.2. ОБЪЕКТНЫЙ ПОДХОД К ПРОЕКТИРОВАНИЮ ИНТЕРФЕЙСА

КОНЦЕПЦИЯ ИНТЕРФЕЙСА, УПРАВЛЯЕМОГО ДАННЫМИ

Разработка, управляемая данными (сокращенно DCD — Data-centered Design) означает, что проектирование интерфейса поддерживает такую модель взаимодействия пользователя с системой, при которой первичными являются обрабатываемые данные, а не требуемые для этого программные средства. Другими словами, при таком подходе основное внимание пользователя концентрируется на тех данных, с которыми он работает, а не на поиске и загрузке необходимого приложения.

При использовании DCD-технологии основным программным объектом является **документ**, который представляет собой некоторое абстрактное устройство хранения данных, используемых для выполнения заданий пользователей и для их взаимодействия. Документ должен быть доступен как различным приложениям, используемым для его обработки, так и всем взаимодействующим пользователям.

ОБЪЕКТЫ И ОТНОШЕНИЯ МЕЖДУ НИМИ

Рассмотренные выше особенности графических интерфейсов, а также положенная в основу их реализации DCD-технология обуславливают необходимость применения для проектирования GUI объектно-ориентированного подхода. Такой подход предполагает использование аналогий между программными объектами и объектами реального мира. С точки зрения пользовательского интерфейса, объектами являются не только файлы или пиктограммы, но и любое устройство для хранения и обработки информации, включая ячейки, параграфы, символы, и т.д., а также документы, в которых они находятся.

Объекты, независимо от того, относятся ли они к реальному миру или имеют компьютерное воплощение, обладают определенными характеристиками, которые помогают нам понимать, что они собой представляют, и как они ведут себя в тех или иных ситуациях. Следующие понятия описывают основные аспекты и характеристики объектов, имеющих компьютерное воплощение:

- **Свойства объектов.** Объекты имеют определенные характеристики или атрибуты, называемые свойствами, которые определяют их представление или возможные состояния (например, цвет, размер, дату модификации). Свойства не ограничены внешними или видимыми признаками объекта. Они могут отражать их внутреннюю организацию или текущее состояние объекта.

- **Операции над объектами.** Все действия, которые могут быть выполнены с (или над) объектом, считаются допустимыми операциями. Перемещение или копирование объекта являются примерами операций. Пользователь может выполнять операции над объектами, используя те или иные механизмы, предоставляемые интерфейсом, (в частности, командное управление и прямое манипулирование).

- **Связь (отношения) между объектами.** Любой объект тем или иным образом взаимодействует с другими объектами. Во многих случаях взаимоотношения между объектами могут быть описаны как связь определенного типа. Наиболее общими типами отношений являются наборы (Collection), объединения (Constraints), и композиции (Composites).

Набор представляет собой наиболее простой тип отношения, которое отражает наличие у объектов некоторых общих свойств. Результаты запроса (поиска) по образцу или операции множественного выбора объектов — примеры использования данного типа отношения. Важным достоинством этого типа отношения является то,

что он позволяет указывать операции, которые должны относиться к определенному набору объектов.

Объединение отражает более «тесное» отношение между объектами, при котором изменение объекта влияет на некоторый другой объект в наборе. Простейший пример такого отношения — изменение формата соседней страницы при добавлении текста на предыдущей странице документа.

Композиция имеет место в том случае, когда агрегация нескольких объектов может рассматриваться как новый объект со своим собственным множеством свойств и допустимых операций. Столбец ячеек в таблице и параграф в текстовом документе — это примеры композиций.

Еще один распространенный тип отношений между объектами — контейнер.

Контейнер является объектом, который содержит другие объекты (например, рисунок в документе или документ в папке могут рассматриваться как часть содержимого соответствующего контейнера). Свойства контейнера часто влияют на поведение его содержимого. Это влияние может заключаться в расширении или подавлении некоторых свойств содержащихся в нем объектов или в изменении перечня допустимых операций. Кроме того, контейнер управляет доступом к своему содержимому, а также преобразованием типа (формата) включаемого в него объекта. Это, в частности, может сказаться на результате пересылки объекта из одного контейнера в другой.

Рассмотренные выше аспекты обуславливают необходимость отнесения каждого объекта к тому или иному типу (классу) объектов. Объекты одного типа имеют аналогичные свойства и поведение.

Как и в реальном мире, совокупность объектов (возможно, различных типов) образует некоторую среду (окружение) пользователя. Исходя из этого, большинство заданий пользователя могут быть представлены (описаны) как определенная комбинация взаимосвязанных объектов. Так, например, обработка текстового документа может быть описана как композиция операций, выполняемых над его элементами (отдельными словами, параграфами и т.д.). Благодаря такому подходу любые, сколь угодно сложные конструкции могут быть реализованы на основе небольшого числа базовых соглашений. При условии последовательной и согласованной реализации этих соглашений для всего пользовательского интерфейса эффективность работы пользователя существенно возрастает. Кроме того, указанный подход способствует модульной, компонентно-ориентированной разработке приложения, то есть новое задание может быть выполнено путем адаптации или комбинации тех же объектов.

В реальном мире объекты сохраняют свое текущее состояние до тех пор, пока оно не будет изменено под влиянием каких-либо внешних воздействий. Например, если вы, уходя из дому, закрыли окно, оно, скорее всего, останется в таком же состоянии до вашего возвращения. Это же правило должно быть справедливо и для объектов интерфейса. За исключением тех случаев, когда требуется явное указание пользователя на сохранение данных, все объекты окружения должны быть сохранены автоматически. Кроме того, должна сохраняться и визуальная информация о состоянии окружения, такая, например, как позиция курсора, расположение и размер окна, с тем, чтобы это состояние могло быть восстановлено при последующем сеансе работы пользователя.

При всех достоинствах объектного подхода к разработке интерфейса, его использование само по себе не гарантирует требуемого качества интерфейса. Для создания эффективного пользовательского интерфейса необходимо дополнить объектный подход тщательным проектированием всех компонентов интерфейса с ориентацией на потребности потенциального пользователя.

Первым шагом в объектно-ориентированном проектировании интерфейса должен быть анализ целей пользователей и особенностей выполняемых ими заданий. При проведении такого анализа следует определить основные компоненты или объекты, с которыми взаимодействует пользователь, а также характерные особенности объектов каждого типа. Необходимо также выявить перечень операций, выполняемых над объектами, их влияние на состояние и свойства объектов.

После завершения анализа можно переходить к описанию возможных способов взаимодействия пользователя с объектами различных типов. На этом шаге выбирается форма визуального представления объектов. При этом следует иметь в виду, что визуальный образ объекта в зависимости от ситуации может изменяться. Например, контейнер может быть представлен и в виде пиктограммы, и в виде окна, отображающего содержимое этого контейнера.

Следующим этапом проектирования GUI является компоновка и пространственное размещение на экране визуальных элементов интерфейса. Именно на этом этапе должны быть решены такие проблемы, как выбор цвета, размера и других атрибутов этих элементов, а также выбор средств и методов привлечения внимания пользователя к наиболее важной информации, отображаемой на экране.

Проектируя размещение информации на экране, необходимо предусмотреть возможность удобного доступа пользователя к средствам помощи, независимо от того, на каком шаге выполнения задания он находится, и какая именно информация представлена на экране.

3.3. КОМПОНЕНТЫ ГРАФИЧЕСКОГО ИНТЕРФЕЙСА

РАБОЧИЙ СТОЛ

Рабочий стол предоставляет пользователю первичную рабочую область; он заполняет экран и формирует визуальный фон для всех выполняемых операций (как показано на рис. 3.1). Тем не менее, Рабочий стол является не просто фоном. Он может также быть использован в качестве основы для размещения объектов файловой системы. Кроме того, для компьютера, подключенного к сети, Рабочий стол служит в качестве частной рабочей области, через которую пользователь может получить доступ к другим объектам сети.

Рис. 3.1. Рабочий стол

ПИКТОГРАММЫ

Пиктограммы используются для визуального представления на экране объектов или задач. Как правило, это небольшие законченные рисунки, отображающие сущность представляемых объектов или явлений. Поскольку пиктограммы являются одним из основных средств взаимодействия пользователя с приложением, важно не только обеспечить поддержку существующих (системных) пиктограмм, но и разработать новые; от того, насколько они будут соответствовать своему назначению, будет зависеть и эффективность работы пользователя.

ОКНА

В реальном мире взгляд через разные окна позволяет получить различные изображения внешнего мира. Аналогичную роль играют окна и в графическом интерфейсе.

формально понятие окна трактуется следующим образом: окно есть специальная область физического экрана, с помощью которой пользователь имеет возможность получить отображение определенного аспекта решаемой задачи.

Другими словами, окно является средством просмотра и редактирования информации, а также отображения содержимого и свойств объектов. Окна могут использоваться также для вывода на экран значений параметров, результатов выполнения команд, наборов инструментов и сообщений, информирующих пользователя о конкретной ситуации. Рис. 3.2 демонстрирует различные варианты использования окон.

Пользователь может взаимодействовать с объектами приложения, используя различные устройства ввода. Наиболее распространенные из них — мышь, клавиатура и перо.

Рис. 3.2. Использование окон приложением

МЫШЬ

Мышь является основным устройством ввода при использовании графического интерфейса. Другие типы устройств указания, которые эмулируют мышь (например, трэкболлы), также подпадают под этот общий термин.



Мышь функционально связана с графическим символом на экране, который называется **указателем**. Позиционируя указатель и нажимая или *щелкая* кнопку мыши, пользователь может выбирать объекты и операции.

По мере того, как пользователь перемещает указатель через экран, его форма может изменяться, чтобы обеспечить информирование пользователя (**обратную связь**) о конкретной позиции, операции или состоянии. Табл. 3.1 содержит некоторые наиболее распространенные формы указателя, используемые в MS Windows, и их назначение.

Таблица 3.1.

Формы указателя

Форма	Позиция на экране	Допустимое или выполняемое действие
	Поверх большинства объектов	Указание, выбор или перемещение
	Над текстом	Выбор текста
	Над любым объектом или позицией	Выполнение операции
	Над любым объектом или позицией	Выполнение операции в фоновом режиме; указатель остается интерактивным
	Поверх большинства объектов	Режим контекстно-зависимой помощи
	Внутри окна	Масштабирование изображения
	Поверх регулируемой границы	Изменение размера по вертикали
	Поверх регулируемой границы	Изменение размера по горизонтали
	Поверх регулируемой границы	Изменение размера по диагонали
	Поверх регулируемой границы	Изменение размера по диагонали
	Вдоль границы столбца	Изменение ширины столбца
	Вдоль границы строки	Изменение высоты строки
	Над блоком разбиения окна	Изменение размера (ширины) вертикальных панелей

	Над блоком разбиения окна	Изменение размера (высоты) горизонтальных подокон
	Над объектом-приемником	Приемник недоступен для перетаскиваемого объекта

Каждый указатель имеет определенную точку, называемую *горячей точкой*, которая идентифицирует точную позицию указателя мыши на экране. Горячая точка определяет, на какой объект воздействует пользователь посредством мыши. Для объектов на экране может быть также определена горячая зона; *горячая зона* — это область, в пределах которой должна (или может) находиться горячая точка, чтобы считаться расположенной над объектом. Как правило, горячая зона совпадает с границами объекта, но может быть как больше, так и меньше, чтобы облегчить работу пользователя.

Взаимодействие пользователя с приложением посредством мыши основано на использовании левой и правой кнопок мыши (в дальнейшем соответственно ЛКМ и ПКМ). Действия, выполняемые с помощью ПКМ, обычно дублируют функции, которые доступны через ЛКМ, но могут быть реализованы альтернативным способом. Система позволяет пользователю изменять распределение кнопок.

В табл. 3.2 приведено описание основных действий, выполняемых с помощью мыши.

Таблица 3.2.

Основные действия, выполняемые с помощью мыши

Действие	Описание
Указание (Pointing)	Установка указателя на конкретный объект на экране без использования кнопок мыши. Указание обычно является частью подготовки к выполнению некоторого другого действия. Для указания во многих случаях имеется возможность обеспечить визуальную обратную связь с пользователем (например, посредством изменения формы указателя)
Щелчок (Clicking)	Щелчок одной из кнопок мыши на объекте; как правило, в момент щелчка мышь не перемещается; щелчок идентифицирует (выбирает) или активизирует объект
Двойной щелчок (Double-Clicking)	«Сдвоенный» щелчок одной из кнопок мыши на объекте; объект может быть как выбран ранее, так и не выбран; двойной щелчок обеспечивает выполнение предопределенной операции для данного объекта
Нажатие (Pressing)	Предполагает установку указателя над одним из объектов (или позиций) на экране с последующим нажатием кнопки мыши; кнопка удерживается в нажатом состоянии достаточно продолжительное время (не менее 1с); обычно нажатие является начальной фазой операции выбора
Перемещение (Dragging)	Перемещение указателя при нажатой кнопке мыши; как правило, используется для выполнения операций выбора и прямого манипулирования

Для большинства операций, выполняемых с помощью мыши, нажатие кнопки только идентифицирует начало выполнения операции. Сама же операция выполняется при освобождении кнопки. Исключением является функция **автоповторения**. Например, установив указатель на стрелку полосы прокрутки и удерживая нажатой кнопку мыши, можно обеспечить непрерывную прокрутку до тех пор, пока кнопка не будет отпущена.

Теоретически, возможны и другие приемы работы с мышью, например, **аккорд** (нажатие нескольких кнопок мыши одновременно) и **мульти-щелчок** (тройной или даже «учетверенный» щелчок), однако они требуют наличия соответствующих моторных навыков пользователя и на практике используются очень редко.

КЛАВИАТУРА

Клавиатура — это основное средство ввода или редактирования текстовой информации. Тем не менее, при реализации графического интерфейса она может использоваться для ввода других типов данных, а также для управления, переключения режимов и как средство ускоренного доступа к объектам и операциям.

С точки зрения организации взаимодействия пользователя с объектами приложения все клавиши можно разделить на следующие функциональные группы:

- текстовые клавиши и клавиши пунктуации;
- клавиши навигации;
- функциональные клавиши;
- клавиши доступа;
- клавиши режима;
- клавиши-акселераторы.

Клавиша доступа (иногда именуемая также мнемонической клавишей) — это текстовая клавиша, которая при использовании в комбинации с клавишей <Alt> активизирует соответствующий элемент управления. Клавиша доступа должна соответствовать одному из символов текстовой метки этого элемента управления. Например, комбинация клавиш <Alt> + О может активизировать элемент управления, имеющий метку *Открыть*. Эффект активизации элемента управления зависит от типа этого элемента.

Клавиша доступа должна быть уникальной в пределах своей области действия (т.е. одна и та же клавиша не должна использоваться для доступа более чем к одному элементу управления, расположенному в этой области). В противном случае с помощью данной клавиши всегда будет активизироваться только один элемент — расположенный первым в этой области.

Клавиши режима изменяют способ действия других клавиш (или других устройств ввода). Различают два типа таких клавиш: клавиши-переключатели и клавиши-модификаторы.

Клавиша-переключатель включает или выключает конкретный режим при каждом очередном нажатии. Например, нажатие клавиши <Caps lock> приводит к переключению с верхнего регистра на нижний и наоборот.

Клавиша-модификатор, в отличие от клавиш-переключателей, устанавливает режим, который остаётся в силе, только пока клавиша-модификатор удерживается в нажатом состоянии. К ним относятся клавиши <Shift>, <Ctrl> и <Alt>. Их использование, во многих случаях, оказывается предпочтительнее из-за того, что они требуют привлечения внимания пользователя к выбору режима и, вместе с тем, позволяют легко отменить выбранный режим.

Клавиши-акселераторы (называемые также *горячими клавишами*) — это клавиши или комбинации клавиш, которые обеспечивают быстрый доступ к часто выполняемым операциям. В качестве таких комбинаций рекомендуется использовать <Ctrl>+<символ> и функциональные клавиши (с F1 по F12). По определению, клавиши-акселераторы являются «клавиатурным эквивалентом» других элементов пользовательского интерфейса. Исходя из этого, избегайте использования клавиши-акселератора как единственного средства доступа к какому-либо объекту или функции.

При назначении клавиш-акселераторов следует учитывать приведенные ниже рекомендации.

- Используйте комбинацию <Shift>+<клавиша> для расширения или дополнения действия, выполняемого с помощью этой <клавиши> без <Shift>. Например, если комбинация клавиш <Alt>+<Tab> обеспечивает переключение между окнами сверху вниз, то комбинация <Shift>+<Alt>+<Tab> переключает окна в обратном порядке.

- Используйте комбинации <Ctrl>+<клавиша> для усиления эффекта действия, выполняемого с помощью <клавиши>. Например, при редактировании текста клавиша <Home> обеспечивает переход на начало строки, а комбинация <Ctrl>+<Home> — в начало текста.

- Избегайте комбинаций <Alt>+<клавиша>, поскольку они могут конфликтовать со стандартным клавиатурным доступом к меню и элементам управления интерфейса. Комбинации <Alt>+<Tab>, <Alt>+<Esc> и <Alt>+<Spacebar> зарезервированы для системного использования; комбинации <Alt>+<цифра> обеспечивают ввод специальных символов.

- Учитывайте назначения клавиш-акселераторов, использованные разработчиками системного программного обеспечения. Например, в ОС MS Windows комбинация <Ctrl>+C используется для выполнения команды *Copy* (Копировать).

- Предоставляйте пользователю право изменять назначения клавиш-акселераторов в вашем приложении, когда это возможно.

- Используйте клавишу <Esc> для прерывания выполняемой операции; как правило, она также используется в качестве клавиши-акселератора для выполнения команды *Cancel* (Отменить).

- Сделайте ваше приложение нечувствительным к случайным (ошибочным) комбинациям клавиш.

Некоторые клавиатуры содержат три новых клавиши: клавишу <Application> (Приложение) и две клавиши <Windows> (Окна). Основное назначение клавиши <Application> — вызов всплывающего меню для текущего выбора (т.е. она аналогична комбинации <Shift>+F10). Вы можете также использовать ее совместно с клавишами-модификаторами для выполнения специализированных функций приложения. Нажатие любой из клавиш <Windows> — левой или правой — приводит к появлению меню. Эти клавиши также используются системой как модификаторы для специфических системных функций. Не используйте эти клавиши в качестве модификаторов для функций несистемного уровня.

3.4. ВЗАИМОДЕЙСТВИЕ ПОЛЬЗОВАТЕЛЯ С ПРИЛОЖЕНИЕМ

Основные операции взаимодействия пользователя с приложениями, такие как навигация, выбор, просмотр, редактирование, создание новых объектов базируются на парадигме объектной обработки, в которой пользователь идентифицирует объект и действие, относящееся к этому объекту. Последовательно реализуя эту технику, вы даете возможность пользователю переносить их навыки и знания на новые задания.

Большинство создаваемых приложений поддерживают основные операции взаимодействия для мыши, клавиатуры и пера. Дополняя или расширяя основной набор операций, учитывайте возможность их реализации с помощью указанных устройств ввода. Техника использования не должна быть единой для всех устройств. Наоборот, она должна быть реализована таким образом, чтобы оптимизировать применение конкретного устройства с учетом его особенностей. Кроме того, старайтесь облегчить пользователю переход между устройствами, с тем, чтобы он мог менять устройства в зависимости от выполняемых действий.

НАВИГАЦИЯ

Перемещая мышь, пользователь может переместить указатель в любую позицию на экране.

Навигация с помощью пера подобна навигации посредством мыши, за исключением того, что пользователь перемещает перо, не касаясь входной поверхности.

Клавиатурная навигация требует от пользователя нажатия специальных клавиш и их комбинаций. Положение позиции (фокуса) ввода определяется контекстом (текущей ситуацией); в частности, при работе с текстом оно идентифицируется положением текстового курсора.

Основные клавиши навигации

Клавиши навигации — это четыре клавиши управления курсором (которые мы в дальнейшем для краткости будем называть <Вправо>, <Влево>, <Вверх>, <Вниз>), а также клавиши <Home>, <End>, <Page Up>, <Page Down> и клавиша <Tab>. Нажатие клавиш навигации в сочетании с клавишей <Ctrl> позволяет увеличивать шаг перемещения. Например, нажатие клавиши <Вправо> перемещает курсор в текстовой области вправо на один символ, а нажатие той же клавиши совместно с <Ctrl> обеспечивает перемещение курсора на одно слово.

Табл. 3.3 содержит перечень основных клавиш навигации и их функции. Вы можете определить дополнительные клавиши для навигации.

Таблица 3.3

Основные клавиши навигации

Клавиша	Перемещение курсора	Перемещение курсора для комбинации <Ctrl>+<клавиша>
<Вправо>	на один элемент вправо	на один (более крупный) элемент вправо
<Влево>	на один элемент влево	на один (более крупный) элемент влево
<Вверх>	вверх на один элемент или строку	на один (более крупный) элемент вверх
<Вниз>	вниз на один элемент или строку	на один (более крупный) элемент вниз
<Home>	в начало строки	в начало данных или файла (в самую верхнюю позицию)
<End>	в конец строки	в конец данных или файла (в самую нижнюю позицию)
<Page Up>	на один экран вверх (в ту же позицию предшествующего экрана)	на один экран влево (или на предшествующий экран)

<Раде Down>	на один экран вниз (в ту же позицию следующего экрана)	на один экран вправо (или на следующий экран)
<Tab>	на следующее поле (комбинация <Shift>+<Tab> дает перемещение в обратном направлении)	на следующую большую область

В отличие от навигации с помощью мыши и пера, клавиатурная навигация обычно влияет на текущий выбор. В связи с этим вы можете дополнительно определить использование клавиши <Scroll Lock> таким образом, чтобы навигация выполнялась без изменения текущего выбора. При этом величина шага приращения остается прежней.

ВЫБОР

Выбор является основным средством, с помощью которого пользователь идентифицирует интересующие его объекты. Следовательно, реализация модели взаимодействия, основанной на использовании выбора — один из наиболее важных аспектов проектирования интерфейса.

Выбор, как правило, предполагает прямое указание пользователем идентифицируемого объекта. Этот механизм известен как **явный выбор**. Если объект выбран, пользователь может определить действие для него.

Возможны также ситуации, когда идентификация (выбор) объекта производится «косвенно», на основе некоторого логического правила или исходя из текущего контекста. Косвенный выбор работает наиболее эффективно в тех случаях, когда между объектом и действием существует простая и видимая ассоциация. Например, когда пользователь «протаскивает» полосу прокрутки, он одновременно определяет и выбор объекта «полоса прокрутки», и связанное с ним действие «перемещение». Косвенный выбор может быть реализован и посредством имеющейся связи между объектами. Например, выбирая символ в текстовом документе, вы, тем самым, подразумеваете выбор параграфа, в котором содержится данный символ.

Операция выбора может относиться как к единственному объекту, так и к множеству объектов. Соответственно различают **единичный** и **множественный** выбор. Множественный выбор может быть **непрерывным** (известен также как **выбор области**), когда операция выбора выполняется для группы расположенных рядом объектов, либо **раздельным**, когда выбор содержит группу объектов, которые пространственно или логически разнесены.

Множественный выбор может также быть классифицирован как **однородный** (гомогенный) или **разнородный** (гетерогенный), в зависимости от типа или свойств выбранных объектов. Однородность или разнородность выбора влияет на доступность операций, выполняемых над всеми выбранными объектами.

Всегда обеспечивайте визуальную обратную связь для отображения результатов явного выбора, с тем, чтобы пользователь мог контролировать свои действия. Форма отображения выбора зависит от объекта и текущего контекста.

При наличии косвенного выбора обеспечивать визуальную обратную связь значительно сложнее; однако вы можете отобразить эффект косвенного выбора другими способами. Например, когда пользователь протаскивает полосу прокрутки, целесообразно отобразить перемещение указателя. Аналогично, если указание слова в параграфе подразумевает выбор параграфа, вы не должны использовать средства выделения для всего параграфа, достаточно учитывать свойства параграфа, когда пользователь выполняет над ним те или иные действия.

Область выбора — это совокупность объектов одного окна, для которых сделан выбор; при этом область выбора не обязательно будет совпадать с областью видимости этих объектов. Например, вы можете выбрать два файла в одном и том же окне, удаленных друг от друга настолько, что в пределах видимости находится только один из них.

Одновременно может существовать несколько областей выбора. При этом в каждом окне может быть определена только одна область выбора. Область выбора в одном окне не зависит от областей выбора в других окнах.

Учитывать имеющиеся области выбора важно из-за того, что они определяют набор допустимых операций для выбранных объектов и способ выполнения этих операций»

Как правило, области выбора содержат объекты, относящиеся к одному уровню иерархии (например, только файлы внутри папки, либо папки, имеющие равный уровень вложенности). Тем не менее, вы можете предоставить пользователю возможность распространить область выбора на объект следующего, более высокого уровня, если он непосредственно содержит исходный объект выбора (но в пределах того же окна). При этом у пользователя должна сохраняться возможность возврата на исходный уровень. Например, если требуется распространить область выбора, состоящую из одной ячейки в таблице, на соседнюю ячейку (как показано на рис. 3.3), необходимо сначала поднять выбор с символьного уровня на уровень ячейки; при изменении уровня выбора в обратном направлении, следует восстановить выбор на символьном уровне.

Electricity	Telephone
└─	

Electricity	Telephone
Electricity	Telephone
└─	

Рис. 3.3. Иерархический выбор

ОСНОВНЫЕ КОНЦЕПЦИИ ВЫБОРА С ПОМОЩЬЮ МЫШИ

Выбор с помощью мыши основан на использовании двух основных действий: нажатии клавиши мыши и перемещении. В общем случае, нажатие обеспечивает выбор единственного объекта или позиции, а перемещением идентифицируется область, состоящая из всех объектов, начиная от позиции «кнопка нажата» до позиции «кнопка отпущена».

Предоставьте пользователю возможность использовать для выбора обе кнопки мыши. Когда пользователь нажимает кнопку мыши, зафиксируйте начальную точку области выбора. Если, нажав кнопку, пользователь перемещает мышь, расширьте область выбора до объекта, ближайшего к текущей позиции указателя. Если, продолжая удерживать кнопку, пользователь перемещает мышь в пределах области выбора, ограничьте ее объектом, ближайшим к указателю. Согласование области выбора с перемещением указателя при нажатой кнопке мыши позволяет пользователю динамически регулировать область выбора.

Если, завершив выбор, пользователь нажимает вторую (правую) кнопку мыши, отобразите контекстное всплывающее меню для выбранных объектов.

Описанная выше общая форма выбора оптимальна для указания единственного объекта или единственной области. В том случае, если новая область выбора создается в пределах уже существующей области (например, в пределах того же окна), она

отменяет предыдущий выбор. Такой подход обеспечивает простой выбор, который должен выполняться быстро и легко. Ту же технику можно использовать и для отмены выбора: если пользователь нажимает кнопку мыши за пределами любой существующей области выбора (но в том же окне), результат выбора должен быть аннулирован. Вместе с тем, при повторном нажатии кнопки мыши над выбранным пунктом не следует отменять прежний выбор. Лучше определите операции, выполняемые над выбранным объектом (областью) при нажатии правой или левой кнопки мыши.

Если пользователь нажимает первую (левую) кнопку мыши и указатель при этом не перемещается, то последующее освобождение кнопки может иметь различный эффект, который определяется контекстом выбора. Вы можете использовать один из следующих вариантов, в зависимости от сущности выполняемого пользователем задания:

- Игнорировать данное событие. Это наиболее распространенный и самый безопасный вариант.
- Объект под указателем может получить некоторое специальное обозначение или признак.
- Выбор может быть отнесен только к объекту, находящемуся под указателем. Как правило, при щелчке ПКМ на области выбора целесообразно отображать для этой области всплывающее меню.

Хотя выбор обычно выполняется посредством установки указателя над объектом, он может быть сделан косвенно, на основе логической связи между объектом и положением указателя. Например, выбирая текст, пользователь может установить указатель в свободной области после конца строки; при этом результат выбора будет таким же, как при указании на конец строки.

Корректировка выбора

Корректировка выбора (добавление или удаление элементов выбора) выполняется на основе совместного использования мыши и клавиш-модификаторов. Например, клавиша <Ctrl> может использоваться как переключатель режима: если пользователь нажимает эту клавишу, выбирая новый объект, добавьте его к существующему выбору. Однако имейте в виду, что непересекающийся выбор может быть полезен далеко во всех ситуациях.

Если выбор, модифицированный клавишей <Ctrl>, выполнен перемещением указателя, состояние выбора изменяется на противоположное для всех объектов, попавших в область выбора.

При использовании клавиши <Ctrl> для изменения выбора следует нажать клавишу прежде, чем использовать кнопку мыши. Непересекающийся выбор выполняется до тех пор, пока пользователь не отпустит кнопку мыши (даже если будет отпущена клавиша <Ctrl>).

Клавиша <Shift> позволяет расширить область выбора. Когда пользователь нажимает ЛКМ, удерживая эту клавишу, фиксируется текущая позиция указателя. Она соответствует началу области выбора. Последующее нажатие ЛКМ при нажатой клавише <Shift> указывает последний объект (граничную точку) области выбора.

Состояние конкретного объекта в области выбора связано с состоянием первого объекта, включенного в область выбора. Если первый объект отображен как выбранный, остальные объекты области также отображаются как выбранные.

При корректировке выбора пользователь должен нажать и удерживать клавишу <Shift> прежде, чем нажать ЛКМ. Корректировка выбора продолжается до тех пор,

пока пользователь не отпустит кнопку мыши. Для обозначения объекта (или позиции), с которого было начато выделение области, используется понятие **якоря**. Все последующие корректировки выбора, выполняемые с помощью клавиш <Shift> и <Ctrl>, производятся относительно позиции-якоря.

Рис. 3.4 показывает, как эта техника может применяться в электронной таблице.

- а) пользователь выбирает 4 ячейки, перемещая указатель от ячейки А2 к ячейке В3;
- б) пользователь нажимает клавишу <Shift> и, не отпуская ее, щелкает ЛКМ на ячейке С4;
- в) пользователь нажимает клавишу <Ctrl> и, не отпуская ее, щелкает ЛКМ на ячейке А6;
- г) пользователь нажимает клавишу <Shift> и, не отпуская ее, щелкает ЛКМ на ячейке С6.

Выбор области

В некоторых более сложных ситуациях, когда объекты (точнее, их пиктограммы) могут перекрываться, начальная точка области выбора может находиться в фоновой области окна (иногда называемой *белым полем*). В таких случаях для визуального отображения области выбора на экране рисуется условная граница области. Обычно она имеет вид пунктирного прямоугольника, но возможны и другие формы.

Рис. 3.4. Расширение выбора в пределах электронной таблицы

Когда пользователь нажимает ЛКМ и перемещает указатель, должна изменяться и граница области выбора (рис. 3.5).

После того, как выбор сделан, условная граница области должна быть убрана. Впоследствии выбор может быть скорректирован с помощью клавиш-модификаторов <Ctrl> и <Shift>.

Дополнительно следует определить, должен ли выбираемый объект полностью лежать в границах выделяемой области, или достаточно, чтобы он только пересекался ею.

КЛАВИАТУРНЫЙ ВЫБОР

Клавиатурный выбор объектов основан на использовании понятия **фокус ввода**. Фокус ввода может быть представлен на экране в виде позиции ввода с текстовым курсором, прямоугольным полем ввода, либо другим курсором или визуальным указанием позиции, в которой пользователь может выполнить ввод данных с клавиатуры.

В некоторых случаях выбор может быть выполнен косвенно, посредством использования клавиш навигации. Когда пользователь нажимает клавишу навигации, фокус ввода перемещается в соответствующую позицию (определяемую клавишей) и идентифицирует объект, находящийся в этой позиции.

В некоторых случаях более удобно не только переместить фокус ввода, но и потребовать от пользователя сделать явный выбор с помощью **клавиши выбора**. Рекомендуемая клавиша выбора — <Spacebar> (пробел), если это назначение не противоречит текущей ситуации (в этом случае вы можете использовать сочетание клавиш <CTRL>+<Spacebar>, либо определить другую клавишу, которая лучше подходит в данной ситуации). Иногда клавиша выбора может также использоваться для изменения состояния выбранного объекта.

Непрерывный выбор

При работе с текстовой информацией пользователь перемещает курсор на желаемую позицию, используя клавиши навигации. Зафиксируйте эту позицию в качестве *якоря*. Когда пользователь нажимает клавишу <Shift> одновременно с любой клавишей навигации (или комбинацией клавиш навигации, например, <Ctrl>+<End>), зафиксируйте соответствующую позицию как активную границу области выбора; все символы, расположенные между якорем и этой позицией, включаются в область выбора. Если пользователь после этого нажимает другую клавишу навигации, отмените выбор и переместите границу области в позицию, определяемую клавишей. Если пользователь нажимает клавиши управления курсором, переместите курсор на границу прежней области выбора.

Вы можете использовать эту технику и при работе с другими типами данных, например, со списками, где объекты логически взаимосвязаны. Тем не менее, в таких ситуациях состояние объектов, включенных в область выбора, зависит от состояния выбора объекта, соответствующего позиции-якорю. Например, если объект в этой позиции выбран, то считаются выбранными также все объекты в области, независимо от их текущего состояния. Если объект в позиции-якоря не выбран, то такое же состояние устанавливается для всей области.

Раздельный выбор

Установка начального выбора выполняется с помощью одной из клавиш навигации или клавиши навигации, модифицированной клавишей <Shift>. Пользователь может затем использовать клавиши навигации, чтобы перейти на новую позицию и впоследствии использовать клавишу выбора, чтобы создать дополнительный выбор.

Создание раздельного выбора требует использования клавиш-модификаторов для реализации режима добавления (например, в виде комбинации <Shift>+F8). В этом режиме фокус ввода перемещается, не влияя на существующие выборы или положение позиции-якоря. Когда пользователь нажимает клавишу выбора, установите состояние выбора для новой позиции и обновите связь позиции-якоря с конечной границей области выбора. Чтобы скорректировать выбор относительно текущей позиции-якоря, пользователь может использовать в любой точке области выбора комбинацию клавиши <Shift> с клавишей навигации.

Когда пользователь вторично нажимает клавишу перехода в режим добавления, обеспечьте выход из этого режима, сохраняя установленную область выбора.

Ускоренный выбор

Двойной щелчок ЛКМ представляет собой ускоренный способ выбора объекта. При работе с текстом этот прием обычно используется для выбора слова (без включения знаков пунктуации).

Вы можете определить дополнительный ускоренный способ выбора для некоторых специфических ситуаций. Например, выбор пользователем заголовка столбца может одновременно обеспечить выбор всего столбца. Поскольку такие средства не могут быть распространены на весь интерфейс, не используйте их в качестве единственного способа указания области выбора.

3.5. ОБЩИЕ ПРАВИЛА ВЗАИМОДЕЙСТВИЯ С ОБЪЕКТАМИ

Существуют различные способы организации интерфейса пользователя для выполнения операций над объектами: прямое манипулирование объектами, выбор команд из меню, посредством диалоговых панелей и, наконец, описание требуемых операций на каком-либо языке программирования. Использование в приложении любого из перечисленных подходов не исключает возможности совмещения его с другой техникой. Например, пользователь может изменять размер окна, либо используя команду *Размер*, либо перемещая границу окна с помощью мыши.

Вместе с тем, целесообразно для каждого типа объектов определить собственное подмножество допустимых операций и способов их применения. При этом и подмножество допустимых операций, и способы их применения могут корректироваться в зависимости от текущей ситуации. В связи с этим операции, которые могут быть выполнены над данным объектом в данный момент времени, называют **контекстными**. От текущей ситуации зависит, как правило, и перечень свойств объекта, которые может просматривать и редактировать пользователь. Например, меню для некоторого объекта может содержать и команды, определенные типом объекта, и команды, обусловленные типом контейнера, содержащего объект.

ОПЕРАЦИИ МНОЖЕСТВЕННОГО ВЫБОРА

В качестве операций, доступных для объектов множественного выбора, следует использовать пересечение наборов операций, относящихся к каждому из участников этого выбора. Таким образом, контекст множественного выбора может расширить либо сократить перечень операций или команд, доступных пользователю.

Кроме того, для множественного выбора может быть переопределен эффект выполнения некоторых операций над каким-либо объектом этого выбора. Например, когда пользователь выбирает несколько графических объектов и применяет к ним команду выравнивания, некоторые из них могут остаться на прежней позиции, «не реагируя» на данную команду.

Следует иметь в виду, что область действия операций, выполняемых над объектами множественного выбора, должна ограничиваться пределами активного окна. Например, если пользователь удаляет выбранное слово в одном окне, это не должно приводить к удалению выбранных фрагментов текста в других окнах (если в этих окнах выбраны не те же самые объекты).

ПРЕДОПРЕДЕЛЕННЫЕ ОПЕРАЦИИ

Объект может иметь *предопределенные операции*. **Предопределенные операции** — это операции, выполняемые над объектом по умолчанию, когда пользователь использует технику ускоренного взаимодействия с данным объектом (в частности, двойной щелчок ЛКМ). Например, двойной щелчок ЛКМ на пиктограмме каталога файлов приводит к открытию окна с содержимым этого каталога; при редактировании текста двойной щелчок ЛКМ обеспечивает выбор слова. Для различных объектов могут использоваться разные предопределенные операции; в приведенном выше примере в качестве предопределенной операции для каталога выполнялась команда *Открыть*, а для текста — *Выбрать*.

Аналогично, когда пользователь перемещает объект в новую позицию с помощью мыши, используя технику **drag-and-drop** («перетаски и оставь»), поведение объекта зависит от установленной для него предопределенной операции. Перетаскивание

объекта в некоторую позицию может быть проинтерпретировано, в частности, и как перемещение, и как копирование, и как связывание с другим объектом. В данном случае выполняемая предопределенная операция зависит от позиции, куда был перемещен объект.

Применение предопределенных операций для техники ускоренного взаимодействия с объектами обеспечивает большую эффективность интерфейса с точки зрения повышения скорости работы, что является важным фактором для более опытных пользователей. Тем не менее, поскольку такая техника требует определенных навыков, и не все объекты могут иметь соответствующие предопределенные операции, то рассмотренный подход не должен использоваться в качестве единственного средства взаимодействия пользователя с объектами. Например, даже если для открытия окна каталога используется двойной щелчок ЛКМ на его пиктограмме, всплывающее меню этого объекта должно содержать команду *Открыть*.

ОПЕРАЦИИ ПРОСМОТРА

Ниже приведена краткая характеристика некоторых общих операций, связанных с просмотром объектов (табл. 3.4). Хотя эти операции могут быть доступны не всегда и не для всех объектов, но в случае их использования следует придерживаться указанных соглашений.

Таблица 3.4

Операции просмотра

Операция (команда)	Действие
Открыть (Open)	Открывает первичное окно объекта. Для контейнерных объектов, таких как папки и документы, это окно отображает содержимое объекта
Заккрыть (Close)	Закрывает окно
Свойства (Properties)	Отображает свойства объекта в соответствующем окне, как правило, в окне панели свойств
Помощь (Help)	Отображает окно с контекстной справочной информацией об объекте

Замечание.

В приведенной выше таблице использовано понятие *первичного* окна. Этот термин введен для обозначения окон, именуемых в англоязычной литературе *primary window*. Окна такого типа обычно используются для представления содержимого объектов или в качестве основной рабочей области приложения. В русскоязычных (в том числе переводных) изданиях они либо вообще не имеют специального обозначения (просто *окна*), либо именуются *главными* окнами. Предлагаемый нами вариант кажется нам более предпочтительным по следующим причинам. Во-первых, из-за указанного выше предназначения таких окон: если термин «Главное окно приложения» является во многих случаях достаточно корректным, то про «Главное окно объекта» этого сказать нельзя. Во-вторых, приложение может иметь несколько первичных окон; среди них можно назначить «главное», но как тогда называть остальные? В-третьих, предложенный термин лучше согласуется с весьма устоявшимся термином *вторичное окно*.

Когда пользователь открывает новое окно, оно должно отображаться поверх других окон того же уровня и устанавливаться в активное состояние (как правило, все первичные окна относятся к одному уровню). Дополнительные, или *вторичные* окна, относящиеся к данному приложению, также должны отображаться поверх других вторичных окон того же уровня.

Если пользователь, запустив приложение, успел вернуться к работе с другим окном прежде, чем открылось новое окно, то при открытии нового окна оно не должно «заслонить» активное окно. Например, если пользователь, открыв окно А, затем открывает окно В, то окно В появляется поверх окна А. Если же пользователь щелкнет мышью в окне А прежде, чем откроется окно В, то окно В появляется «за» окном А.

Повторное выполнение команды, с помощью которой было открыто окно, должно активизировать существующее окно вместо открытия другого экземпляра окна. Например, если пользователь выбирает команду *Свойства* для объекта, панель свойств которого уже открыта, активизируется существующая панель, а не открывается второе аналогичное окно.

Необходимо иметь в виду, что приведенное выше правило относится к локальному Рабочему столу пользователя. Если же два пользователя, работающие в сети, открывают окно для одного и того же сетевого объекта, то каждый из них может видеть окно этого объекта на своем Рабочем столе.

Заккрытие окна не обязательно означает завершение процессов, связанных с объектом, представленном в окне. Например, закрытие окна принтера не отменяет печать документов, ожидающих своей очереди. Выход из приложения всегда приводит к закрытию всех его окон, но закрытие какого-либо окна не обязательно приводит к выходу из приложения.

Именно поэтому во вторичных окнах не рекомендуется использовать слово «Заккрыть» для обозначения кнопок, связанных с закрытием окна (но не приложения!);

более подходящими в этом случае являются надписи «Применить», «ОК», «Отменить» и т.п. Не рекомендуется также использовать команду *Заккрыть* в качестве эквивалента команды *Отменить*. Тем не менее, побочный эффект закрытия окна с помощью команды *Заккрыть* зависит от текущей ситуации.

Если в окне имеются внесенные изменения, которые еще не переданы приложению, и пользователь выбирает команду *Заккрыть*, то эти изменения будут потеряны;

чтобы предотвратить такую ситуацию, следует вывести на экран сообщение с просьбой уточнить, хочет ли пользователь применить или отвергнуть изменения, либо вообще отложить выполнение команды *Заккрыть*.

Для быстрого выполнения команд просмотра могут быть определены клавиши-акселераторы; их рекомендуемое назначение приведено ниже (табл.3.5).

Таблица 3.5.

Назначение клавиш-акселераторов

Клавиша-акселератор (комбинация клавиш)	Действие
CTRL + O	Открывает первичное окно для выбранного объекта
ALT + F4	Закрывает окно
F1	Отображает окно с контекстной справочной информацией
SHIFT + F1	Устанавливает режим контекстно-зависимой помощи

Двойной щелчок ЛКМ	Выполняет предопределенную команду или команду Ввод
ALT + двойной щелчок ЛКМ	Открывает панель свойств объекта

ОПЕРАЦИИ РЕДАКТИРОВАНИЯ

Редактирование предполагает изменение (дополнение, удаление, замену) одного или нескольких свойств объекта либо его структуры. Другими словами, не всякое изменение может трактоваться как редактирование объекта. Например, изменение формы представления документа на экране (в частности, увеличение масштаба), не затрагивающее его содержания, не является редактированием. Следующие разделы раскрывают некоторую общую технику взаимодействия пользователя с приложением при редактировании объектов.

ТРАНЗАКЦИИ

Транзакция представляет собой единичное действие по изменению объекта. Степень детализации такого изменения может быть различной, а его результат может определяться выполнением как одной, так и нескольких совмещенных операций. Выполняемые пользователем транзакции (допустимые) должны немедленно приводить к изменению объекта, и пользователь должен видеть их результат; если же транзакция недопустима для данного объекта, необходимо сразу же сообщить об этом пользователю. Другими словами, в любом случае при выполнении транзакции должна осуществляться обратная связь между приложением и пользователем. Кроме того, пользователь должен иметь возможность отменить результат транзакции (внесенные изменения), восстановив предыдущее состояние объекта. Если пользователь закрывает окно, в котором имеются изменения, следует спросить у него, нужно ли их сохранить.

Для сохранения изменений на файловом уровне рекомендуется использовать следующие команды (табл. 3.6).

Таблица 3.6.

Команды сохранения изменений на уровне файла

Команда	Функция
Сохранить (Save)	Сохраняет все временные изменения на диске и начинает новый сеанс редактирования
Сохранить Как (Save as)	Сохраняет файл (со всеми временными изменениями) под другим именем и начинает новый сеанс редактирования
Заккрыть (Close)	Подсказывает пользователю, что следует сохранить внесенные изменения. Если пользователь их подтверждает, временные изменения сохраняются и окно закрывается

Команду *Сохранить* следует использовать в тех случаях, когда внесенные изменения относятся к файлу в целом, например, ко всему документу, и могут быть реализованы одновременно.

Если же выполненная транзакция относится к отдельной записи в файле, то применение команды *Сохранить* не целесообразно. Для управления выполнением транзакций внутри файла рекомендуется использовать команды, перечисленные ниже (табл. 3.7).

Таблица 3.7.

Команды сохранения изменений на уровне транзакций

Команда	Функция
Повторить (Repeat)	Дублирует последнюю транзакцию
Отменить (Undo)	Отменяет результат последней (или какой-то определенной) транзакции
Восстановить (Redo)	Восстанавливает результат последней (или какой-то конкретной) отмененной транзакции
ОК	Реализует внесенные изменения и закрывает окно
Применить (Apply)	Реализует внесенные изменения, но не закрывает окно
Отменить (Cancel)	Отменяет внесенные изменения и закрывает окно

Рекомендуемыми командами для управления процессами являются следующие (табл. 3.8).

Хотя для остановки процесса может быть использована команда *Отменить*, необходимо учитывать, что она не только останавливает процесс, но и возвращает его в исходное состояние.

Таблица 3.8.

Команды управления процессами

Команда	Функция
Пауза (Pause)	Приостанавливает процесс
Продолжить (Resume)	Возобновляет приостановленный процесс
Останов (Stop)	Останавливает процесс

ПРОСМОТР И РЕДАКТИРОВАНИЕ СВОЙСТВ ОБЪЕКТОВ

Определение и согласование свойств компонентов приложения являются ключевыми вопросами при реализации DCD-технологии. Для изменения или переопределения свойств конкретного объекта (или объектов) используются такие команды, как *Свойства*, *Сведения*, *Общие сведения* и *Формат*. Команда *Свойства* является общей командой для доступа к свойствам объекта; когда пользователь выбирает эту команду, следует отобразить на экране вторичное окно, содержащее перечень и текущие значения свойств объекта (*Панель свойств*).

Пользователь должен располагать средствами прямого доступа к свойствам визуальных или легко идентифицируемых объектов, таких как фрагмент текста, ячейка таблицы или рисунок. Сложнее определить способ доступа к свойствам менее

«осязаемых» объектов, например, параграфа. В некоторых случаях может быть реализован косвенный доступ к свойствам таких объектов. Например, по запросу свойств фрагмента текста может быть также предоставлен доступ к свойствам параграфа, содержащего данный фрагмент.

Другой вариант предоставления доступа к таким объектам основан на создании визуального (графического) представление объекта. Например, свойства страницы могут быть доступны через графический образ или другое представление страницы в специальной области (например, в строке состояния) окна.

Возможны и другие подходы к реализации косвенного доступа, например, включение соответствующей команды во всплывающее меню связанного объекта. В частности, всплывающее меню текстового выбора могло бы включать пункт *Свойства параграфа*. В качестве средства доступа к свойствам иерархически связанных объектов может быть создано каскадное меню, каждый уровень которого обеспечивал бы выполнение команды *Свойства* для объекта соответствующего уровня.

Команда *Свойства* не является единственным средством предоставления доступа к свойствам объекта. Например, в окне каталога отображаются некоторые свойства входящих в него файлов (размер, тип, дата последнего изменения и т.д.). Кроме того, для отображения свойств выбранных объектов может использоваться панель инструментов.

ЗОНЫ УПРАВЛЕНИЯ

Графические объекты (окна, рисунки, пиктограммы) могут иметь специальные зоны управления (*handles*), с помощью которых для этих объектов реализуется техника ускоренного взаимодействия. В частности, зоны управления могут использоваться для ускоренного выполнения таких операций, как перемещение, масштабирование, форматирование и автозаполнение. Тип зоны управления зависит от типа объекта. Например, полоса заголовка окна выступает в качестве зоны управления при перемещении окон; границы окна используются в качестве зоны управления при изменении его размеров. Для пиктограммы зоной управления является непосредственно ее изображение, т.е. вся пиктограмма. Для рисунков наиболее общей формой зоны управления является прямоугольный контур, отображаемый вокруг объекта, когда он выбран (рис. 3.6). При этом стороны прямоугольника обычно используются для ускоренного выполнения операций перемещения, а маркеры, расположенные по углам контура — для ускоренного масштабирования рисунка.

Рис. 3.6. Зоны управления графического объекта

РЕДАКТИРОВАНИЕ ТЕКСТА

Редактирование текста требует, чтобы фокус ввода был установлен в той позиции, которая подлежит изменению. Если редактирование выполняется с помощью мыши, фокус ввода всегда совпадает с положением указателя. При использовании клавиатуры положение фокуса ввода определяется действием клавиши навигации, которая была нажата последней. В любом случае визуальным признаком того, что фокус ввода установлен в текстовой области, является присутствие **текстового курсора**, или **точки вставки**.

Вставка текста

Вставка текста предполагает установку пользователем курсора в соответствующей позиции и ввод символов. После каждого набранного символа ваше приложение должно перемещать курсор на один символ вправо (или влево; в зависимости от языка).

Если текстовая область поддерживает ввод нескольких строк, должен быть реализован автоматический перенос текста на следующую строку, когда длина текущей строки превышает ширину текстовой области.

Режим замены

Замена является дополнительным режимом ввода текста, который отличается от вставки только тем, что вводимые символы заменяют существующие.

В качестве визуального признака режима замены рекомендуется использовать курсор специального типа — так называемый **блочный курсор**, который отображается в текущей символьной позиции; это облегчает пользователю идентификацию того символа, который будет заменен (рис. 3.7). Для перехода в режим замены используется клавиша <Insert>.

Внесение изменений

Рис. 3.7. Блочный курсор

Удаление текста

Удаление текста выполняется с помощью клавиш <Delete> и <Backspace>. Клавиша <Delete> удаляет символ справа от курсора, а клавиша <Backspace> удаляет символ слева. В обоих случаях текст сдвигается в направлении удаления, заполняя образовавшийся промежуток (это иногда называют **автослиянием** текста). Удаленный текст не остается в буфере обмена. В связи с этим целесообразно предусмотреть в приложении возможность, по крайней мере, одноуровневой отмены операции удаления.

Если выбран фрагмент текста, принцип действия клавиш <Delete> и <Backspace> остается таким же, как и при удалении одного символа. Если же имеется выбранный фрагмент текста, и пользователь вводит новый текст непосредственно с клавиатуры или с помощью команды *Вставить*, то выбранный фрагмент автоматически удаляется (точнее, заменяется новым текстом).

3.6. ОПЕРАЦИИ ПЕРЕСЫЛКИ И СОЗДАНИЯ ОБЪЕКТОВ

3.6.1. ОПЕРАЦИИ ПЕРЕСЫЛКИ

К операциям пересылки относятся операции перемещения копирования и связывания объектов, а также их производные. Например, печать объекта является формой операции пересылки, поскольку она может быть реализована как копирование объекта на принтер.

Для выполнения любой операции пересылки должны быть указаны три параметра: пересылаемый объект, приемник (получатель) и способ пересылки (т.е. тип выполняемой операции). Эти параметры могут быть определены как явно, так и косвенно, в зависимости от используемой техники взаимодействия.

Тип операции пересылки определяется типом приемника. Поскольку пересылка может интерпретироваться по-разному, в некоторых случаях целесообразно назначить одну из операций в качестве предопределенной (выполняемой по умолчанию), а также ограничить перечень других допустимых операций пересылки, исходя из характеристик источника и приемника. Например, попытка переслать объект в контейнер может закончиться одним из следующих исходов:

- Отклонением объекта
- Приемом объекта
- Включением некоторой части (подмножества) объекта или преобразованием его формата (например, включение только содержания объекта без его внешнего представления).

Большинство действий, связанных с пересылкой объектов, основаны на использовании одной из следующих трех операций (табл. 3.9).

Существуют два метода пересылки объектов: на основе команд и метод прямого манипулирования.

ПЕРЕСЫЛКА НА ОСНОВЕ КОМАНД

Для пересылки объектов используются команды *Вырезать (Cut)*, *Копировать (Copy)* и *Вставить (Paste)*. Эти команды обычно включаются в выпадающее меню *Правка (Edit)* и во всплывающее меню выбранного объекта. Соответствующие им кнопки могут быть вынесены на панель инструментов первичного окна приложения.

Чтобы переслать объект, пользователю необходимо:

1. Выбрать объект, подлежащий пересылке.
2. Выбрать команду *Вырезать* или *Копировать*.
3. Указать позицию вставки.
4. Выбрать команду *Вставить*.

Таблица 3.9.

Операции пересылки объектов

Операция	Описание
Переместить (Move)	Перемещает выбранный объект; поскольку перемещение не изменяет «подлинности» объекта, эту операцию не следует отождествлять с копированием или замещением оригинала
Копировать (Copy)	Создает копию объекта. Результирующий объект независим по отношению к своему оригиналу. Дублирование не всегда обеспечивает полную идентичность копии; некоторые ее свойства могут отличаться от свойств оригинала. Например, копирование объекта может привести к изменению имени или даты создания. Если имеются ограничения на копирование некоторых элементов объекта, то могут быть скопированы только те элементы, для которых ограничения отсутствуют
Связать (Link)	Создает связь между двумя объектами. Результатом обычно является объект, который обеспечивает доступ к оригиналу (ярлык)

Команда *Вырезать* удаляет выбранный объект и помещает его (или ссылку на него) в буфер обмена. Команда *Копировать* создает копию выбранного объекта (или ссылку на него) и помещает ее в буфер обмена.

При использовании команды *Вставить* по умолчанию предполагается, что пересылаемый объект сохранит все свойства оригинала, однако может быть использована и альтернативная форма этой команды для реализации других видов пересылки:

Вставить [*<имя типа>*] как [*<имя типа>* | в *<имя объекта>*].

Например, возможен такой вариант команды:

Вставить [*Ячейку*] как [*Слово*] (предполагается, что *Ячейка* и *Слово* являются конвертируемыми типами).

Ниже приведены общие форматы команды *Вставить* (табл. 3.10).

Таблица 3.10.

Форматы команды *Вставить*

Формат команды	Функция
Вставить (Paste)	Вставляет объект, содержащийся в буфере обмена, сохраняя все свойства его оригинала
Вставить [<i>имя типа</i>] (Paste [<i>type name</i>])	Вставляет объект, содержащийся в буфере обмена, как вложенный объект OLE; такой объект может быть активизирован непосредственно в позиции вставки
Вставить [<i>имя типа</i>] как пиктограмму (Paste [<i>type name</i>] as Icon)	Вставляет объект, содержащийся в буфере обмена, как вложенный объект OLE; объект отображается как пиктограмма
Вставить Ссылку (Paste Link)	Создает ссылку на объект, который был скопирован в буфер обмена; в точку вставки помещается копия содержимого объекта, обладающая всеми его свойствами, однако при этом сохраняется связь с оригиналом, так что любое его изменение приводит к изменению копии
Вставить Ссылку на [<i>имя объекта</i>] (Paste Link to [<i>Object name</i>])	Помещает в точку вставки изображение объекта, который был скопирован в буфер обмена, используя технологию OLE; любые изменения исходного объекта отражаются на его копии
Вставить ярлык (Paste Shortcut)	Помещает в точку вставки ярлык объекта, который был скопирован в буфер обмена, используя технологию OLE; любые изменения исходного объекта отражаются на его копии
Специальная Вставка (Paste Special)	Отображает на экране специальное диалоговое окно, позволяющее пользователю выбрать способ вставки объекта, содержащегося в буфере обмена

Ниже приведены рисунки, поясняющие использование форматов команды *Вставить*, описанных в пунктах 2...5 табл. 3.10.

Рис. 3.8. Вставка объекта как вложенного объекта OLE

Рис. 3.9. Вставка объекта как вложенного объекта OLE;
объект будет отображен в виде пиктограммы

Рис. 3.10. Создается ссылка на объект, который был скопирован в буфер обмена;
в точку вставки помещается копия содержимого объекта

Рис. 3.11. Создается ссылка на объект, который был скопирован в буфер обмена;
в точку вставки помещается пиктограмма объекта

При выборе формата команды *Вставить* необходимо, в первую очередь, учитывать свойства приемника; для повышения гибкости работы пользователя ему может быть предоставлена возможность корректировки параметров команды в зависимости от особенностей пересылаемого объекта и сущности приемника.

Практика показывает, что для большинства приложений оказывается достаточным совместное использование стандартной команды *Вставить* и *Специальной вставки*.

Поскольку выполнение операций пересылки может привести к различным побочным эффектам, ниже описаны некоторые общие сценарии поведения приложения при их реализации.

- Если пользователь выполняет пересылку в приемник, для которого необходимо учитывать конкретную позицию вставки, следует заменить выбранный объект в приемнике переданными данными. Например, если выполняется вставка в текст (где положение выбранного фрагмента соответствует позиции вставки), пересылаемый объект заменяет текущий выбор. Если же в тексте-приемнике выбор не был сделан, точка вставки определяется текущим положением курсора.

- Для приемников, где нет явной точки вставки, следует поместить вставляемый объект в точку, определяемую текущим контекстом, и отобразить его как выбранный. В качестве контекстной информации можно, например, учитывать позицию указателя мыши, либо признак, по которому упорядочены в приемнике имеющиеся объекты (скажем, по алфавиту).

- Если новый объект автоматически связывается с текущим выбором в приемнике (например, табличные данные и отображающий их график), целесообразно добавить новый объект к текущему выбору.

Выполнение команды *Вставить* не должно влиять на содержимое буфера обмена. Это позволяет неоднократно повторно использовать данные, находящиеся в буфере. Вместе с тем, необходимо помнить, что последующее выполнение команд *Копировать* или *Вырезать* приводит к замене содержимого буфера обмена.

МЕТОД ПРЯМОГО МАНИПУЛИРОВАНИЯ

Командный метод эффективен в тех случаях, когда операция пересылки требует от пользователя соответствующего управления источником и приемником. Тем не менее, при выполнении многих операций пересылки прямое манипулирование оказывается более естественным и быстрым методом. При использовании прямого манипулирования пользователь выбирает и перетаскивает объект в желаемую позицию, но поскольку этот метод требует определенных моторных навыков, следует избегать его использования в качестве единственного метода пересылки. В наиболее развитых интерфейсах поддерживаются оба метода: командный — для выполнения основных операций пересылки, а прямое манипулирование — как ускоренная техника работы пользователя.

Прямое манипулирование может применяться для любого визуального объекта. При этом как пересылаемый объект, так и приемник (например, окно или пиктограмма) не обязательно должны быть к настоящему времени активны. Например, пользователь может поместить объект в неактивное окно и в результате выполнения этой операции оно автоматически будет активизировано. Если неактивный объект не допускает пересылку с помощью прямого манипулирования, пользователь должен быть извещен об этом посредством соответствующей обратной связи.

Каким образом переданный объект интегрируется и отображается в приемнике, определяется контекстом приемника. Ранее уже отмечалось, что результатом пересылки может быть как точная копия оригинала, так и отдельные его компоненты или свойства, либо преобразованный объект. Например, приложение может содержать средства для копирования свойств объектов конкретного типа.

Стандартная реализация техники *drag-and-drop*

Стандартная реализация техники *drag-and-drop* основана на использовании левой кнопки мыши. Как именно будет проинтерпретировано приложением перетаскивание объекта, зависит от свойств приемника и от того, какая операция пересылки используется в качестве predetermined. Как и в случае командного метода, результат операции определяется также свойствами перемещаемого объекта (в частности, перечнем разрешенных форматов объекта). Не рекомендуется использовать в качестве predetermined операции такую, которая может привести к удалению или необратимому изменению объекта. Если же необходимость применения одной из таких операций обусловлена текущей ситуацией, следует вывести на экран сообщение пользователю с просьбой подтвердить намерения.

Используя технику *drag-and-drop*, пользователь может непосредственно пересылать объекты, как между окнами приложения, так и между окнами системных ресурсов, такими, например, как папки и принтеры.

Чаще всего в качестве predetermined операции для *drag-and-drop* используется операция *Переместить* (*Move*), но вместо нее может быть определена любая другая операция пересылки, например *Копировать* или *Связать*, или даже специализированная операция пересылки, такая, например, как *Копировать Свойства*.

Нестандартная реализация техники *drag-and-drop*

В основе нестандартной реализации техники *drag-and-drop* лежит использование правой кнопки мыши. Особенность данного подхода заключается в следующем. Когда пользователь щелкает ПКМ на объекте-приемнике (или на позиции, куда пересылается объект), вместо выполнения predetermined операции открывается всплывающее меню для объекта-приемника (на рис. 3.12 — для новой позиции). Это меню содержит набор команд пересылки, разрешенных для данной позиции, но с учетом особенностей объекта, подлежащего пересылке.

Рис. 3.12. Пересылка объекта при нестандартной реализации техники *drag-and-drop*

Тот факт, что команды всплывающего меню относятся к приемнику, а не к пересылаемому объекту, должен учитываться и в названиях этих команд. В частности, если пересылается содержимое объекта (данные), такими командами могут быть: *Переместить сюда* (*Move Here*), *Копировать сюда* (*Copy Here*). Если же приемник

поддерживает пересылку только отдельных компонентов объекта, или требует его преобразования, названия команд должны отражать эту особенность приемника (например, *Копировать свойства сюда*).

Общий формат нестандартных команд пересылки выглядит следующим образом:

[имя команды] [имя типа \ имя объекта] Here как [имя типа].

Ниже приведены пояснения к применению нестандартных команд пересылки (табл. 3.11).

Одна из перечисленных выше команд может быть использована в качестве предопределенной. Это должна быть команда, которая соответствует эффекту перетаскивания объекта при нажатой левой кнопке мыши.

Таблица 3.11.

Нестандартные команды пересылки

Команда	Функция
Переместить сюда (Move Here)	Пересылает в указанную позицию содержимое выбранного объекта (как данные)
Копировать сюда (Copy Here)	Создает в указанной позиции копию содержимого выбранного объекта
Создать связь с приемником (Link Here)	Создает канал связи между выбранным объектом и позицией-приемником; содержимое выбранного объекта добавляется (возможно, после преобразования) к содержимому приемника, но сохраняется связь с источником, так что производимые в нем изменения отражаются в позиции-приемнике
Переместить [имя типа] сюда (Move [type name] Here) или Копировать [имя типа] сюда (Copy [type name] Here)	Перемещает или копирует выбранный объект как объект OLE, который отображается в своем исходном представлении и может быть активизирован непосредственно в позиции-приемнике
Создать для [имя типа] Связь с приемником (Link [type name] Here)	Создает связанный объект OLE, отображаемый в приемнике как изображение выбранного объекта. Копия связывается с исходным объектом таким образом, что любые изменения в исходном объекте отражаются в позиции-приемнике
Переместить [имя типа] сюда как пиктограмму (Move [type name] Here) as Icon или Копировать [имя типа] сюда как пиктограмму (Copy [type name] Here) as Icon	Перемещает или копирует выбранный объект как объект OLE в указанную позицию и отображает его как пиктограмму
Создать здесь ярлык (Create Shortcut Here)	Создает объект OLE, связанный с выбранным объектом; новый объект отображается как ярлык; любые изменения исходного объекта отражаются в позиции-приемнике

Если пользователь, начав перетаскивать объект, возвращает его в исходную позицию, это следует интерпретировать как отмену операции пересылки. Тот же эффект

обеспечивает нажатие пользователем клавиши <Esc> в течение выполнения операции пересылки. Кроме того, целесообразно включить команду *Отменить* во всплывающее меню, используемое при нестандартной технике пересылки.

Поскольку при выполнении операций пересылки совмещается выбор объекта и его перемещение, необходимо установить соглашение, позволяющее пользователю различать эти операции. Используемое вашим приложением соглашение зависит от свойств объектов приложения; в некоторых случаях могут быть предусмотрены различные средства взаимодействия пользователя с объектом при выполнении выбора и пересылки. Наиболее общий подход основан на том, что признаком выполняемой операции является исходная позиция указателя мыши. Если указатель расположен в пределах существующего выбора, то к выбранному объекту применяется операция пересылки; если же в момент нажатия ЛКМ указатель находится вне области выбора, то его последующее перемещение интерпретируется как операция выбора.

ОСОБЕННОСТИ РЕАЛИЗАЦИИ ПРОКРУТКИ ПРИ ВЫПОЛНЕНИИ ОПЕРАЦИЙ ПЕРЕСЫЛКИ

Когда пользователь перетаскивает объект внутри какой-либо перемещаемой области (такой как окно или список) в другую позицию, это может привести к тому, что объект окажется за пределами видимости. В этом случае целесообразно использовать возможность *автоматической прокрутки* области (которая известна также как *автоскроллинг*).

Вместе с тем, при выполнении операций пересылки путем прямого манипулирования пользователю необходимо «вытащить» объект из одного окна и поместить в другое. При этом автоскроллинг не только не нужен, он просто мешает пользователю выполнить требуемую операцию.

Чтобы различать две эти ситуации, можно анализировать скорость перемещения объекта. Например, если пользователь тащит объект медленно к краю перемещаемой области, используется автоскроллинг; если же объект перемещается быстро, выполняется операция пересылки.

Для реализации указанного подхода требуется запомнить позицию указателя в начальный момент перемещения мыши и затем регистрировать его положение через равные промежутки времени (рекомендуемый интервал — 100 миллисекунд). Если же приложение поддерживает пересылку на основе OLE, то замеры через равные промежутки времени будут некорректны. В этом случае следует запоминать каждое новое положение указателя в специальном массиве, достаточно большом, чтобы хранить три последних значения, заменяя самое старое из них текущим. При вычислении скорости указателя используются координаты, по крайней мере, двух последних его позиций.

Чтобы рассчитать скорость перемещения, необходимо вычислить расстояние между соседними позициями указателя и разделить его на величину интервала времени между моментами регистрации. Расстояние вычисляется как модуль разности между координатами x и y двух соседних позиций:

$$R=(abs(x1-x2)+abs(y1-y2))$$

Полученное значение следует умножить на 1024 и разделить на интервал времени. Множитель 1024 предотвращает потерю точности, вызванную целочисленным делением.

Читатель, знакомый со школьным курсом геометрии, вероятно может возразить, что истинное Декартово расстояние определяется как квадратный корень суммы

квадратов разностей координат: $\sqrt{(x1 - x2)^2 + (y1 - y2)^2}$. Однако дополнительные затраты времени на вычисления по этой формуле соизмеримы с временем перемещения указателя в соседнюю позицию, что и заставляет использовать вместо нее приведенное выше выражение.

Следует также определить **горячую зону** автоскроллинга вдоль границ перемещаемой области и величину задержки прокрутки. Как правило, ширина горячей зоны устанавливается равной удвоенной ширине вертикальной полосы прокрутки или удвоенной высоте горизонтальной полосы прокрутки.

Итак, автоматическая прокрутка должна выполняться при одновременном соблюдении трех условий (рис. 3.13):

- в процессе перемещения объекта указатель оказался в горячей зоне окна;
- скорость перемещения объекта ниже определенной (пороговой) скорости;
- видимая область окна может быть перемещена в соответствующем направлении.

Рис. 3.13. Автоматическая прокрутка при перемещении объекта

Рекомендуемая пороговая скорость — 20 пикселей в секунду.

Величина прокрутки зависит от типа перемещаемой информации и допустимого расстояния прокрутки. Например, при работе с текстом вертикальная прокрутка обычно выполняется построчно. Целесообразно также согласовать дискретность автоскроллинга с дискретностью перемещения, которая предусмотрена для полос прокрутки.

Чтобы обеспечить непрерывную прокрутку, необходимо выполнить следующие действия. Во-первых, выбрать частоту прокрутки, например, четыре строки в секунду. Во-вторых, после начала автоскроллинга следует установить таймер на определенный интервал (например, на 100 миллисекунд). Когда установленный интервал истечет, определите величину прокрутки за это время. Если интенсивность прокрутки превышает заданную частоту, переместите видимую область еще на один шаг. Если нет, восстановите таймер и повторите описанную последовательность действий при следующей инициализации автоскроллинга.

ОБРАТНАЯ СВЯЗЬ ПРИ ВЫПОЛНЕНИИ ОПЕРАЦИЙ ПЕРЕСЫЛКИ

Поскольку пересылка объектов является одним из наиболее распространенных действий пользователя, наличие соответствующей обратной связи является важным показателем качества интерфейса создаваемого приложения.

Реализация обратной связи при пересылке на основе команд

Если при пересылке объекта пользователь выбирает команду *Вырезать*, следует визуально удалить выбранный объект. Если в силу каких-либо причин это нецелесообразно, можно отобразить выбранный объект специальным образом, чтобы сообщить пользователю, что команда *Вырезать* выполнена, но пересылка объекта еще не закончена.

Необходимо также восстанавливать визуальное состояние объекта, если пользователь выбирает команду *Вырезать* или *Копировать* для другого объекта прежде, чем выполнить команду *Вставить* для первоначально выбранного объекта.

Команда *Копировать* не требует специальной обратной связи. Выполнение команды *Вставить* также не требует никакой дополнительной обратной связи, кроме как отображение переданного объекта в новой позиции. Тем не менее, если при выполнении команды *Вырезать* образ объекта в исходной позиции был не удален, а заменен альтернативным представлением, необходимо его теперь «окончательно» удалить.

Реализация обратной связи при прямом манипулировании

При перемещении объекта в новую позицию с помощью мыши следует обеспечить пользователя визуальным представлением выполняемой операции па

всем ее протяжении. При этом следует обратить особое внимание на следующие моменты:

- Объект должен отображаться как выбранный все время, пока на нем установлен фокус ввода. Чтобы указать, что объект находится в состоянии пересылки, для его изображения могут быть использованы соответствующие визуальные признаки. Например, для визуализации операции перемещения может быть использовано системное контурное изображение объекта. Визуальный образ объекта в исходной позиции следует сохранять до тех пор, пока пользователь не завершит операцию пересылки.

- Изображение объекта, перемещаемое вместе с указателем, должно быть «прозрачным» и не заслонять позицию вставки, как показано на рис. 3.14.

Рис. 3.14. Возможные способы представления перемещаемых объектов

- Образ объекта в исходной позиции должен сохраняться неизменным до окончания операции пересылки; вместе с тем, его перемещаемое изображение может изменяться в зависимости от типа выполняемой операции. Например, если объект будет вставлен как пиктограмма, то при перемещении указателя на позицию вставки образ объекта должен быть заменен его пиктограммой. Если объект будет включен как часть содержимого приемника, то представление объекта должно отразить это. Например, если объект, вставляемый в документ, будет включен как таблица, он может быть представлен в виде прозрачного силуэта таблицы. И наоборот, если исходный объект — Таблица, но при вставке преобразуется в текст, отобразите его как условное изображение текста, например, как прозрачный силуэт нескольких первых слов.

- Форма указателя также должна соответствовать типу объекта- (позиции-) приемника. Например, если перетаскиваемый объект должен быть вставлен в текст (непосредственно между символами), отобразите указатель в виде стандартного текстового курсора, используемого при редактировании текста.

Кроме того, целесообразно дополнить изображение указателя специальным символом, поясняющим смысл выполняемой операции пересылки; этот символ отображается возле правого нижнего угла указателя, как показано на рис. 3.15. Символ «плюс» (+) используется при выполнении операций копирования, а изображение стрелки — для операций создания ссылок.

Рис. 3.15. Изображение указателя:

а) перемещение б) копирование в) создание ссылки

- Используйте визуальную обратную связь, чтобы указать доступность потенциальных приемников. Она может быть реализована на основе стандартных средств выделения выбора, а также дополнительных средств, в том числе анимации. Если же приемник недоступен для пересылаемого объекта, на это можно указать с помощью специального указателя, напоминающего по форме запрещающий дорожный знак (рис. 3.16.).

Рис. 3.16. Указатель, информирующий о недоступности приемника

СПЕЦИАЛЬНЫЕ КОМАНДЫ ПЕРЕСЫЛКИ

При выполнении некоторых заданий определенная комбинация операций пересылки может использоваться настолько часто, что ее удобнее заменить соответствующей дополнительной специализированной командой. Примером такой комбинации является дублирование существующего объекта, выполняемое с помощью двух операций — *Копировать* и *Вставить*. Вместо них может быть использована специальная команда *Дублировать*. Ниже приведены некоторые наиболее распространенные специализированные команды пересылки (табл. 3.12).

Таблица 3.12.

Специализированные команды пересылки

Команда	Функция
Удалить (Delete)	Удаляет объект из содержащего его контейнера. Если объект — файл, то он пересылается в Корзину
Очистить (Clear)	Удаляет содержимое контейнера
Дублировать (Duplicate)	Создает одну копию выбранного объекта
Печать (Print)	Печатает выбранный объект на принтере, используемом по умолчанию
Отправить (Send To)	Отображает список возможных направлений пересылки выбранного объекта и пересылает его в соответствии с выбором пользователя

На первый взгляд, команды *Удалить* и *Очистить* дают один и тот же эффект. Тем не менее, каждая из них действительно более корректно работает по своему прямому назначению: *Удалить* — при удалении одного (выбранного) объекта, а *Очистить* — при очистке контейнера.

КЛАВИШИ-АКСЕЛЕРАТОРЫ ДЛЯ ОПЕРАЦИЙ ПЕРЕСЫЛКИ

Перечисленные ниже комбинации клавиш обеспечивают ускоренный доступ к операциям пересылки (табл. 3.13).

Таблица 3.13.

Клавиши-акселераторы для операций пересылки

Комбинация клавиш	Функция
CTRL + X	Выполняет команду Вырезать
CTRL + C	Выполняет команду Копировать
CTRL + V	Выполняет команду Вставить
CTRL + <перемещение объекта>	Заменяет предопределенную операцию пересылки операцией Копировать (если приемник ее поддерживает); модификатор может использоваться с любой кнопкой мыши
ESC	Отменяет выполнение операции пересылки (если она еще не закончена)

Из-за широкого применения приведенных комбинаций клавиш для всего интерфейса, не рекомендуется использовать их в других целях.

3.6.2. ОПЕРАЦИИ СОЗДАНИЯ НОВЫХ ОБЪЕКТОВ

Возможность создания новых объектов пользователем предусмотрена в подавляющем большинстве приложений. Хотя каждое из них ориентировано на создание объектов определенного типа, разработчик приложения должен, как правило, воздерживаться от применения для этих целей каких-то специфических средств. Обычно создание нового объекта основано на использовании некоторого предопределенного объекта (шаблона) или спецификации и может быть реализовано несколькими способами, поддерживаемыми стандартными средствами пользовательского интерфейса.

Команда Копировать (Copy)

Получение копии существующего объекта является основным способом создания новых объектов. Объекты-копии могут модифицироваться и обычно используются в качестве прототипов для создания новых объектов, обладающих иными свойствами по сравнению с объектом-оригиналом. Технология создания новых объектов указанным способом определяется соглашениями по использованию операций пересылки, рассмотренными выше.

Команда Создать (New)

Команда *Создать (New)* облегчает создание новых объектов. Она обеспечивает автоматическое создание нового экземпляра объекта определенного типа и может рассматриваться как комбинация команд *Копировать (Copy)* и *Вставить (Paste)*.

Команда Поместить (Insert)

Команда *Поместить (Insert)* работает аналогично команде *Создать (New)*, за исключением того, что она применяется к контейнеру для создания нового объекта (обычно определенного типа) внутри этого контейнера. Используя различные форматы команды *Поместить*, можно предоставить пользователю возможность включать в контейнер объекты различных типов. Если создаваемое приложение поддерживает технологию OLE, с помощью команды *Поместить* может быть реализовано создание широкого класса объектов. Кроме того, объекты, поддерживаемые вашим приложением, могут включаться в файлы данных, созданные другими приложениями OLE.

Использование элементов управления

С целью автоматического создания новых объектов могут использоваться элементы управления. Например, в графических приложениях часто используются кнопки, соответствующие различным инструментальным средствам или способам создания новых объектов, таких как линия определенной формы или

геометрическая фигура. Кнопки могут также использоваться для внедрения объектов по технологии OLE.

Использование шаблонов

Шаблон является специальным объектом, который автоматизирует создание нового объекта. Чтобы пользователю было понятно его предназначение, пиктограмма шаблона должна содержать небольшой рисунок, отражающий тип создаваемых с его помощью объектов, как показано на рис. 3.17.



Рис. 3.17. Пример пиктограммы шаблона

В качестве предопределенной операции для объекта *Шаблон* используется команда *Создать*; она инициирует процесс создания объекта, который может выполняться как полностью автоматически, так и в режиме диалога с пользователем. Вновь созданный объект должен находиться в той же позиции, где находился шаблон. Если текущий контекст не позволяет это сделать, объект следует установить в другой доступной пользователю позиции, например, на Рабочем столе. Размещение нового объекта может также быть указано пользователем в процессе диалога при создании объекта; в этом случае по завершении создания объекта следует вывести на экран сообщение, информирующее пользователя о местоположении созданного объекта.

3.6.3. ОПЕРАЦИИ СВЯЗЫВАНИЯ ОБЪЕКТОВ

В результате выполнения операции связывания объектов создается специальная форма отношения двух объектов — **ссылка**, которая отображает или предоставляет доступ от одного объекта к другому, расположенному либо в том же контейнере, либо в другом, отдельном контейнере. Компонентами этого отношения являются **источник связи (референт)** и **связанный объект**. Перечень допустимых операций и свойств связанного объекта, как правило, не зависит от объекта-источника, и определяется типом контейнера, в котором он находится.

Связь между объектами может быть представлена различными средствами интерфейса. В частности, **связь по данным** реализуется посредством передачи соответствующих величин между двумя объектами, например, между двумя ячейками в таблице или между столбцом таблицы и диаграммой. **Переходы** (называемые также **гиперссылками** — **hyperlinks**) предоставляют навигационный доступ к другому объекту. Этот вид связи можно назвать также **связью по управлению**.

Объект, связанный с источником с помощью механизма OLE, предоставляет доступ ко всем операциям, разрешенным для источника, и, кроме того, обеспечивает визуальное представление связи с источником.

Когда пользователь пересылает связанный объект, следует сохранить как абсолютный, так и относительный маршрут доступа к его источнику связи.

Абсолютный маршрут — это точное описание положения объекта в его иерархии. **Относительный маршрут** является описанием позиции объекта относительно текущего контейнера.

Какой из двух маршрутов будет использоваться, зависит от направления пересылки. Как правило, по умолчанию используется относительный маршрут. Тем не менее, если по одному из маршрутов обеспечить доступ к источнику не удалось, следует использовать альтернативный маршрут. Например, если пользователь копирует связанный объект и источник связи в другую позицию, создаются копии, как связанного объекта, так и источника связи. В этом случае относительным маршрутом для копии связанного объекта является описание расположения копии источника связи, а абсолютным маршрутом — описание позиции исходного источника связи (оригинала). Другими словами, когда пользователь обращается к копии связанного объекта, следует активизировать связь с копией источника связи. Если же это сделать не удастся (например, из-за того, что пользователь удалил копию источника), необходимо использовать абсолютный маршрут, обеспечивающий доступ к исходному источнику связи.

В качестве дополнительного средства настройки интерфейса пользователю может быть предоставлена возможность выбора используемого маршрута (абсолютного или относительного) для связанного объекта. Такой выбор может выполняться с помощью соответствующего элемента управления на панели свойств связанного объекта.

Если пользователь применяет операцию связывания к связанному объекту, должна создаваться ссылка на него, а не на источник. В случае, если связанный объект не может использоваться в качестве источника, то, когда пользователь выбирает такой связанный объект, следует сделать недоступными для него команды связывания.

Способ активизации связанного объекта зависит от типа связи. Например, активизация перехода может выполняться с помощью однократного щелчка ЛКМ. Тем не менее, такой механизм эффективен только для выбора связанных данных или связанного объекта OLE. Если же он используется, чтобы сделать что-то еще, кроме выбора объекта, следует использовать различные формы визуального представления для каждой из допустимых операций. Например, если переход выполняется с помощью кнопки-акселератора, то при выборе кнопки изменяется форма указателя (он отображается в виде руки, как показано на рис. 3.18), а когда пользователь щелкает на ней ЛКМ, то изменяется изображение кнопки и производится переход по ссылке.

Рис. 3.18. Идентификация выполняемой операции с помощью изменения формы указателя

При таком подходе обеспечивается обратная связь с пользователем, который может либо только выбрать ссылку на связанный объект, либо и воспользоваться ею.

4. ОКНА И ПИКТОГРАММЫ

В соответствии с концепциями, положенными в основу графического интерфейса, объекты приложения могут быть визуальными представлены на Рабочем столе либо в виде пиктограмм, либо в виде окон, отображающих содержимое объекта.

Во многих случаях для реализации взаимодействия пользователя с объектами приложения или с приложением в целом оказывается достаточным единственного первичного окна (возможно, дополненного набором вторичных окон).

Однооконная модель приложения облегчает пользователям ассоциативную связь между объектами и их визуальным представлением; кроме того, она существенно упрощает пользователю работу с окнами. При этом технология OLE позволяет в пределах единственного первичного окна создавать сложные структуры данных, объединяющие информацию различных типов.

Объекты некоторых типов (например, устройства) могут даже не требовать создания первичного окна и использовать только вторичное окно для просмотра и редактирования их свойств. Иногда (весьма редко) объект вообще может быть представлен в приложении лишь своей пиктограммой.

Вместе с тем, при выполнении некоторых заданий, однооконная модель не обеспечивает достаточно эффективного управления приложением (или отдельными его объектами); обычно такая ситуация имеет место в тех случаях, когда пользователю требуется работать одновременно с несколькими различными формами представления одних и тех же данных или с несколькими видами взаимосвязанных данных в пределах одного окна. В таких случаях следует продумать возможность использования других моделей приложения, например, на основе многодокументного интерфейса (MDI) или Проекта.

Техника взаимодействия пользователя с приложением существенно зависит от выбранной модели его построения; в данной главе, а также в двух последующих рассмотрены основные особенности различных моделей построения GUI-приложения, учет которых на этапе проектирования позволяет обеспечить соответствие конечного результата исходным целям разработки.

4.1. ПРОЕКТИРОВАНИЕ ПИКТОГРАММ

Все пиктограммы, используемые в приложении, следует разрабатывать как единый набор; при этом должна обеспечиваться их согласованность и друг с другом, и с заданиями пользователя. Если первоначальный вариант не удовлетворяет потенциального пользователя, нельзя жалеть времени на внесение изменений.

Каждая пиктограмма должна быть реализована в трех стандартных форматах: 16x16 пикселей (для 16 цветов), 32x32 пиксела (также для 16 цветов) и 48x48 пикселей (для 256 цветов), как показано на рис. 4.1. Хотя для меньших размеров также может быть использована большая глубина цвета, это требует увеличения памяти для хранения пиктограмм и не может быть реализовано на всех конфигурациях компьютера.

Система автоматически формирует цветовую схему пиктограммы для монохромных конфигураций. Тем не менее, целесообразно заранее оценить качество зрительного восприятия разработанных пиктограмм в монохромном режиме. Если результат окажется неудовлетворительным, следует создать собственные монохромные варианты пиктограмм.

Пиктограммы необходимо разрабатывать не только для исполняемого файла приложения, но также для всех типов файлов данных, поддерживаемых вашим приложением.

При этом пиктограммы для файлов данных (или документов) должны отличаться от пиктограммы приложения. Как правило, они могут содержать некоторый общий элемент, однако основное содержание рисунка пиктограмм файлов данных должно отражать сущность хранимой в них информации.

Все созданные пиктограммы должны быть зарегистрированы в системном реестре. Если какая-либо из пиктограмм не будет зарегистрирована, система автоматически использует вместо нее основную пиктограмму приложения. Тем не менее, вряд ли такая замена будет удачнее того, что мог бы предложить сам разработчик.

В основу рисунка, отображаемого на пиктограмме, должен быть положен образ объекта реального мира, точнее, те его детали, которые действительно необходимы для однозначного восприятия объекта пользователем. Где это возможно, лучше использовать трехмерное изображение и светотень, чтобы лучше передать образы реального мира, как показано на рис. 4.2.

Рис. 4.1. Три формата пиктограммы

Рис. 4.2. Примеры «плоского» и трехмерного изображения объекта

Подводя итог обсуждению стиля создаваемых пиктограмм, можно сказать, что выводимое на них изображение должно вызывать у потенциального пользователя вполне определенную предсказуемую ассоциацию с объектами реального мира.

4.2. ПЕРВИЧНЫЕ ОКНА

Поскольку окна предоставляют доступ к различным видам информации (текстовой, графической), они классифицируются согласно их основному назначению. Взаимодействие с объектами обычно реализуется с помощью так называемого *первичного окна*, в котором происходит первоначальный просмотр и редактирование данных. Кроме того, могут использоваться дополнительные (*вторичные*) окна, которые позволяют устанавливать дополнительные параметры обработки, или обеспечивают доступ к более специфическим деталям взаимодействия с объектами, включенными в первичное окно.

4.2.1. СТРУКТУРА ПЕРВИЧНОГО ОКНА

Типовое первичное окно состоит из *рамки* (или границы), которая определяет размеры окна, и *заголовка окна*, который идентифицирует информацию, представленную в окне. Если объем выводимой информации превышает текущий размер окна, оно дополняется *полосами прокрутки*. Окно может также содержать другие элементы интерфейса (меню, панель инструментов, строку состояния), как показано на рис. 4.3.

Внешний вид рамки окна определяется типом окна. Изменяемое окно имеет четкую границу, которая обеспечивает управление размерами на основе прямого манипулирования. Если окно не может изменять размеры, граница «сливается» с краем окна.

В верхней части окна находится заголовок окна. Он выводится в так называемой *полосе заголовка*, которая занимает всю ширину окна. Полоса заголовка служит также в качестве опорной точки для перемещения окна и для

отображения свернутого окна. Кроме того, она используется в качестве средства доступа к командам управления окном. В частности, щелчок правой кнопкой мыши в полосе заголовка приводит к появлению всплывающего меню для данного окна.

Первичное окно содержит также уменьшенную копию пиктограммы объекта или приложения, к которому оно относится. Она выводится в верхнем левом углу окна (в полосе заголовка). Если окно относится к какой-либо компоненте приложения, которая не создает свои собственные файлы данных, то рекомендуется в качестве такой пиктограммки использовать пиктограмму самого приложения.

Рис. 4.3. Структура первичного окна.

Если приложение обеспечивает работу с различными форматами документов или файлов, и данное окно отображает один из этих файлов, используйте в полосе заголовка пиктограмму, соответствующую его типу (рис. 4.4).

Рис. 4.4. Использование пиктограммы типа файла в полосе заголовка окна

Если приложение использует многодокументный интерфейс (MDI), поместите пиктограмму приложения в полосе заголовка родительского окна, а в полосе заголовка дочернего окна — пиктограмму, которая отражает конкретный тип файла данных приложения (как показано на рис. 4.5).

Рис. 4.5. Полосы заголовков родительского и дочернего окон MDI

Когда пользователь разворачивает дочернее окно, и полоса заголовка обоих окон объединяется, отобразите пиктограмму заголовка дочернего окна в полосе меню родительского окна (рис. 4.6). Если открыто несколько дочерних окон, то в указанной позиции следует отобразить только пиктограмму активного (самого верхнего) дочернего окна.

Рис. 4.6. Полоса заголовка родительского окна при наличии развернутого дочернего окна

Полоса заголовка окна используется также в качестве интерактивного элемента для вызова всплывающего меню окна: когда пользователь щелкает на пиктограмме заголовка окна ПКМ, на экране появляется всплывающее меню для соответствующего объекта. Обычно такое меню содержит типовой набор команд (открыть/закрыть окно, изменить размер окна и т.п.). Рекомендуется также поддерживать использование двойного щелчка ЛКМ на пиктограмме окна в качестве ускоренного способа закрытия окна.

Заголовок окна идентифицирует объект, отображаемый в окне. Он также должен быть согласован с пиктограммой этого объекта, используемой в файловой системе. Например, если пользователь открывает файл данных с названием *Мой_Документ*, то в полосе заголовка окна должна быть помещена пиктограмма для этого типа документа, сопровождаемая именем файла данных. Дополнительно можно также указать имя используемого приложения.

Для приложения, использующего MDI, его имя отображается в родительском окне, а в дочернем окне — имя данных (рис. 4.6).

Если пользователь непосредственно открывает приложение, которое отображает новый *файл* данных, укажите его имя в заголовке окна, даже если пользователь еще не сохранял файл. Используйте для него какое-либо общее имя, отражающее тип данных, например *Документ(n)*, *Лист(n)*, *Диаграмма(n)*, где *n* — порядковый номер файла этого типа. Убедитесь, что предлагаемое имя не будет конфликтовать с существующими именами в текущем каталоге. Это имя может быть использовано как предлагаемое по умолчанию при сохранении файла. В некоторых случаях для нового файла удобнее использовать более универсальное название — *Безымянный (Untitled)*.

Следуйте этому же соглашению, если ваше приложение содержит команду *Создать*, которая создает новые файлы. Избегайте указания имени пользователя для идентификации окна. Вместо этого вы можете открыть диалоговую панель *Сохранить как..*, которая позволяет пользователю подтверждать или изменять предлагаемое вами имя, когда они сохраняют или закрывают файл, либо пытаются создать новый файл.

Отображайте имя файла в заголовке окна точно в таком же виде, в каком оно представлено в файловой системе, в том числе с учетом использования символов верхнего и нижнего регистров.

Если имя отображаемого в окне объекта изменилось (например, после того как пользователь отредактирует его), необходимо отразить это изменение в заголовке окна. Всегда пытайтесь поддержать явную ассоциацию между объектом и открытым окном.




Когда ширина окна не позволяет отображать полностью название объекта, можно его сократить, однако это следует делать таким образом, чтобы сохранить существенную информацию, которая позволяет пользователю быстро идентифицировать окно.

Избегайте включения в полосу заголовка рисунков либо дополнительных элементов управления. Такие дополнительные элементы могут затруднить восприятие имени окна, особенно в тех случаях, когда размер заголовка изменяется вместе с изменением размера окна. Кроме того, система использует эту область для отображения специальных средств управления. Например, в некоторых локализованных версиях Windows в полосе заголовка выводится информация или элементы управления, связанные с использованием определенных языков.

Вместе с тем, рекомендуется включать в полосу заголовка кнопки, предназначенные для управления размерами и состоянием первичного окна. В табл. 4.1 приведено описание стандартных кнопок управления окном.

Таблица 4.1

Кнопки управления первичным окном

Кнопка	Команда	Выполняемые действия
	Заккрыть (Close)	Закрывает окно
	Свернуть (Minimize)	Сворачивает окно
	Развернуть (Maximize)	Устанавливает максимальный (полноэкранный) размер окна

	Восстановить (Restore)	Устанавливает номинальный размер окна
---	---------------------------	--

В данном случае под **номинальным** понимается размер, который был установлен пользователем перед выполнением команд *Заккрыть* или *Свернуть*.

Система не поддерживает для первичных окон использование кнопки вызова контекстно-зависимой справочной информации (Help), доступной во вторичных окнах. Если вы считаете, что ее наличие необходимо, можно включить кнопку *Help* в состав панели инструментов первичного окна. С другой стороны, не рекомендуется вставлять указанные выше кнопки управления (*Развернуть*, *Свернуть*, *Восстановить*) в полосу заголовка вторичных окон, поскольку соответствующие им команды не поддерживаются системой для этих окон.

Отображая кнопки управления первичного окна, используйте следующие руководящие принципы:

- Если команда не поддерживается окном, не отображайте соответствующую ей кнопку.
- Кнопка закрытия окна всегда должна быть самой правой кнопкой. Оставляйте промежуток между ней и всеми другими кнопками.
- Кнопка *Свернуть* должна предшествовать кнопке *Развернуть*.
- Кнопка *Восстановить* всегда заменяет кнопку *Развернуть* или кнопку *Свернуть* после выполнения соответствующей команды.

Когда пользователь открывает новое первичное окно, для него на Панели задач создается **кнопка входа**. Эта кнопка предоставляет пользователю доступ к командам соответствующего окна и обеспечивает переключение на это окно после работы с каким-либо другим из открытых окон. Размер кнопок входа на Панели задач регулируется автоматически таким образом, чтобы разместить на ней по возможности кнопки для всех открытых окон. Если при этом размер кнопки оказывается слишком мал, чтобы уместить полное название окна, для нее автоматически создается всплывающая подсказка (как показано на рис. 4.7), которая отображает полное название первичного окна.

Когда окно сворачивается, его кнопка входа по-прежнему остается на Панели задач; она удаляется только в том случае, если пользователь закрывает окно.

Рис. 4.7. Всплывающая подсказка для кнопки входа

Кнопки входа могут также использоваться в качестве объекта-приемника при выполнении операций пересылки. Когда пользователь помещает перетаскиваемый объект над кнопкой входа, система активизирует связанное с этой кнопкой окно, позволяя пользователю поместить объект в окно.

4.2.2. ОСНОВНЫЕ ОПЕРАЦИИ С ОКНАМИ

К основным операциям с окнами относятся: изменение состояния (активно/неактивно), открытие и закрытие, перемещение, изменение размера, прокрутка, разбиение.

Изменение состояния окна

Если даже система поддерживает многооконность, пользователь, тем не менее, обычно выполняет некоторую последовательность действий в пределах одного окна. Это окно называется **активным**. Активное окно, как правило, является окном самого верхнего уровня (т.е. расположено «поверх» других открытых окон). Визуально оно отличается своим заголовком, который подсвечивается специальным «активным» цветом. В каждый момент времени может быть активным только одно окно. Хотя Другие окна могут относиться к активным (выполняющимся) процессам, но только одно — активное — окно способно воспринимать информацию (команды или данные), вводимую пользователем. Заголовок неактивного окна отображается системным «неактивным» цветом (при необходимости приложение может запросить установленные системные цвета заголовка для активного и неактивного окна).

Пользователь активизирует требуемое первичное окно, переключаясь на него; это автоматически приводит к изменению состояния всех других первичных окон. Для того, чтобы активизировать окно с помощью мыши, пользователю достаточно нажать любую ее клавишу, поместив указатель в произвольную точку окна, включая его контур. Если окно находится в свернутом виде, то для его активизации следует щелкнуть мышью на полосе заголовка окна.

При работе с клавиатурой для переключения между первичными окнами используется комбинация клавиш <Alt>+<Tab>. Комбинация клавиш <Shift>+<Alt>+<Tab> также обеспечивает переключение между окнами, но в обратном порядке (для переключения между окнами система также поддерживает комбинацию <Alt>+<Esc>). Повторная активизация окна не должна влиять на любой предшествующий выбор в пределах этого окна; другими словами, при активизации окна область выбора и фокус ввода восстанавливаются в том же виде, какой они имели в предыдущем активном состоянии. Кроме того, при повторной активизации первичного окна само окно и все связанные с ним вторичные окна становятся окнами верхнего уровня, причем восстанавливается их взаимное расположение.

Когда окно становится неактивным, уберите визуальное отображение выбора в пределах этого окна, чтобы явно указать, какое именно окно получает данные, вводимые с клавиатуры. Исключением является выполнение операций прямого манипулирования (например, копирования объекта из одного окна в другое). В этом случае можно восстановить визуальное отображение выбора в окне-получателе на то время, пока указатель находится над окном (т.е. в течение выполнения операции прямого манипулирования).

Поскольку разрабатываемое приложение должно, как правило, выполняться на различных модификациях аппаратных средств, нельзя делать его зависящим от конкретных технических характеристик этих средств (например, от разрешающей способности монитора). При использовании стандартного системного интерфейса система автоматически позиционирует окна приложения в соответствии с характеристиками используемого монитора.

Открытие и закрытие окон

При открытии первичного окна оно автоматически становится активным и располагается на самом верхнем уровне. Если пользователь пытается открыть первичное окно, которое ранее уже было открыто в пределах того же Рабочего стола, активизируйте существующее окно, используя приведенные ниже

рекомендации (табл. 4.2). Если существующее окно свернуто, разверните его, восстановив его предыдущее состояние.

Пользователь закрывает первичное окно, нажимая кнопку *Заккрыть* в заголовке окна или выбирая команду *Заккрыть* из всплывающее меню окна. Хотя система поддерживает двойной щелчок мышью на пиктограмме заголовка окна как ускоренный способ закрытия окна (для совместимости с предшествующими версиями Windows), старайтесь избегать использования этой операции в качестве основного способа закрытия первичного окна.

Если ваше приложение автоматически не сохраняет результаты редактирования, или последние внесенные изменения еще не сохранены в файле, то при закрытии пользователем окна отобразите сообщение, запрашивающее у пользователя необходимость сохранения этих изменений прежде, чем закрыть окно. Если не сохраненные изменения отсутствуют, такой запрос выводить не требуется, просто закройте окно. Следуйте тому же соглашению для любой другой команды, которая заканчивается закрытием первичного окна (например, при выходе из приложения).

Таблица 4.2.

Действия при повторном выполнении операции Открыть

Тип файла	Действие при повторном выполнении операции <i>Открыть</i>
Документ или файл данных	Активизируется существующее окно объекта, которое отображается поверх других открытых окон
Файл приложения	Выводится сообщение о том, что открытое окно этого приложения уже существует; пользователю предлагается либо переключиться на открытое окно, либо открыть новое. В любом случае окно активизируется и отображается поверх других открытых окон
Файл, который уже открыт в окне MDI-приложения	Активизируется существующее окно файла; при этом содержащее его родительское окно MDI-приложения отображается поверх других окон
Файл еще не открыт, но связанное с ним MDI-приложение уже загружено (открыто)	Открывается новый экземпляр связанного MDI-приложения (поверх других окон) и в родительском окне отображается дочернее окно для данного файла. В качестве альтернативного варианта дополнительно может выводиться сообщение о том, что открытое окно этого приложения уже существует и пользователю предлагается выбор: использовать существующее окно или открыть новое родительское окно

Закрывая первичное окно, закройте также все его подчиненные вторичные окна. При разработке приложения определите, что закрытие первичного окна означает также завершение работы приложения и всех связанных с ним процессов. Например, закрытие окна текстового документа, как правило, вызывает завершение процессов, связанных с вводом новой информации в документ или с его форматированием. Тем не менее, закрытие окна принтера не должно приводить к удалению заданий из очереди.

Перемещение окон

Пользователь может переместить окно, либо установив указатель мыши на полосу заголовка окна, либо используя команду *Переместить* из всплывающего меню этого окна. В большинстве конфигураций ОС Windows вместе с указателем перемещается только структурированное представление окна (его контуры), а полное изображение окна восстанавливается в новой позиции после завершения перемещения.

При выборе команды *Переместить* пользователь может переместить окно с помощью клавиатуры, используя клавиши управления курсором и нажимая клавишу <Enter>, чтобы закончить операцию и установить новую позицию окна. Никогда не разрешайте пользователю переместить окно так, чтобы оно стало недоступно.

Никакое другое окно не должно стать активным прежде, чем пользователь завершит перемещение данного окна. Перемещение окна подразумевает его активизацию.

Перемещение окна может привести к усечению выводимой в окне информации, или наоборот, сделать видимой другую часть информации. Кроме того, активизация окна при перемещении может повлиять на изображение окна (например, может быть отображен текущий выбор). Тем не менее, когда пользователь перемещает окно, избегайте внесения любых изменений в содержимое данного окна.

Изменение размеров окна

Предоставьте пользователю возможность изменять размеры первичного окна, если информация, отображенная в окне, не должна фиксироваться (как, например, в стандартной программе *Калькулятор* ОС Windows). Система поддерживает различные соглашения, которые обеспечивают изменение размеров окна пользователем.

Калибровка границ окна

Пользователь может изменять размеры первичного окна, передвигая его границы с помощью мыши, или используя команду *Размер* в меню окна. При этом контурное изображение окна перемещается вместе с указателем (в некоторых конфигурациях система может включать опцию вывода, позволяющую динамически перерисовывать окно по мере его изменения). После завершения изменения размера окно принимает новый вид.

Используя клавиатуру, пользователь может корректировать размер окна, выбрав команду *Размер* и, используя клавиши перемещения курсора; нажатие клавиши <Enter> означает завершение операции.

Никакое другое окно не должно стать активным прежде, чем пользователь завершит изменение размеров данного окна. Изменение размеров окна подразумевает его активизацию, и это окно остается активным после завершения операции.

Когда пользователь уменьшает размеры окна, следует определить, как отображать выводимую в нем информацию. Выбор метода определяется, как правило, текущей ситуацией и типом отображаемой информации. В общем случае сокращение окна должно приводить к усечению выводимой информации. Тем не

менее, в тех ситуациях, когда вы хотите, чтобы пользователь видел как можно больше, вы можете использовать другие методы, например, масштабирование информации. Эти изменения должны быть тщательно продуманы и не менее тщательно реализованы, так как во многих случаях достаточно сложно соблюсти соответствие между изменением размеров окна и изменением масштаба отображаемой в нем информации. Кроме того, избегайте применения таких методов, когда важны удобочитаемость или сохранение структурного представления выводимой информации.

Хотя размер первичного окна может изменяться в соответствии с предпочтениями пользователя, вы можете определить максимальный размер окна. Определяя этот размер, учитывайте необходимость рационального соотношения между размерами окна и размерами экрана.

Развертывание окна

Несмотря на то, что пользователь может непосредственно изменять размеры окна (в пределах установленных вами ограничений), команда *Развернуть* оптимизирует выполнение этой операции. Включите эту команду в управляющее меню окна, а также выведите в виде соответствующей кнопки в полосе заголовка.

Расширение окна приводит к максимально возможному («оптимальному») увеличению его размеров. По умолчанию максимальный размер устанавливается равным размеру экрана монитора (исключая пространство, используемое для вывода панели задач или инструментальной панели Рабочего стола). Для дочернего окна при использовании MDI максимальным размером является размер родительского окна. Но в некоторых случаях вы можете определить максимально допустимый размер и по-иному.

Когда пользователь расширяет окно, замените кнопку *Развернуть* кнопкой *Восстановить*. Соответственно, в всплывающем меню окна сделайте недоступной команду *Развернуть* и, наоборот, разрешите использование команды *Восстановить*.

Сворачивание окна

Сворачивание окна приводит к удалению его с экрана и переходу в неактивное состояние. Чтобы обеспечить выполнение этой команды, включите ее во всплывающее меню окна, а также выведите в виде соответствующей кнопки в полосе заголовка окна.

При сворачивании первичного окна его кнопка входа остается на Панели задач. Для свернутого дочернего окна MDI-приложения создается кнопка входа, расположенная в пределах его родительского окна (рис. 4.8).

Рис. 4.8. Кнопка входа для дочернего окна

Когда пользователь сворачивает окно, сделайте недоступной в меню окна команду *Свернуть* и разрешите использование команды *Восстановить*.

Восстановление окна

Команда *Восстановить* обеспечивает восстановление предшествующего размера окна и его расположение на экране после того, как пользователь разворачивал или сворачивал окно. Данная команда должна быть доступна в меню окна для развернутых окон; в полосе заголовка окна кнопка *Восстановить* заменяет кнопку *Развернуть*.

Для свернутых окон команда *Восстановить* также должна быть доступна в меню окна.

Пользователь может восстановить свернутое первичное окно до его прежнего размера и в предыдущую позицию, либо щелкнув ЛКМ на кнопке входа в окно на Панели задач, либо выбрав команду *Восстановить* в меню окна, либо используя комбинацию клавиш <Alt>+<Tab> (или <Shift>+<Alt>+<Tab>).

Регулятор окна

При создании масштабируемого окна вы можете включить в его состав специальную графическую область, используемую для пропорционального изменения окна по обоим измерениям (*регулятор*). Установив указатель мыши на регулятор и нажав ее левую кнопку, пользователь может изменять размер окна по обоим измерениям, перемещая указатель в нужном направлении. Наличие регулятора не исключает возможность масштабирования окна путем перемещения его границ (то есть по каждому измерению в отдельности).

Всегда располагайте регулятор в правом нижнем углу окна. Обычно эта позиция совпадает с правым концом горизонтальной полосы прокрутки или с нижним концом вертикальной полосы прокрутки. Если окно содержит строку состояния, отобразите регулятор окна в крайней правой позиции этой строки. Никогда не отображайте регулятор в двух позициях окна одновременно.

Прокрутка окна

В том случае, когда выводимая в окне информация превышает размер этого окна, оно должно поддерживать возможность *прокрутки*. Прокрутка дает пользователю возможность просматривать части объекта, которые в настоящий момент времени не видны в окне. Эта функция обычно реализуется с помощью *полос прокрутки*. Полоса прокрутки представляет собой прямоугольную область, содержащую стрелки, которые указывают разрешенное направление прокрутки, и ползунок, величина и положение которого отражают размер невидимой части объекта. Как стрелки, так и ползунок являются интерактивными элементами (то есть они реагируют на воздействие пользователя). Стрелка полосы прокрутки представляет собой кнопку, при нажатии которой информация в окне перемещается в соответствующем направлении на одну дискету; величина шага прокрутки устанавливается разработчиком приложения.

Окно может содержать либо только одну полосу прокрутки (вертикальную или горизонтальную), либо обе. Если известно, что содержимое окна никогда не будет перемещаться в конкретном направлении, не включайте в окно полосу прокрутки для этого направления.

Практика показывает, что для тех окон, где может потребоваться перемещение информации, полосы прокрутки должны отображаться всегда, даже если окно становится неактивным или выводимая информация не требует

прокрутки. Это обусловлено следующими обстоятельствами. Удаляя полосы прокрутки, когда окно становится неактивным, вы увеличиваете свободное пространство окна. Тем самым вы изменяете рациональное соотношение между размерами окна и объемом отображаемой информации. Кроме того, в этом случае пользователь вынужден явно активизировать окно, прежде чем выполнить прокрутку. И, наконец, последовательность в отображении полос прокрутки, обеспечивает большую стабильность рабочей среды пользователя.

Подробнее техника использования полос прокрутки рассматривается в разделе «Элементы управления».

Автоматическая прокрутка

Перемещение информации в окне с помощью полос прокрутки предполагает явное указание пользователя на необходимость прокрутки. Вместе с тем, прокрутка может также быть обусловлена каким-либо другим действием пользователя. Такой тип прокрутки называется *автоматической прокруткой*. Целесообразно поддерживать автоматическую прокрутку в следующих ситуациях:

- Когда пользователь формирует непрерывную область выбора и в нее должны быть включены объекты, не видимые в данный момент в окне;
- Когда пользователь перемещает объект и достигает края видимой области, но хочет продолжить перемещение;
- Когда пользователь вводит текст с клавиатуры в крайнюю позицию окна, либо перемещает или копирует объект в такую позицию;
- Если выполнение операции заканчивается изменением текущей активной позиции (например, перемещением границы области выбора или фокуса ввода) и новая позиция находится за пределами видимой области. Например, при выполнении операции *Найти* следует сделать видимым фрагмент документа, содержащий найденный объект.

Автоматическую прокрутку могут вызвать и другие виды навигации. Например, при заполнении экранной формы завершение ввода информации в одном поле может потребовать переход к другому полю. В том случае, если новая область ввода не видима, форма может быть соответствующим образом перемещена.

Клавиатурная прокрутка

Для выполнения операции прокрутки с помощью клавиатуры используются клавиши навигации. Когда пользователь нажимает клавишу навигации, курсор переходит на соответствующую позицию. Дополнительно к перемещению курсора, нажатие любой из этих клавиш может приводить к перемещению отображаемой области в соответствующем направлении. В свою очередь, использование клавиш <PgUp> и <PgDn> аналогично щелчку ЛКМ в чувствительной области полосы прокрутки.

4.2.3. ИСПОЛЬЗОВАНИЕ ПОДОКОН

Окно может разделяться на две или более относительно независимые области, которые называются **подокнами**. Разделение окна позволяет пользователю, например, просматривать одновременно две части одного документа. Технологию разделения окна можно также использовать для того, чтобы отобразить одну и ту же информацию в различной форме (рис. 4.9).

Рис. 4.9. Окно, разделенное на подокна

Если же наоборот, необходимо одновременно получить доступ к нескольким файлам в рамках выполнения одного задания, следует использовать другую технику управления окнами, например, MDI.

Разбиение окна на подокна может быть установлено либо разработчиком приложения (как основная форма окна), либо пользователем, посредством задания соответствующего параметра. Для того чтобы поддерживать разбиение окна, которое не определено заранее, включите в состав создаваемой программы так называемый **блок разделения**. Блок разделения является специальным элементом управления, который отображается в конце полосы прокрутки окна и обозначает регулируемую границу между подокнами. Размер блока разделения должен быть достаточно большим, чтобы пользователь мог работать с ним, используя указатель мыши.

Пользователь может изменять размеры подокон, перемещая блок разделения в нужную позицию. Когда указатель мыши находится над блоком разделения, целесообразно изменить его образ, чтобы обеспечить обратную связь и помочь пользователю правильно установить блок разделения. С той же целью рекомендуется отображать контуры перемещаемого блока разделения и **полосы разделения** (границы подокон) до тех пор, пока пользователь не закончит операцию (рис. 4.10).

Рис. 4.10. Перемещение границы подокон

Вы можете разрешить перемещение полосы разделения (или блока разделения) в конец полосы прокрутки. При этом одно из подокон окажется закрытым. Дополнительно вы можете определить двойной щелчок ЛКМ в некоторой позиции, заданной по умолчанию (например, в середине окна) как ускоренный способ разбиения окна или для удаления разделения. Такой способ наиболее удобен в тех случаях, когда результирующие подокна должны быть одного размера.

Для того, чтобы обеспечить клавиатурный интерфейс для разбиения окна, включите во всплывающем меню окна (или в раздел меню *View*) команду *Разделить*. Когда пользователь выбирает эту команду, разделите окно посередине или в контекстно-определенной позиции.

Для перемещения блока разделения могут быть использованы клавиши управления курсором; нажатие клавиши <Enter> устанавливает разделение в текущей позиции, а нажатие клавиши <Esc> отменяет режим разделения.

Для разделения окна могут применяться и другие средства. Например, вы можете определить специальные режимы отображения окна, которые позволяют пользователю выбирать фиксированное или изменяемое расположение подокон, либо удалять разделение окна, закрывая подокно (или используя какую-либо другую команду управления форматом окна).

Когда пользователь разделяет окно, дополните его полосами прокрутки для тех подокон, которые этого требуют. Кроме того, следует разместить

информацию в подокнах таким образом, чтобы полоса разделения не закрывала содержимое окна, над которым она появляется. Используйте одну общую полосу прокрутки для тех подокон, которые перемещаются вместе; если же каждое из подокон требует независимую прокрутку, полоса прокрутки должна создаваться в каждом из них. Например, если основное окно разделено на подокна по горизонтали, то, очевидно, каждое из них должно иметь собственную вертикальную полосу прокрутки, управляемую отдельно.

При использовании подокон каждое из них должно иметь собственные значения атрибутов, таких, например, как тип шрифта и состояние выбора. При этом область выбора следует отображать только в активном подокне. Тем не менее, если состояние выбора распространяется через подокна, отобразите выбор во всех подокнах.

Когда основное окно закрывается пользователем, запомните состояние подокон (количество, расположение, отображаемая информация, состояние выбора) как часть информации о состоянии этого окна, с тем, чтобы оно могло быть восстановлено при следующем открытии в том же виде.

4.2.4. МНОГОДОКУМЕНТНЫЙ ИНТЕРФЕЙС (MDI)

В процессе работы с одним и тем же приложением пользователю может потребоваться иметь на экране несколько открытых окон, содержащих информацию различных типов, либо представляющих собой разное изображение одних и тех же данных.

Для создания таких окон и управления ими существует специальная технология — **многодокументный интерфейс**, сокращенно MDI (Multiple Document Interface). В этом разделе описаны особенности её применения с точки зрения реализации пользовательского интерфейса.

СТАНДАРТНАЯ РЕАЛИЗАЦИЯ MDI

Во многих случаях взаимодействие пользователя с приложением ограничено рамками единственного первичного окна, которые могут быть несколько расширены за счет применения дополнительных вторичных окон. Рабочий стол и Панель задач обеспечивают управление несколькими первичными окнами, относящимися к разным приложениям. То окно, которое было открыто последним, отображается поверх других окон и для него создается вход на Панели задач; такая техника обеспечивает пользователям возможность переключения между окнами и перемещения их по поверхности экрана. Подход, при котором на экране может быть оставлено открытым единственное окно, соответствующее выбранному пользователем объекту, обеспечивает визуализацию взаимно-однозначного отношения между объектом и окном.

Некоторые типы объектов, например, устройства, могут даже не требовать создания собственного первичного окна и использовать только вторичное окно для просмотра и редактирования их свойств. В очень редких случаях единственным средством визуального представления объекта является его пиктограмма.

Вместе с тем, для выполнения некоторых заданий может оказаться недостаточным наличие одного первичного окна. В таких ситуациях работа приложения должна быть построена на основе многодокументного интерфейса.

Техника MDI заключается в использовании одного первичного окна, называемого **родительским окном**, которое может содержать набор связанных с ним **дочерних окон** (рис. 4.11). Каждое дочернее окно — это, по существу, также первичное окно, единственным ограничением для которого является то, что оно может появиться только в пределах родительского окна. Родительское окно обеспечивает как визуальное, так и «операционное» пространство для своих дочерних окон. Например, на дочернее окно обычно распространяется область действия меню родительского окна и, возможно, других элементов его интерфейса (панели инструментов, строки состояния и т.д.). Их вид может изменяться, если необходимо отразить команды и атрибуты активного дочернего окна.

Вторичные окна, такие как диалоговые панели, окна сообщений или панели свойств, появляются на экране как результат тех или иных действий пользователя в родительском или дочернем окне. Эти окна должны активизироваться и отображаться в соответствии с общими соглашениями для вторичных окон, связанных с первичным окном, даже если они относятся к дочернему окну.

Заголовок родительского окна обычно содержит пиктограмму и имя приложения или объекта, который оно представляет. Заголовок дочернего окна содержит пиктограмму, представляющую тип документа или файла данных, и имя файла.

Рис. 4.11. Родительское и дочерние окна MDI

Как для родительского окна, так и для всех его дочерних окон должны поддерживаться всплывающие меню; перечень команд в таком меню аналогичен рассмотренному выше применительно к первичным окнам.

Пользователь может активизировать MDI-приложение либо непосредственно открыв его, либо открыв документ (файл данных) того типа, который поддерживается этим приложением. Если MDI-приложение активизировано посредством открытия документа, сначала открывается родительское окно, а затем внутри его рабочей области — дочернее окно, отображающее выбранный документ или файл. Для того, чтобы упростить пользователю открытие других документов, связанных с этим приложением, включите в его интерфейс диалоговую панель *ОТКРЫТЬ*.

В том случае, когда пользователь непосредственно открывает документ за пределами и родительского окна MDI-приложения (например, с помощью двойного щелчка ЛКМ на пиктограмме файла), то если родительское окно уже открыто, следует создать второй экземпляр приложения (т. е. еще одно родительское окно), а не окно документа (дочернее) в существующем родительском окне. Хотя открытие нового дочернего окна может быть более эффективным, его появление может нарушить среду задания, уже установленную в этом родительском окне. Например, если вновь открытый файл является макросом, его открытие в уже открытом родительском окне может повлиять на другие документы, открытые в этом окне. Чтобы пользователь мог открыть файл в конкретном родительском окне, следует определить команды, поддерживающие такую возможность.

Поскольку дочерние окна MDI являются разновидностью первичных окон, при их закрытии следует придерживаться тех же соглашений, которые были рассмотрены ранее для первичных окон. Когда пользователь закрывает дочернее окно, любые не сохраненные изменения должны быть обработаны в соответствии с общими соглашениями для всех первичных окон.

Приложение не должно разрешить пользователю закрыть дочернее окно, если это не позволит ему продолжить работу с приложением.

Когда пользователь закрывает родительское окно, закройте все его дочерние окна. Где возможно, сохраняйте состояние дочернего окна (размер и положение внутри родительского окна) и восстанавливайте это состояние, когда пользователь вновь открывает окно.

MDI позволяет пользователю перемещать или сворачивать дочернее окно таким же образом, как и родительское окно. Когда пользователь перемещает родительское окно, сохраните относительные позиции открытых дочерних окон в рабочей области родительского окна. Перемещение дочернего окна ограничивается размерами его родительского окна; в некоторых случаях это может привести к усечению дочернего окна (рис. 4.12). В связи с этим дополнительно может быть определена возможность автоматического изменения размеров родительского окна при перемещении или изменении размеров дочернего окна.

Рис. 4.12. Свёрнутое и смещённое дочерние окна

Рекомендуемая форма отображения минимизированного дочернего окна в MS Windows — часть полосы заголовка окна, окруженная рамкой. Это позволяет избежать возможной путаницы между пиктограммами минимизированных дочерних окон и пиктограммами, представляющими объекты.

Хотя свернутое родительское окно представляется кнопкой входа в окно на Панели задач, свернутое дочернее окно располагается в пределах своего родительского окна, как показано на рис. 4.12.

Когда пользователь разворачивает родительское окно, оно должно приобретать свой максимальный (полноэкранный) размер, подобно любому другому первичному окну. Это же правило должно выполняться и для дочернего окна. Если его полноэкранный размер превышает размер родительского окна, объедините дочернее окно с родительским. При этом пиктограмма заголовка дочернего окна, кнопки *Восстановить*, *Заккрыть*, и *Свернуть* (если они поддерживаются) должны располагаться в полосе меню родительского окна в тех же позициях, в которых они находились в полосе заголовка дочернего окна (рис. 4.13). Кроме того, текст названия дочернего окна добавляется к заголовку родительского окна.

Если пользователь, развернув одно из дочерних окон, затем переключается на другое, отобразите это окно тоже как развернутое. Аналогично если пользователь восстанавливает одно из дочерних окон в его номинальное состояние, восстановите все другие дочерние окна в их номинальных размерах.

Рис. 4.13. Развёрнутое дочернее окно

Для активизации дочерних окон и навигации между ними с помощью мыши используются те же общие соглашения, что и для первичных окон. Для быстрого клавиатурного доступа и переключения между дочерними окнами рекомендуется использовать комбинации клавиш <Ctrl>+F6 и <Ctrl>+<Tab> (и <Shift>+ модифицирующая комбинация для циклического повторения). Кроме того, целесообразно включить в меню *Окно* родительского окна команды для переключения между дочерними окнами и команды управления размещением окон в пределах родительского окна (например, *Мозаикой* или *Каскадом*).

Когда пользователь переключается на дочернее окно, приложение может изменить интерфейс родительского окна, например, видоизменить меню, панель инструментов или строку состояния, чтобы отразить перечень действий, применимых к этому дочернему окну. Тем не менее, старайтесь обеспечить согласованность между интерфейсом родительского и дочерних окон, в частности, сохраните неизменными меню, которые обеспечивают работу с файлами и управление приложением или общей средой родительской окна (например, меню *Файл* или меню *Окно*).

АЛЬТЕРНАТИВНЫЙ ПОДХОД К РЕАЛИЗАЦИИ MDI

Технология MDI имеет свои ограничения. В частности, MDI-приложение постоянно находится в центре внимания пользователя, нарушая тем самым принцип работы, управляемой данными. Имеется в виду следующее. Хотя пользователь может запустить приложение MDI непосредственно открытием одного из своих документов или файлов данных, тем не менее чтобы работать с

несколькими документами в одном и том же родительском окне, он должен использовать интерфейс приложения для открытия этих документов.

Когда пользователь открывает несколько файлов в пределах одного родительского окна, нарушается согласованность связи между дочерними окнами и между отображаемыми в них объектами. Несогласованность заключается в том, что хотя родительское окно визуальнo объединяет дочерние окна (как бы играет роль контейнера), это не приводит к аналогичному объединению отображаемых файлов. Это делает отношения между файлами и их окнами более абстрактными, затрудняя уяснение принципов MDI начинающими пользователями.

Вследствие того, что родительское окно в действительности не содержит объекты, представленные в дочерних окнах, технология MDI не может обеспечить эффект непрерывной работы пользователя. Речь идет о том, что когда пользователь закрывает родительское окно и затем вновь открывает его, созданная ранее рабочая среда не восстанавливается и пользователь вынужден вновь выполнять операции, связанные с открытием файлов, с которым он работал в последний раз.

Необходимо также иметь в виду, что технология MDI может усложнить некоторые аспекты использования OLE. Например, если пользователь открывает текстовый документ в MDI-приложении и затем открывает (с помощью другого приложения) электронную таблицу, вставленную в этот текстовый документ, то визуальная связь между текстовым документом и таблицей нарушается, поскольку окно с электронной таблицей не становится дочерним для того же родительского окна MUI.

И, наконец, еще одно обстоятельство. Ограничения, налагаемые технологией MDI на размещение дочерних окон в родительском окне, могут оказаться критичными для некоторых заданий, например, при проектировании окон или выборе формата инструментальных средств. Кроме того, иногда пользователь может не отличить дочерние окна, «вложенные» в родительское окно, от первичных окон, расположенных поверх него.

Перечисленные недостатки MDI могут быть в значительной степени преодолены за счет применения альтернативных средств, таких как **Рабочие области** (Workspaces), **Рабочие книги** (Workbooks) и **Проекты** (Projects). Хотя эти средства реализуют однооконную модель интерфейса, тем не менее они обнаруживают целый ряд достоинств, присущих технологии MDI. В частности, с их помощью можно получить различные формы представления одних и тех же данных. Эти средства обеспечивают также большую гибкость относительно включаемых в них типов объектов. Тем не менее, как и любой контейнер, они могут быть настроены на хранение и управление только определенным типом объектов.

Рассмотренные ниже примеры только иллюстрируют некоторые альтернативные подходы к реализации MDI-технологии и не исключают подложности использования других подходов. Кроме того, здесь опущены некоторые специфические детали, реализация которых зависит от разработчика приложения.

РАБОЧАЯ ОБЛАСТЬ

Рабочая область обладает многими характеристиками MDI, в том числе возможностью управления набором взаимосвязанных окон, отображаемых в пределах родительского окна, а также использования ими элементов интерфейса родительского окна, таких как меню, панель инструментов и строка состояния (рис. 4.14).

Использование Рабочей области в качестве контейнера

Основное отличие Рабочей области от MDI заключается в использовании концепции объединения отображаемых объектов. Это означает, что объекты, отображаемые в Рабочей области, могут соответствовать файлам, содержащимся в одном и том же контейнере. Внешне же соответствующие им окна выглядят как дочерние окна, расположенные в пределах родительского окна. Таким образом, концепция

Рис. 4.14. Пример Рабочей области

использования Рабочей области подобна концепции использования Рабочего стола, за исключением того, что она сама является объектом, который может быть представлен в виде пиктограммы и отображен в виде открытого окна. Чтобы окно объекта могло появиться в Рабочей области, сам объект должен входить в состав соответствующего контейнера.

В настоящее время реализация механизма хранения объектов зависит от типа используемого контейнера. Содержимое родительского окна может представлять либо единственный файл, либо вы можете разработать ваш собственный механизм, позволяющий отобразить содержимое файловой системы. Продумайте возможность использования технологии OLE, чтобы облегчить взаимодействие между вашей Рабочей областью, оболочкой и другими приложениями. Предоставьте пользователю возможность перемещать объекты из Рабочей области в другие контейнеры, например, на Рабочий стол или в произвольный каталог. Вместе с тем, пользователь должен иметь возможность открыть объект в его индивидуальном окне, не входящем в окно Рабочей области и обладающем собственным интерфейсом.

Поскольку Рабочая область является объектом, то для нее необходимо определить набор свойств и перечень разрешенных команд, в том числе команды создания новых объектов в пределах Рабочей области и команду *Сохранить все*, позволяющую запомнить состояние всех объектов, открытых в рабочей области.

Использование Рабочей области для группирования заданий

Поскольку Рабочая область визуально объединяет пиктограммы и окна объектов, вы можете определить ее таким образом, чтобы разрешить пользователю создавать набор объектов, необходимых для выполнения

конкретного задания. Подобно MDI, это облегчит пользователям перемещение или переключение между взаимосвязанными окнами.

Рабочая область аналогична также MDI в том, что интерфейс родительского окна может распространяться на его дочерние окна. Например, если Рабочая область содержит меню, то входящие в него команды могут быть использованы для всех его дочерних окон. Если же Рабочая область не имеет меню, или если вы предоставляете пользователю право скрыть его, меню должно появляться в дочернем окне. Продумайте также вариант совместного использования дочерними окнами панели инструментов и строки состояния родительского окна.

Управление окнами в Рабочей области

При управлении окнами в Рабочей области используются те же соглашения, что и в MDI-приложениях. Когда Рабочая область закрывается, все дочерние окна также закрываются. Следует сохранить состояние этих окон (в частности, их размер и расположение), чтобы его можно было восстановить при следующем открытии Рабочей области.

Подобно большинству первичных окон, когда пользователь сворачивает окно Рабочей области, оно исчезает с экрана, а на Панели задач появляется кнопка входа в него. Свернутые дочерние окна Рабочей области выглядят и ведут себя так же, как свернутые дочерние окна MDI-приложения. Если размер развернутого дочернего окна превышает размер окна Рабочей области, то оно объединяется с окном Рабочей области, а пиктограмма заголовка окна и управляющие кнопки окна появляются в полосе меню родительского окна Рабочей области.

Рабочая область должна обеспечить средства управления дочерними окнами в пределах Рабочей области. В частности, меню *Окно* и всплывающее меню родительского окна должны дополнительно содержать команды, предназначенные для прямой активизации и перемещения открытых дочерних окон.

РАБОЧАЯ КНИГА

Рабочая книга — это еще один альтернативный вариант управления формой представления отображаемых данных, в основе которого лежит метафора книги или записной книжки. В Рабочей книге различные формы данных представляются как отдельные разделы в пределах одного первичного окна. Рис. 4.15 иллюстрирует один из возможных способов реализации Рабочей книги.

В качестве средства навигации между разделами Рабочей книги могут использоваться этикетки вкладок, которые будут подробнее описаны в главе 5. Их расположение в Рабочей книге определяется разработчиком приложения в зависимости от содержания

Рис. 4.15. Пример Рабочей книги

и организации отображаемой информации. Каждый раздел представляет данные, которые могли бы быть отдельным документом. В отличие от папки или Рабочей области, Рабочая книга лучше подходит для представления таких данных, которые могут быть определенным образом упорядочены и этот

порядок имеет существенное значение. Кроме того, вы можете дополнительно включить специальный раздел, перечисляющий содержание Рабочей книги подобно оглавлению обычной книги. Он может также быть использован как часть навигационного интерфейса Рабочей книги.

Для Рабочей книги действительны те же соглашения, которые регламентируют взаимосвязь родительского и дочерних окон в MDI-приложении. В частности, разделы могут использовать элементы интерфейса родительского окна, например, меню и строку состояния. Когда пользователь переключает разделы, меню может изменяться таким образом, чтобы в наибольшей степени соответствовать текущему объекту. Если пользователь закрывает Рабочую книгу, должны соблюдаться общие правила обработки не сохраненных изменений, в том числе должна существовать возможность их отмены.

Целесообразно также поддерживать для Рабочей книги технологию OLE, чтобы обеспечить возможность выполнения операций пересылки, копирования и связывания объектов. Может быть также реализована команда *Поместить (Insert)*, чтобы разрешить пользователю создавать новые объекты, в том числе новые разделы Рабочей книги. И, наконец, для Рабочей книги, как и для Рабочей области, применима команда *Сохранить все*, которая сохраняет перед закрытием книги внесенные изменения или уточняет у пользователя, нужно ли эти изменения сохранить или отвергнуть.

ПРОЕКТ

Проект реализует ещё один альтернативный способ управления окном, который предусматривает возможность отображения в одном окне взаимосвязанных объектов, представленных их пиктограммами. В этом смысле Проект подобен каталогу: для выбранного объекта открывается окно, которое относится к тому же уровню, что и родительское окно. В результате, каждое дочернее окно Проекта может также иметь собственную кнопку входа на Панели задач. В отличие от каталога, Проект обеспечивает управление из родительского окна окнами входящих в него объектов. Например, когда пользователь, открыв документ, закрывает окно каталога, это никак не отражается на окне открытого документа. И напротив, когда пользователь закрывает окно Проекта, все окна его объектов также закрываются. Кроме того, когда пользователь открывает окно Проекта, это действие должно восстанавливать окна объектов в их предшествующем состоянии.

Рис. 4.16. Пример Проекта

Окна объектов, входящих в Проект, не могут использовать меню или элементы управления окна Проекта. Поэтому окно каждого объекта необходимо включить собственные элементы интерфейса. Тем не менее, вы можете создать окно палитры, содержащее набор инструментов, которые могут совместно использоваться окнами объектов Проекта.

По аналогии с Рабочей областью и Рабочей книгой, Проект должен иметь команды для создания новых объектов, для их пересылки в Проект и из него, а также для сохранения любых изменений объектов, входящих в Проект. Кроме того, окно Проекта должно содержать средства для работы с самим Проектом как с объектом (в том числе для изменения его свойств).

4.2.5. ВЫБОР МОДЕЛИ ОКНА

При выборе формы представления заданий или процессов, связанных с работой приложения, следует принимать во внимание целый ряд факторов: уровень знаний и навыков предполагаемых пользователей, особенности используемых объектов и решаемых с помощью приложения задач, требования эффективному использованию пространства экрана монитора, а также ориентацию на разработку, управляемую данными.

В частности, форма представления объекта зависит от способа его использования и взаимосвязи с другими объектами. Простые объекты, которые являются «самодостаточными», обычно не требуют создания собственного первичного окна; средства взаимодействия пользователя с ними могут быть ограничены набором команд меню и использованием напели свойств. Примером объекта такого типа может служить кнопка.

Объект, обладающий некоторым внутренним содержанием, которое должно быть доступно пользователю, дополнительно к перечню свойств требует наличия собственного первичного окна; примером такого объекта является текстовый документ. Во многих случаях для представления объекта оказывается достаточным наличие единственного окна, даже если его содержимое может изменяться. При этом изменение формы представления объекта в том же окне обеспечивается с помощью элементов пользовательского интерфейса, которые были описаны выше. Система использует единый стиль интерфейса окна для большинства используемых объектов, таких, например, как папки.

В тех случаях, когда структура объекта требует представления его одновременно в нескольких видах, или когда пользователю необходимо работать одновременно с несколькими объектами, более эффективным является применение технологии MDI, либо использование Рабочих областей, Рабочих книг и Проектов. Эти конструкции обеспечивают формирование рабочей среды пользователя, ориентированной на выполнение определенного задания. При этом технология MDI является наиболее подходящей для работы с несколькими однотипными объектами, а использование Рабочих книг позволяет оптимизировать навигацию пользователя между различными представлениями одного объекта. Недостатком Рабочей книги можно считать то, что та ограничивает способность пользователя видеть одновременно несколько представлений объекта. Рабочие области и Проекты обеспечивают более гибкую технику для просмотра и совмещения объектов и их окон. Используйте Рабочую область в тех случаях, когда пользователю может потребоваться сгруппировать пиктограммы объектов (или их окна), используемые при выполнении некоторого задания. Применение Проекта позволяет снять ограничения на расположение и формат дочерних окон. Вместе с тем, это преимущество достигается за счет увеличения сложности работы пользователя; кроме того, пользователю бывает весьма сложно отличить дочерние окна Проекта от окон других активных приложений.

После того, как выбрана модель (концепция) формируемых приложением окон, следует тщательно продумать требования к формату отображаемых в них информации. Для современных мониторов с высоким разрешением использование меню, панелей инструментов и строк состояния не представляет большой проблемы с точки зрения адекватного отображения любой требуемой информации. Вместе с тем, компоненты интерфейса не должны доминировать над рабочей областью пользователя, поскольку то может затруднить ему отыскание требуемых данных или манипулирование ими.

MDI, Рабочие области, Рабочие книги и Проекты допускают возможность совместного использования некоторых компонентов интерфейса несколькими окнами. В связи с этим пользователю всегда должно быть ясно, когда и в каком окне доступен конкретный элемент интерфейса. Хотя вы можете предусмотреть автоматическую коррекцию содержимого таких компонентов, как меню или строка состояния, тем не менее следует учитывать необходимость поддержания согласованности интерфейса в части использования общих функций. Например, если во всех дочерних окнах предусмотрена команда *Печать*, то соответствующая ей кнопка должна в них находиться в одной и той же позиции. Несоблюдение этого правила может существенно снизить эффективность работы пользователя. Необходимо так же иметь в виду, что наличие общих элементов интерфейса, усложняет его настройку при переключении пользователя с одного окна на другое.

Независимо от выбранной модели окна следует всегда разрешать пользователям индивидуальную настройку интерфейса, хотя бы на уровне «скрыть/отобразить» тот или иной элемент интерфейса. Однако при этом необходимо предусмотреть возможность альтернативного доступа пользователя к той функции, которая связана со скрытым элементом (например, с помощью всплывающего меню).

Необходимо еще раз отметить, что использование одного первичного окна, применение MDI Рабочих областей, Рабочих книг и Проектов не являются взаимоисключающими вариантами реализации пользовательского интерфейса приложения. Как и во многих других областях человеческой деятельности, наиболее аффективными оказываются, как правило, компромиссные решения. Например, рабочая книга и Проект могут быть реализованы как объекты, отображаемые в одной Рабочей области; аналогично, Проект может включать Рабочую книгу в качестве одного из своих объектов.

4.3. ВТОРИЧНЫЕ ОКНА

Вторичные окна предназначены для приема от пользователя или отображения дополнительной информации, которая, как правило, связана с объектами, представленными в первичном окне. Они позволяют значительно расширить диапазон средств диалогового взаимодействия пользователя с приложением, являясь дополнением к первичным окнам.

4.3.1. ОСНОВНЫЕ СВОЙСТВА ВТОРИЧНЫХ ОКОН

Вторичные окна обладают некоторыми и свойствами и первичных окон, тем не менее они отличаются от первичных окон во многих аспектах поведения и

использования. В частности, для вторичных окон не создаются кнопки входа на Панели задач.

Стандартное вторичное *с-кчо* содержит полосу заголовка окна и поле, ограниченное рамкой (рис. 4.17); пользователь может перемещать его, как и первичное окно, с помощью мыши, установив указатель на *полосу* заголовка окна. Тем не менее, вторичное окно не имеет кнопок *Развернуть* и *Свернуть*, поскольку данные операции обычно не применяются к вторичному окну. Чтобы закрыть окно, может использоваться кнопка *Заккрыть*, Заголовок окна является его меткой и поясняет назначение окна; полоса заголовка вторичного окна не содержит пиктограммы.

Рис.4.17. Вторичное окно

Разрешается включать во вторичные окна строку состояния, но не рекомендуется дублировать в ней элементы, используемые в строке состояния первичного окна.

Как и для первичного окна, для вторичного поддерживается всплывающее меню с командами управления окном; работа с всплывающим меню реализуется таким же образом, как и для первичного окна.

Вторичное окно может также содержать в полосе заголовка окна кнопку вызова контекстной помощи (*Что это ?*). Эта кнопка позволяет пользователю получать контекстно-зависимую справочную информацию о компонентах, отображённых в окне.

Вторичное окно может быть независимым или модальным.

Независимое вторичное окно позволяет пользователю взаимодействовать с другими вторичными или первичными окнами, а также переключаться между первичными окнами. Независимое вторичное окно целесообразно использовать в тех ситуациях, где пользователю может потребоваться повторить действие, связанное с этим окном (например, при поиске слова в тексте или при форматировании текста).

Модальное вторичное окно требует от пользователя завершить ввод данных в пределах данного окна и закрыть его, прежде чем продолжить работу за пределами окна. Вторичное окно может быть модальным по отношению к своему первичному окну или по отношению к системе. В последнем случае пользователь должен выполнить требующиеся действия и закрыть окно прежде, чем взаимодействовать с любыми другими окнами или объектами.

Поскольку модальные вторичные окна ограничивают диапазон возможных действий пользователя, используйте их только в ситуациях определенного типа.

В частности, в таких, когда для выполнения команды требуется ввести дополнительную информацию, или когда необходимо приостановить дальнейшую работу пользователя, пока не будет реализовано некоторое условие. Избегайте использования системных модальных вторичных окон, если ваше приложение не исполняется в качестве системной утилиты; но даже и в этом случае применяйте их только в наиболее серьезных ситуациях, игнорирование которых может привести к фатальной системной ошибке.

ВЗАИМОДЕЙСТВИЕ С ДРУГИМИ ОКНАМИ

Вторичные окна, которые отображаются при выполнении команд, выбранных в первичном окне, зависят от состояния первичного окна; это означает, что когда первичное окно закрывается или сворачивается, вторичные окна также закрываются или сворачиваются. Когда пользователь вновь открывает или восстанавливает первичное окно, восстановите вторичные окна в их прежнем положении и состоянии. Тем не менее, если открытие вторичного окна является результатом действия вне первичного окна (например, если пользователь выбирает команду *Свойства* для объекта, расположенного на Рабочем столе), то такое вторичное окно является независимым по отношению ко всем открытым первичным окнам и относится к тому же уровню иерархии, что и они (хотя для него, как было указано выше, и не создается кнопка входа на Панели задач).

Когда пользователь открывает вторичное окно или переключается на него, оно активизируется подобно любому другому окну. Активизация вторичного окна может быть выполнена как с помощью мыши, так и с помощью клавиатуры. Для переключения между вторичным и первичным окнами используется комбинация клавиш <Alt>+F6; эта же комбинация служит для перехода между другими вторичными окнами одного уровня с данным первичным окном. Чтобы поддержать такой способ переключения, вторичное окно должно быть **независимым**.

Когда пользователь активизирует первичное окно, оно отображается поверх других окон; при этом все его подчиненные вторичные окна также переносятся наверх, с сохранением их взаимного расположения. Аналогично, активизация любого вторичного окна приводит к «всплывтию» наверх его первичного окна и других связанных с ним вторичных окон.

Подчиненное вторичное окно всегда появляется поверх своего первичного окна, перекрывая также связанные с ним другие вторичные окна, открытые ранее. Когда вновь активизируется одно из этих окон, открытое последним вторичное окно может им перекрываться, оставаясь при этом поверх своего первичного окна.

Вы можете определить вторичное окно таким образом, чтобы оно всегда появлялось поверх других вторичных окон того же уровня. Для такого окна должно быть установлено свойство *Всегда сверху* (*Always On Top*). Как правило, такая техника используется только для окна палитры, но даже и в этой ситуации следует дать пользователю возможность самому управлять данным свойством окна. Если свойство *Всегда сверху* установлено для нескольких вторичных окон, то они размещаются на экране по правилам, общим для всех вторичных окон того же уровня. Не используйте свойство *Всегда сверху*, когда пользователь работает с окном приложения, для которого установка такого свойства невозможна.

Когда пользователь открывает вторичное окно, отображайте в нем информацию с учетом текущей ситуации (контекста). Например, при открытии панели свойств на ней должен быть представлен текущий выбор (то есть текущие значения свойств объекта). В общем случае старайтесь отобразить открываемое окно в том же состоянии, в котором его оставил пользователь. Например, диалоговая панель *Открыть* должна сохранить текущий каталог, установленный перед закрытием этого окна. Такой подход облегчает пользователю повторное выполнение операций, связанных с данным окном. Тем не менее, если команда или задание подразумевает или требует, чтобы пользователь выполнял процесс в определенной последовательности (например, при использовании Мастера), вторичное окно должно быть представлено в соответствующем фиксированном виде.

ДОПОЛНЕНИЕ ВТОРИЧНОГО ОКНА

За исключением окна палитры, избегайте создания вторичных окон с изменяемыми размерами, поскольку любое вторичное окно предназначено для отображения конкретной предопределенной информации. Вместе с тем, в некоторых случаях может возникнуть необходимость последовательного уточнения или дополнения отображаемой в окне информации; в таком окне может использоваться специальная кнопка — *Дополнить (Unfold)*, позволяющая отобразить в окне дополнительные параметры. Кнопка *Дополнить* может обозначаться двумя символами «больше чем» (»»). При нажатии указанной кнопки вторичное окно отображается в альтернативной, расширенной форме, имеющей фиксированный размер. Для возврата в исходное: состояние альтернативная форма окна может содержать соответствующую кнопку (*Refold*).

КАСКАДИРОВАНИЕ ВТОРИЧНОГО ОКНА

Пользователю также может быть предоставлен доступ к дополнительным параметрам (или информации) посредством использования во вторичном окне кнопок, открывающих дополнительные вторичные окна. Если открываемое окно является независимым, закройте вторичное окно, из которого пользователь открыл его, и оставьте на экране только новое окно. Если же новое окно предоставляет информацию, необходимую для ввода данных в исходном вторичном окне, то отобразите оба окна на экране одновременно; при этом подчиненное окно должно появляться немного правее и ниже исходного вторичного окна. Чтобы избежать хаотичного расположения на экране дополнительных вторичных окон, ограничьте их количество минимально необходимым числом.

РАЗМЕЩЕНИЕ ВТОРИЧНОГО ОКНА

При выборе расположения вторичного окна на экране следует учитывать большое число факторов, в том числе назначение окна, причину его появления, размеры экрана монитора и т.д. Как правило, вторичное окно следует отображать в той позиции, где оно появлялось в последний раз. При первом

открытии окна установите его в позиции, удобной для работы пользователя, причем окно обязательно должно отображаться полностью. Во многих случаях удобнее всего располагать вторичное окно таким образом, чтобы оно находилось в центре окна по горизонтали и ниже заголовка окна, меню и всех пристыкованных к верхнему краю окна панелей инструментов.

При создании вторичного окна может быть назначена специальная так называемая **предопределенная кнопка**, которая активизируется по нажатию клавиши <Enter>. Система отличает предопределенную кнопку от других командных кнопок по утолщенной границе, которая отображается вокруг кнопки (рис. 4.18).

В качестве предопределенной рекомендуется назначать такую кнопку, которая связана с наиболее вероятным действием пользователя, например, с подтверждением выбора, установленного по умолчанию. Избегайте использовать в качестве предопределенной кнопку, связанную с действием, которое нельзя отменить или которое может привести к разрушительным последствиям. Например, при обработке текста опасно использовать во вторичном окне *Найти и Заменить* в качестве предопределенной кнопку *Заменить везде*.

Рис. 4.18. Предопределённая кнопка

В качестве элемента настройки интерфейса предопределенная кнопка может динамически изменяться в процессе взаимодействия пользователя с окном. Например, если пользователь выбирает кнопку, которая изначально не является предопределенной, новая кнопка временно становится таковой; при этом она принимает внешний вид предопределенной кнопки (вместо исходной). Если же пользователь перемещает фокус ввода в пределах окна на другой элемент интерфейса, не являющийся кнопкой, в качестве предопределенной восстанавливается исходная кнопка.

Если в окне имеется другой элемент интерфейса, требующий управления с помощью клавиши <Enter> (например, многострочное текстовое поле), то вы не можете назначить для такого окна предопределенную кнопку. Кроме того, когда фокус ввода установлен на какой-либо другой элемент интерфейса, и это требует использования клавиши <Enter>, предопределенная кнопка временно перестает быть таковой.

В качестве средства ускоренного выбора предопределенной кнопки и одновременного выполнения связанной с ней команды, для нее может поддерживаться двойной щелчок ЛКМ.

НАВИГАЦИЯ ВО ВТОРИЧНОМ ОКНЕ

Навигация во вторичном окне с помощью мыши полностью идентична навигации в первичном окне.

Клавиатурный интерфейс для вторичных окон основан на использовании клавиши <Tab> и комбинации <Shift>+<Tab>, при нажатии которых фокус ввода перемещается между соседними элементами интерфейса (на следующий и предшествующий соответственно). Каждый элемент интерфейса имеет атрибут, который определяет для него порядок навигации. Установите значение этого атрибута таким образом, чтобы пользователь мог перемещаться по вторичному окну в естественном при чтении порядке: слева направо и сверху вниз; соответственно, в исходном состоянии фокус ввода должен быть установлен на элементе, расположенном в верхней левой области окна. Вместе с тем, порядок перемещения по окну должен также учитывать наличие логически связанных групп элементов интерфейса. Например, если окно содержит группу переключателей, выстроенных вертикально, то фокус ввода должен последовательно пройти сверху вниз через всю группу, и только затем переместиться на элемент, расположенный правее. Кнопки, реализующие команды, общие для всего окна, располагаются, как правило, последними (в нижней или правой нижней области окна).

Совсем не обязательно назначать клавишу-акселератор для каждого элемента вторичного окна. Для тех же элементов, для которых он кажется разработчику полезным, клавиатурный доступ может быть реализован через статические текстовые области.

При реализации клавиатурной навигации могут дополнительно использоваться клавиши управления курсором (например, клавиши <Вверх> и <Вниз> обеспечивают переход между строками текстового поля или между переключателями внутри группы). Всегда используйте клавиши управления курсором для перехода между флажками и элементами списка.

Немодифицируемые текстовые клавиши также поддерживают навигацию, если элемент интерфейса, на который в данный момент установлен фокус ввода, не использует эти клавиши для ввода информации. Например, если фокус ввода установлен в текстовом поле или списке, текстовая клавиша не может быть использована в качестве средства управления для этого элемента.

Клавиши доступа не только позволяют пользователю переходить от одного элемента к другому, они обеспечивают тот же эффект, что и щелчок ЛКМ на выбранном элементе. Так, если клавиша доступа определена для кнопки, то нажатие клавиши приводит к выполнению команды, связанной с этой кнопкой.

Клавиши доступа могут быть определены таким образом, чтобы обеспечить возврат фокуса ввода на логически связанный элемент интерфейса. Например, если при работе со списком пользователь использует клавишу для доступа к кнопке, которая модифицирует содержимое списка, то после модификации фокус ввода может быть автоматически возвращен в список.

Для кнопок *ОК* и *Отменить* обычно не назначаются клавиши доступа, если они являются основными кнопками для данного вторичного окна. В этом случае для доступа к ним по умолчанию используются клавиши соответственно <Enter> и <Esc>.

Нажатие клавиши <Enter> всегда приводит к выбору предопределенной кнопки (если она существует) и выполнению связанного с ней действия. Если

предопределенная кнопка отсутствует, то клавиша <Enter> может использоваться разработчиком по его усмотрению.

ПОДТВЕРЖДЕНИЕ КОРРЕКТНОСТИ ВВОДА

Всегда подтверждайте корректность ввода данных пользователем во вторичном окне, причем с минимально возможной задержкой. В идеальном случае корректность ввода

должна подтверждаться сразу после окончания ввода. При обнаружении ошибки можно либо запретить ввод, либо использовать звуковую и визуальную обратную связь, чтобы предупредить пользователя об ошибке. Вы можете также вывести на экран окно сообщения, содержащее пояснения по возникшей ситуации, особенно если пользователь многократно пытается повторить неправильный ввод. Вероятность ошибок пользователя может быть снижена за счет использования элементов интерфейса, которые ограничивают возможный выбор пользователя в конкретной ситуации (к таким элементам относятся, например, переключатели и списки), либо путем инициализации области ввода значением, установленным по умолчанию.

Если невозможно подтвердить правильность ввода непосредственно после его завершения, следует предусмотреть способ подтверждения действий пользователя по окончании его работы с данным вторичным окном. Если и это затруднительно, то подтверждайте корректность ввода всякий раз, когда пользователь пытается закрыть окно. В случае ошибки оставьте окно открытым и выведите соответствующее сообщение; после того, как пользователь его закроет, установите фокус ввода на том элементе, который содержит неподходящие данные.

4.3.2. ПАНЕЛИ СВОЙСТВ И КОНТРОЛЯ ПАРАМЕТРОВ

Свойства объекта могут быть представлены с точки зрения интерфейса пользователя несколькими способами. В частности, изображение и имя пиктограммы объекта на Рабочем столе отражают специфические свойства соответствующего объекта. Для отображения свойств объектов могут использоваться и другие элементы интерфейса, например, панель инструментов, строка состояния, или даже полосы прокрутки. Наиболее же универсальное средство представления свойств объекта — это вторичное окно **панель свойств (Property Sheet)**. Панель свойств является независимым вторичным окном, которое отображает доступные пользователю (т.е. видимые) свойства объекта, причем не обязательно пользователю должно быть предоставлено право изменять их. Панель свойств отображается на экране по команде *Свойства* для конкретного (выбранного) объекта.

Панель контроля параметров (Property Inspector) отличается от панели свойств тем, что она реализуется, как правило, в виде модального вторичного окна, связанного с тем объектом, свойства которого она отображает. Если пользователь выбирает другой объект, панель свойств продолжает отображать свойства исходного объекта. В отличие от нее, панель контроля параметров всегда относится к выбранному в данный момент объекту.

ПАНЕЛЬ СВОЙСТВ

Заголовок окна панели свойств должен идентифицировать тот объект, к которому она относится. Если объект имеет имя, оно может быть использовано в качестве заголовка окна либо непосредственно, либо в сочетании со словом *Свойства*. Если словосочетание *Свойства* <имя объекта> превышает ширину полосы заголовка окна, система исключает имя объекта и добавляет эллипсис. Если объект не имеет собственного имени, используйте в заголовке окна имя типа объекта. Для панели свойств, представляющей свойства нескольких выбранных объектов, в качестве названия также используется имя типа этих объектов. Когда такой подход не применим (например, выбор содержит разнотипные объекты), укажите в заголовке, что панель содержит перечень свойств выбранных объектов (например, *Свойства выбора*).

Поскольку может потребоваться отобразить на панели свойств большое количество атрибутов объекта и текущей ситуации, целесообразно разбить их на относительно самостоятельные группы в пределах окна панели. Существует два способа решения этой задачи.

Первый из них — разбиение панели свойств на отдельные страницы определенного формата, обычно называемые **вкладками**. Каждая вкладка имеет собственное имя (обозначение), отображаемое на этикетке вкладки. Пользователю всегда видны этикетки всех вкладок, однако, в каждый момент времени может быть открыта только одна из них (рис. 4.19).

Рис. 4.19. Панель свойств, разделённая на вкладки

Второй подход заключается в том, что перечень отображаемых в окне свойств может динамически корректироваться в соответствии с текущей ситуацией или с учетом иерархической взаимосвязи некоторых объектов.

Например, если пользователь работает с текстом, на панели свойств могут отображаться свойства отдельного параграфа. Аналогично, если пользователь выбирает ячейку в электронной таблице, ему на той же панели может быть предоставлен доступ к набору свойств строки и столбца. В подобных ситуациях в состав панели свойств целесообразно включить выпадающий список, отражающий перечень объектов, для которых на данной панели может быть получен набор свойств (рис. 4.20).

Рис. 4.20. Применение списка для доступа к иерархически связанным наборам свойств

Где это, возможно, следует делать значения свойств, отображаемых на панели, редактируемыми. Вы можете реализовать специальный набор команд редактирования, обеспечивающих копирование свойств одного объекта для другого объекта. Например, может быть реализовано воспроизведение свойств для объектов, которые являются элементами одного списка.

В большинстве случаев панель свойств позволяет пользователю изменять значения тех или иных свойств объекта. Для обработки приложением внесенных пользователем изменений следует включить в окно панели свойств следующие кнопки (табл. 4.3).

Таблица 4.3.

Стандартные кнопки панели свойств

Кнопка (команда)	<i>Действие</i>
ОК	Передаёт приложению все внесённые изменения и закрывает окно панели свойств
Применить (Apply)	Передаёт приложению все внесённые изменения, но оставляет окно панели свойств открытым
Отменить (Cancel)	Отменяет все внесённые (но не переданные приложению) изменения и закрывает окно панели свойств. Не отменяет изменения, которые уже были реализованы

Дополнительно может использоваться команда *Сбросить (Reset)* для отмены внесенных изменений без закрытия окна.

Как было указано выше, при наличии большого числа параметров панель свойств может быть разделена на несколько вкладок. Поэтому расположение кнопок на панели свойств имеет большое значение. Если кнопка размещается на одной из вкладок, связанное с ней действие должно отображаться в пределах этой же вкладки. Если же кнопка установлена таким образом, что она не входит ни в одну из вкладок панели, связанное с ней действие должно распространяться на все окно панели.

Кнопки, являющиеся общими для панели свойств — *ОК*, *Отменить*, *Применить* — лучше всего установить за пределами вкладок, поскольку пользователи рассматривают вкладки просто как форму группирования элементов интерфейса. Это означает, что если пользователь вносит изменение на одной вкладке, то оно не реализуется, когда пользователь переходит на другую вкладку. Тем не менее, если пользователь внес изменение на новой вкладке и затем нажимает кнопки *ОК* или *Применить*, то реализуются изменения, сделанные на обеих вкладках (при нажатии кнопки *Отменить* изменения игнорируются на обеих вкладках).

Если создаваемое приложение требует, чтобы пользователь устанавливал значения свойств на вкладках последовательно, «страница за страницей», то следует размещать кнопки *ОК*, *Отменить*, *Применить* на каждой вкладке в одной и той же позиции. Когда пользователь переходит на следующую вкладку, внесенные изменения могут сохраняться автоматически, либо пользователю может выдаваться сообщение с просьбой подтвердить или отменить сделанные изменения.

В некоторых случаях в панель свойств целесообразно поместить образец, иллюстрирующий эффект вносимых изменений. Например, если пользователь выбирает фрагмент текста и открывает для него панель свойств, этот фрагмент может быть помещен в панель свойств в качестве образца (рис. 4.21).

При внесении изменений образец должен отображать соответствующее состояние объекта.

Рис. 4.21. Использование образца в панели свойств

При закрытии пользователем окна панели свойств, приложение должно вести себя также, как и при закрытии окна содержимого объекта (например документа). При этом не рекомендуется интерпретировать кнопку *Закрыть* как кнопку *Отменить*. Если в окне имеются изменения, которые не были сохранены, напомните об этом пользователю, используя окно сообщения, как показано на рис. 4.22. Если же таких изменений нет, просто закройте окно.

Рис. 4.22. Окно сообщения для подтверждения внесенных изменений

Если пользователь выбирает кнопку *Да*, то внесенные изменения сохраняются, и окна сообщения и панели свойств закрываются. Если же пользователь выбирает кнопку *Нет*, то оба окна закрываются без сохранения изменений. Чтобы позволить пользователю отменять закрытие окна панели свойств, включите в окно сообщения соответствующую кнопку (*Отменить*).

ПАНЕЛЬ КОНТРОЛЯ ПАРАМЕТРОВ

Вы можете также отобразить свойства объекта, используя средства динамического просмотра — *броузер*, который выводит на экран свойства выбранного объекта. Такое окно называется *панель контроля параметров*. В качестве основы для создания панели контроля параметров обычно используют окно панели инструментов или окно палитры, как показано на рис. 4.23.

Рис.4.23. Панели контроля параметров

Изменения значений свойств объекта, которые пользователь вносит в панель контроля параметров, динамически заменяют предыдущие значения. При этом внесенные изменения должны сразу же применяться к выбранному объекту.

Панель контроля параметров и панель свойств не являются взаимоисключающими элементами интерфейса; в приложении могут использоваться оба варианта, поскольку каждый из них имеет свои преимущества. Наиболее важные или часто корректируемые свойства целесообразно отображать с помощью панели контроля параметров, а полный их перечень может быть реализован в виде панели свойств. В одном приложении может использоваться несколько разных панелей контроля параметров, каждая из которых предназначена для объектов определенного типа.

В качестве дополнительного средства настройки интерфейса вы можете предоставить пользователю право изменять форму взаимодействия с панелью контроля параметров и панелью свойств. В частности, вы можете обеспечить такое управление панелью контроля параметров, когда фиксируется некоторый выбранный пользователем набор значений свойств объекта, а последующие вносимые изменения игнорируются.

ОТОБРАЖЕНИЕ СВОЙСТВ ДЛЯ МНОЖЕСТВЕННОГО И ДРУГИХ СЛОЖНЫХ ВИДОВ ВЫБОРА

Когда пользователь выбирает несколько объектов и запрашивает их свойства, используйте для их представления одну общую панель свойств (или панель контроля параметров), вместо того, чтобы открывать отдельное окно для каждого объекта. Если некоторые свойства имеют у нескольких объектов разные значения, отобразите соответствующие им элементы интерфейса в состоянии

«неопределено». Вместе с тем, следует сохранить возможность вывода на экран нескольких панелей свойств одновременно, если пользователь пожелает получить их для каждого объекта в отдельности. Такой подход обеспечивает требуемую гибкость интерфейса. Если пользователю при работе с приложением может потребоваться доступ к отдельным свойствам на панели свойств множественного выбора, включите и нее список (или выпадающий список): это позволит пользователю указать объект, для которого производится установка нового значения.

Если множественный выбор содержит объекты различного типа, отобразите на панели только те свойства, которые являются общими для всех объектов.

Отображая свойства нескольких объектов, учитывайте различие между множественным выбором и группой (композицией) объектов, поскольку группа отражает более тесную связь между объектами. Результат группирования (*агрегат*) может сам считаться новым объектом, со своими собственными свойствами и допустимыми операциями. Следовательно, когда пользователь запрашивает свойства группы объектов, отобразите свойства такого составного объекта. При этом свойства отдельных членов группы могут включаться или не включаться в панель свойств в зависимости от конкретной ситуации.

4.3.3. ДИАЛоговые ПАНЕЛИ

Диалоговая панель (Dialog box) обеспечивает обмен информацией или ведение диалога между пользователем и приложением. Как правило, она используется для получения от пользователя дополнительной информации, необходимой для выполнения некоторой команды или задания (в этом ее основное отличие от панели свойств).

Поскольку диалоговые панели обычно появляются после выбора какого-либо пункта меню (в том числе всплывающих и каскадных меню) или «нажатия» кнопки, название диалоговой панели должно отражать имя связанной с ней команды. Не рекомендуется использовать эллипсис в названии окна, даже если он содержится в имени команды.

Подобно панели свойств, диалоговые панели обычно содержат кнопки *ОК* и *Отменить*. Кнопка *ОК* используется для того, чтобы передать приложению введенные в диалоговой панели данные и закрыть окно. Если пользователь выбирает кнопку *Отменить*, внесенные изменения игнорируются и окно закрывается. Как правило, кнопка *ОК* назначается в качестве предопределенной кнопки.

Диалоговая панель может содержать и другие кнопки, дополняющие или заменяющие кнопки *ОК* и *Отменить*. Названия используемых кнопок должны как можно более точно отражать их назначение, но быть краткими. Длинные, многословные названия затрудняют пользователю быстрое восприятие структуры и предназначения диалоговой панели. Должны соблюдаться и другие соглашения относительно использования кнопок, рассмотренные ранее.

КОМПОНОВКА ЭЛЕМЕНТОВ ДИАЛОГОВОЙ ПАНЕЛИ

При размещении элементов диалоговой панели следует учитывать привычное для людей направление чтения: слева направо и сверху вниз. Поэтому лучше всего расположить тот элемент, с которого пользователь начнет работу, как можно ближе к верхнему левому углу панели. Если диалоговая панель содержит группы взаимосвязанных элементов, то при размещении элементов внутри группы следует придерживаться тех же принципов.

Наиболее важные кнопки лучше всего сгруппировать и разместить сверху вдоль правой границы диалоговой панели, либо выстроить в ряд в нижней части окна. Одна из них может быть назначена предопределенной кнопкой. Если используются кнопки *ОК* и *Отменить*, сгруппируйте их вместе. Конечно, может использоваться и другая организация панели, если на то есть убедительная причина, такая, например, как привычное для пользователя размещение реальных объектов. Так, если на панели присутствуют кнопки, соответствующие направлениям сторон света (Север, Юг, Восток, и Запад), то имеет смысл разместить их в том же порядке, как на реальном компасе. Аналогично, кнопка, связанная с каким-либо элементом интерфейса другого типа, может располагаться рядом с этим элементом. Вместе с тем, старайтесь не использовать такие кнопки в качестве предопределенных.

Система поддерживает использование стандартных диалоговых панелей для выполнения наиболее распространенных операций. Включение этих панелей в создаваемое приложение может существенно сократить затраты времени на разработку при обеспечении высокой степени согласованности интерфейса. Если стандартная диалоговая панель модифицируется по заказу пользователя, или для приложения создается собственная диалоговая панель, они должны быть согласованы со стандартными диалоговыми панелями. Например, если вы создаете собственную панель для управления шрифтами, она должна быть аналогична по внешнему виду стандартной системной диалоговой панели *ШРИФТ*. Подобная согласованность интерфейса, как уже отмечалось, способствует применению пользователями имеющихся у них знаний и навыков при освоении новых программных продуктов. В связи с этим ниже приведено описание наиболее распространенных системных диалоговых панелей, которые могут быть использованы при создании различных приложений.

ДИАЛОГОВАЯ ПАНЕЛЬ *ОТКРЫТЬ* (*OPEN*)

Диалоговая панель *ОТКРЫТЬ* позволяет пользователю просматривать содержимое файловой системы, включая просмотр сети, и открывать определенный файл (рис. 4.24). Данная диалоговая панель отображается на экране по команде *Открыть* из меню *Файл* или с помощью одноименной кнопки панели инструментов.

Системные диалоговые панели автоматически поддерживают использование длинных имен файлов, операции прямого манипулирования, а также всплывающие меню для пиктограмм. Диалоговая панель *ОТКРЫТЬ* позволяет отображать расширения имен файлов только для зарегистрированных типов файлов.

Чтобы открыть файл, пользователь выбирает файл из списка в диалоговой панели, или заносит имя файла в текстовое поле *Имя файла* и затем выбирает команду *Открыть*. Пользователь может также вызвать всплывающее меню для файла и выбрать

Рис. 4.24. Диалоговая панель *ОТКРЫТЬ*

в нем команду *Открыть*. В качестве ускоренного способа открытия файла используется двойной щелчок ЛКМ на его имени. Выбор кнопки *Отменить* приводит к закрытию окна без открытия файла.

Когда пользователь выбирает какую-либо пиктограмму и дважды щелкает па ней ЛКМ, открывается связанный с ней файл. Другими словами, это дает такой же эффект, как если бы пользователь непосредственно открыл файл.

Список файлов, представленный в диалоговой панели, определяется текущим каталогом и типом фильтра, установленного в выпадающем списке *Тип файла*.

Текущий каталог отображается в текстовом поле выпадающего списка *Папка*. Открыв этот список, пользователь может просмотреть иерархию файловой системы и перемещаться по дереву каталогов. Диалоговая панель поддерживает также возможность выбора формы представления списка файлов и некоторые другие функции.

Текущий каталог устанавливается по умолчанию исходя из контекста. Если пользователь открывает файл, находящийся в другом каталоге, в окне отображается маршрут доступа к нему. После запуска приложения оно может само установить маршрут, наиболее удобный для работы пользователя с его файлами.

Для изменения текущего каталога пользователь либо выбирает соответствующий пункт в списке маршрутов доступа, либо вводит требуемый маршрут в текстовое поле *Имя файла* и нажимает кнопку *Открыть*. Выбор пользователем кнопки *Отменить* не должен приводить к изменению маршрута. Всегда сохраняйте самый последний маршрут доступа между последовательными открытиями диалоговой панели. Если приложение поддерживает открытие нескольких файлов, подобно тому, как это реализовано в MDI, установите маршрут, соответствующий местоположению файла,

открытого последним, хотя бы соответствующее ему дочернее окно и не являлось на этот момент активным. Если имеется несколько работающих экземпляров приложения, то установка текущего каталога должна выполняться отдельно для каждого из них

При запуске приложения оно устанавливает в диалоговой панели определенный тип файла для фильтра *Тип файла*. Это может быть либо тип файла, открытого последним, либо тип файла, последним установленный пользователем, либо определенный тип файла, соответствующий специфике приложения.

Пользователь может изменить настройку фильтра, выбрав другой тип в выпадающем списке *Тип файла*, или набрав его в текстовом поле списка и нажав кнопку *Открыть*. Фильтр может содержать расширение имени файла. Например, если пользователь заносит *.txt и выбирает кнопку *Открыть*, в списке отображаются только файлы с расширением .txt. Занесение расширения в это текстовое поле приводит также к изменению типа, установленного в выпадающем списке *Тип файла*. Если приложение не поддерживает работу с файлами данного типа, соответствующий ему элемент списка отображается в состоянии «не определено».

В список *Тип файла* следует включить как все зарегистрированные типы файлов, так и поддерживаемые вашим приложением. Для описания каждого пункта в списке рекомендуется использовать стандартные наименования.

Например, для текстовых файлов должен быть установлен тип *Текстовый документ (Text Document)*. Вы можете также включить пункт *Все файлы (All Files)*, чтобы отобразить все файлы, имеющиеся в текущем каталоге, независимо от их типа.

В тех случаях, когда пользователь вводит имя файла в текстовом поле *Имя файла* и выбирает кнопку *Открыть*, возможны различные варианты поведения как системы, так и приложения. Ниже приведено описание наиболее распространенных ситуаций.

- Если введено *имя файла без расширения*, то система пытается сначала отыскать файл среди файлов, имеющих тип вашего приложения, затем — среди файлов, имеющих тип, установленный на данный момент в списке *Тип файла*. Например, если пользователь вводит имя *Мой_документ*, и встроенное расширение приложения — .doc, то система пытается в первую очередь открыть файл *Мой_документ.doc*. Если же системе не удастся найти файл с таким именем и расширением, и этот тип не выбран в списке *Тип файла*, то система пытается открыть имеющийся в списке файл с тем же именем, независимо от расширения. Если таких файлов несколько, то система выводит соответствующее сообщение.

- Если имя файла введено *с расширением*, то система сначала проверяет, совпадает ли оно с встроенным типом приложения, либо с любым другим из числа зарегистрированных или имеющихся в списке *Тип файла*. Если это не так, то система пытается открыть файл с таким же именем, но имеющий встроенный тип приложения или текущий тип, установленный в списке *Тип файла*. Если это оказывается невозможным, система действует таким же образом, как для файла без расширения, выполняя поиск среди всех файлов, которые представлены в списке.

- Если имя файла введено *в кавычках*, то система интерпретирует его как строку без кавычек и выполняет поиск таким же образом, как для имени файла без расширения.

- Если системе не удастся найти файл с указанным именем, она выводит соответствующее сообщение и советует пользователю проверить корректность ввода имени файла и маршрута доступа к нему. Тем не менее, вы можете выбрать другой способ обработки такой ситуации.

- Если имя файла содержит символы, запрещенные к использованию, система сообщает об этом пользователю.

Для диалоговой панели *ОТКРЫТЬ* система выполняет только сравнение имени файла. Ответственность за корректное использование файлов различных типов и за уведомление пользователя о несоответствии типа возлагается на приложение.

ДИАЛОГОВАЯ ПАНЕЛЬ СОХРАНИТЬ КАК (SAVE AS)

Диалоговая панель *СОХРАНИТЬ КАК* используется для того, чтобы пользователь мог сохранить файл, указав для него конкретное имя, местоположение и тип. Как правило, одноименная команда используется в тех приложениях, которые поддерживают создание файлов нескольких типов. Если же ваше приложение поддерживает только файлы одного определенного типа и создает их автоматически, в использовании такой диалоговой панели нет необходимости.

Данная диалоговая панель появляется на экране, когда пользователь выбирает команду *Сохранить как* (*Save as...*) или какую-либо другую команду с аналогичной функцией, например, *Экспорт файла* (*Export*). Рекомендуется отображать эту диалоговую панель также в тех случаях, когда пользователь выбирает команду *Сохранить*, не указав при этом (или не подтвердив) имя файла либо маршрут записи. Если ваше приложение использует панель **СОХРАНИТЬ КАК** при выполнении других команд, связанных с сохранением файлов, подберите для нее такое название, которое бы отражало сущность этих команд.

По внешнему виду и функционированию панель **СОХРАНИТЬ КАК** аналогична панели **ОТКРЫТЬ**, за исключением того, что выпадающий список *Тип файла* содержит только такие типы файлов, которые поддерживаются данным приложением;

эти же типы файлов используются в качестве фильтров при формировании списка файлов, отображаемого в окне панели (рис. 4.25).

Чтобы сохранить файл, пользователь выбирает кнопку *Сохранить* и сохраняет файл под именем, которое появляется в текстовом поле *Имя файла*. Хотя пользователь может ввести имя или выбрать файл из списка файлов, ваше приложение должно проинициализировать это текстовое поле именем текущего (открытого) файла. Если файл еще не имеет индивидуального имени, предложите для него стандартное имя для файлов данного типа, например, *Текстовый_документ(2)*.

В текстовом поле выпадающего списка *Сохранить в* (или *Папка:*) указывается текущий каталог (папка). Пользователь может изменить маршрут, введя его имя в текстовом поле или выбрав из списка. Если файл уже существует, всегда сохраняйте его в прежней позиции. Это означает, что для диалоговой панели **СОХРАНИТЬ**

Рис. 4.25. Диалоговая панель **СОХРАНИТЬ КАК**

КАК в качестве текущего маршрута должен всегда устанавливаться тот маршрут, по которому файл был сохранен последний раз. Если файл еще не сохранялся, предложите маршрут, используемый по умолчанию.

Если пользователь выбирает кнопку *Отменить*, не сохраняйте файл или другие установленные параметры и восстановите исходный маршрут.

Как уже было отмечено, выпадающий список *Тип Файла* содержит только такие типы *файлов*, которые поддерживаются данным приложением. В качестве дополнительной информации целесообразно включить в список описание форматов файлов как часть описания типа. В связи с этим необходимо сделать следующее замечание. Хотя формат файла может быть обусловлен его типом, формат и тип файла — это не одно и то же. Например, файл, содержащий растровое изображение, может быть сохранен в монохромном, 16,24 или 256-битовом цветном формате, но тип файла останется один тот же — *.bmp*. Как правило, описание элементов списка *Тип Файла* выглядит следующим образом:

Имя типа [Описание формата].

В свою очередь, в качестве описателя формата файла указывается его расширение, например

Шаблон документа [*.dot].

Когда пользователь вводит имя файла, поведение диалоговой панели *СОХРАНИТЬ КАК* подчиняется тем же правилам, которые были рассмотрены выше применительно к панели *ОТКРЫТЬ*. Если пользователь не указывает расширение, система использует тип файла, выбранный в списке *Тип файла* или установленный по умолчанию вашим приложением. Если пользователь вводит расширение, то система сравнивает его с расширением файлов приложения и зарегистрированными расширениями, имеющимися в списке *Тип файла*. Если введенное расширение оказывается одним из них, система сохраняет файл с соответствующим расширением (расширение остается скрытым, если параметры системы не установлены иным образом). В противном случае система интерпретирует введенное пользователем расширение как часть имени файла и Добавляет к нему расширение, установленное в списке *Тип файла*. Имейте в виду, это не означает, что расширение файла соответствует его истинному типу. Ответственность за такое соответствие возлагается на ваше приложение.

Убедитесь, что вы обеспечиваете сохранение даты создания файлов, которые пользователь перезаписывает после просмотра или редактирования. Если ваше приложение сохраняет файлы, создавая временный файл, то удаляйте подлинник, присваивая временному файлу имя оригинала; при этом очередная копия должна иметь дату создания исходного файла. Это объясняется тем, что управление некоторыми файловыми системами может зависеть от сохранения тождества исходного файла.

Если введенное пользователем имя файла заключено в кавычки, то система сохраняет файл, не добавляя к имени расширение. Если введено зарегистрированное расширение, файл сохраняется как файл соответствующего типа. Если при вводе имени файла пользователь указывает неправильный символ или указанный маршрут не существует, и приложение в таких условиях может работать некорректно, система выводит соответствующее сообщение.

Таблица 4.4.

Интерпретация системой имени файла

Имя, введенное пользователем	Имя сохраняемого файла	Комментарий
My File	My File.txt	Файлу присваивается расширение, используемое по умолчанию, или установленное в списке <i>Тип файла</i>
My File.txt	My File.txt	Файл сохраняется с тем же расширением, поскольку оно соответствует используемому по умолчанию
My File for T.Book	My File for T.Book.txt	Расширение Book не является зарегистрированным или включенным в список типов файлов, поэтому имя файла дополняется расширением, используемым приложением по умолчанию
«My File»	My File	Тип файла не задан; в качестве имени файла используется последовательность символов внутри кавычек

«My» File.	Файл не сохраняется	Система (или приложение) выводит на экран сообщение, уведомляющее пользователя о том, что имя файла содержит ошибку
------------	---------------------	---

В табл. 4.4 приведены примеры того, как система интерпретирует введенное пользователем имя файла (предполагается, что файлы приложения должны иметь расширение *.txt).

ДИАЛОГОВЫЕ ПАНЕЛИ НАЙТИ (FIND) И ЗАМЕНИТЬ (REPLACE)

Диалоговые панели *НАЙТИ* и *ЗАМЕНИТЬ* обеспечивают поиск заданной текстовой строки (последовательности символов) и замену ее другой текстовой строкой (последовательностью символов), определенной пользователем. Эти диалоговые панели показаны на рис. 4.26.

Рис. 4.26. Диалоговые панели *НАЙТИ* и *ЗАМЕНИТЬ*

ДИАЛОГОВАЯ ПАНЕЛЬ ПЕЧАТЬ (PRINT)

Диалоговая панель *ПЕЧАТЬ*, показанная на рис. 4.27, позволяет пользователю указать, что печатать, количество копий и последовательность печати, а также выбрать принтер. Кроме того, панель содержит кнопку, которая предоставляет пользователю быстрый доступ к панели свойств принтера.

Рис. 4.27 Диалоговая панель *ПЕЧАТЬ*

ДИАЛОГОВАЯ ПАНЕЛЬ МАКЕТ СТРАНИЦЫ (PAGE SETUP)

Данная диалоговая панель обеспечивает возможность выбора формата, ориентации и источника бумаги, а также других параметров выводимого на печать документа (рис. 4.28).

В общем случае значения параметров страницы, установленные на данной панели, отменяют значения одноименных параметров, установленных для принтера. Однако время их действия ограничено временем печати этой страницы или документа.

Кнопка *Принтер* обеспечивает вывод на экран дополнительной диалоговой панели (рис. 4.29), которая содержит информацию о текущем (используемом по умолчанию) принтере.

Рис. 4.28. Диалоговая панель *МАКЕТ СТРАНИЦЫ*

Рис. 4.29. Дополнительная диалоговая панель *ПРИНТЕР*

ДИАЛоговая панель *ШРИФТ (FONT)*

Эта диалоговая панель отображает список доступных шрифтов, их возможный размер и начертание (рис. 4.30). Приложение может скорректировать этот список, чтобы показать только те шрифты, которые используются при работе с ним.

Рис. 4.30. Диалоговая панель *ШРИФТ*

ДИАЛоговая панель *ЦВЕТ (COLOR)*

Диалоговая панель ЦВЕТ позволяет пользователю выбрать требуемый цвет из набора доступных (рис. 4.31). Перечень объектов, для которых пользователь может устанавливать собственный цвет, определяется приложением.

Верхняя часть окна (*Базовая палитра — Basic colors*) содержит набор цветов, используемых по умолчанию. Их перечень определяется установленным драйвером монитора. Другая часть окна — *Дополнительные цвета (Custom colors)* — предоставляет пользователю возможность расширить цветовую гамму, используя различные оттенки, приведенные справа в дополнительном окне (рис. 4.32); оно открывается, когда пользователь нажимает кнопку *Определить цвет (Define Custom Colors)*.

Рис. 4.31. Диалоговая панель *ЦВЕТ* (сокращенный формат)

Рис. 4.32. Диалоговая панель *ЦВЕТ* (расширенный формат)

4.3.4. ДРУГИЕ ТИПЫ ВТОРИЧНЫХ ОКОН

ОКНО *ПАЛИТРА (PALETTE)*

Окно *ПАЛИТРА* является независимым вторичным окном, которое содержит набор взаимосвязанных элементов управления. Например, в виде такого окна может быть представлена панель инструментов или отдельная ее часть (рис. 4.33). Каждое окно *ПАЛИТРА* может иметь индивидуальное название, отображаемое в полосе заголовка окна, и собственный формат. Полоса заголовка окна *ПАЛИТРА* содержит только одну кнопку — *Заккрыть*.

Название окна должно отражать его назначение. Размер и шрифт для полосы заголовка окна *ПАЛИТРА* устанавливаются системой по умолчанию, но их значения могут быть изменены разработчиком приложения. Размер самого окна может быть как фиксированным, так и регулируемым пользователем (последний вариант является более предпочтительным). На возможность изменения размеров окна указывают два визуальных признака: изменение формы указателя, когда он находится на границе окна, и наличие команды *Размер* во всплывающем меню окна. Когда пользователь закрывает окно, запомните его размер и позицию на экране, чтобы оно могло быть восстановлено в прежнем виде при последующем открытии. Окно *ПАЛИТРА* закрывается автоматически, когда закрывается связанное с ним первичное окно.

Как и в других окнах, щелчок ПКМ в полосе заголовка окна обеспечивает появление на экране всплывающего меню окна. Меню может содержать

следующие команды: *Заккрыть*, *Переместить*, *Размер* (если окно является масштабируемым), *Всегда сверху* и *Свойства* (рис. 4.34).

Команда *Свойства* обеспечивает доступ пользователя к панели свойств окна *ПАЛИТРА*, что, в свою очередь, позволяет ему изменять некоторые свойства окна (в частности, свойство *Всегда сверху*), или модифицировать содержимое окна (рис. 4.35).

Рис. 4.33. Окно *ПАЛИТРА*

Рис. 4.34. Всплывающее меню для окна *ПАЛИТРА*

Рис. 4.35. Панель свойств для настройки окна *ПАЛИТРА*

ОКНО СООБЩЕНИЕ (MESSAGE BOX)

Окно *СООБЩЕНИЕ* является вторичным окном, предназначенным для вывода на экран сообщений пользователю; обычно это информация о конкретной ситуации или условиях выполнения операций. Как правило, окна сообщений содержат графический символ, который указывает на тип выводимого сообщения, и собственно текст сообщения (рис. 4.36).

Сообщения являются важной компонентой поддержки пользователя и могут быть реализованы в нескольких форматах. Более подробно особенности их проектирования рассмотрены в разделе «Проектирование средств поддержки пользователя».

Рис. 4.36. Окно *СООБЩЕНИЕ*

ВСПЛЫВАЮЩИЕ ОКНА (POP-UP WINDOWS)

Всплывающие окна используются для отображения дополнительной информации в тех случаях, когда в основном окне она представлена в сокращенной форме. Например, всплывающее окно может использоваться, чтобы показать пользователю полный маршрут доступа, когда он не помещается полностью в отведенной области, либо какую-то другую текстовую информацию (рис. 4.37).

Рис. 4.37. Использование всплывающего окна для «увеличения» текстовой области

Всплывающие окна используются также для вывода контекстно-зависимой справочной информации, как показано на рис. 4.38.

Рис. 4.38 Всплывающее окно контекстно-зависимой помощи

Другим вариантом всплывающего окна, используемого для вывода контекстной информации, является *всплывающая подсказка (Tooltip)*; основное ее назначение- вывод пояснений для элементов управления, расположенных на панели инструментов.

Выбор средств доступа пользователя к всплывающим окнам зависит от особенностей их использования в приложении, но обычно такими средствами являются либо установка указателя на интересующий элемент (*указание*), либо щелчок ЛКМ либо вызов окна с помощью соответствующей команды. Если в качестве средства вызова всплывающего окна используется указание, окно должно появляться на экране после некоторой задержки. Система автоматически управляет задержкой для стандартных окон всплывающей подсказки. Если же в приложении реализован собственный вариант такого окна, то для его вызова и

удаления может применяться двойной щелчок ЛКМ. В этом случае необходимо изменять форму указателя, когда пользователь устанавливает его на интересующий элемент (тем самым обеспечивается обратная связь с пользователем). Клавиатурный доступ к всплывающей подсказке может быть реализован посредством клавиши <Spacebar>.

5. ПРОЕКТИРОВАНИЕ ЭЛЕМЕНТОВ УПРАВЛЕНИЯ

Под элементами управления обычно понимаются компоненты графического интерфейса, которые предоставляют пользователю возможность изменять содержимое или форму представления отображаемой информации, а также управлять работой приложения. К элементам управления относятся, в частности, списки, полосы прокрутки, кнопки и т.д.

Каждый элемент управления имеет уникальный образ и обеспечивает определенную форму взаимодействия пользователя с приложением. Система также поддерживает возможность создания собственных элементов управления. Определяя такие элементы, следует учитывать существующие системные соглашения, принятые для стандартных элементов управления.

Подобно большинству элементов интерфейса, элементы управления обеспечивают обратную связь с пользователем, изменяя определенным образом свой внешний вид в тех случаях, когда они активизированы. Например, когда пользователь взаимодействует с элементами управления, использующими мышь, каждый из них «реагирует» на выбор при нажатии кнопки мыши, но не активизируется до тех пор, пока пользователь не отпустит кнопку. Если пользователь перемещает указатель за пределы элемента управления при нажатой кнопке мыши, он перестает воспринимать входное воздействие. Если пользователь возвращается на элемент управления, тот снова переходит в активное состояние. *Горячая зона*, которая определяет, будет ли реагировать элемент управления на указатель, зависит от типа элемента. Для некоторых элементов управления, таких, например, как кнопки, горячая зона совпадает с видимой границей элемента. Для других горячая зона может включать графический символ элемента управления и относящуюся к нему текстовую область (например, это справедливо для флажков и переключателей).

Для большинства элементов управления система обеспечивает вывод текстовой подсказки. Поскольку подсказка помогает пользователю определить предназначение данного элемента управления, всегда идентифицируйте элемент, с которым в данный момент взаимодействует пользователь. Если элемент не имеет подсказки, ее можно реализовать в виде статической текстовой области или в виде всплывающей подсказки (tooltip).

Поскольку некоторые из элементов управления могут обеспечивать какие-либо специфические способы взаимодействия пользователя с приложением, целесообразно создать для них всплывающее меню. Оно может оказаться эффективным средством в тех случаях, когда элемент управления используется либо для передачи некоторой величины, имеющей несколько возможных значений, либо для доступа к контекстно-зависимой справочной информации. Для таких меню должны соблюдаться стандартные правила создания всплывающих меню, приведенные ниже, за исключением того, что в данном случае щелчок ЛКМ на выбранном пункте не запускает действие, связываемое с элементом управления. Другими словами, всплывающее меню элемента уп-

равления позволяет пользователю определить, какие действия он реализует в текущей ситуации, но не позволяет непосредственно выполнить эти действия.

Несмотря на то, что меню по своему предназначению имеет полное право входить в число элементов управления, практически во всех изданиях, посвященных инструментальным средствам создания приложений, его рассматривают как самостоятельную компоненту интерфейса. Объясняется это отличием программной реализации меню от других элементов управления (которые в англоязычной литературе именуются *controls*). Тем не менее на этапе проектирования пользовательского интерфейса многие вопросы, относящиеся к использованию меню, должны решаться совместно с проектированием других элементов управления. Именно в силу указанного обстоятельства технология проектирования и использования меню как средства взаимодействия пользователя с приложением рассматривается в данной главе.

5.1. МЕНЮ

Меню содержит перечень команд, имеющихся в распоряжении пользователя при выполнении определенного шага задания или задания в целом. Меню предоставляет пользователю возможность выбора необходимого средства решения задачи, не требуя от него запоминания имен команд и их синтаксиса.

Существуют различные типы меню, в том числе выпадающие, всплывающие и каскадные (иерархические) меню. Применение каждого из них имеет свои особенности, которые рассматриваются ниже.

5.1.1. ГЛАВНОЕ МЕНЮ ОКНА И ВЫПАДАЮЩИЕ МЕНЮ

Одна из наиболее распространенных форм меню - линейная последовательность команд (или разделов). Именно в таком виде выполнено *главное меню* окна, расположенное непосредственно под полосой заголовка первичного окна (рис. 5.1). В связи с этим главное меню называют также *полосой меню*.

Полоса меню содержит названия пунктов меню, каждый из которых предоставляет доступ к выпадающему меню.

Рис. 5.1. Главное меню первичного окна

Содержание главного меню и связанных с ним выпадающих меню определяется функциональным предназначением вашего приложения и контекстом выполняемого пользователем задания. Вы можете обеспечить выбор пользователем дополнительной конфигурации окна, при которой главное меню не отображается. В этом случае необходимо предусмотреть элементы управления, обеспечивающие доступ к тем же функциям приложения, что и главное меню.

ВЗАИМОДЕЙСТВИЕ С ВЫПАДАЮЩИМ МЕНЮ

Когда пользователь выбирает пункт главного меню, отображается связанное с ним выпадающее меню. Чтобы указать выбираемый пункт меню с помощью мыши, пользователь устанавливает указатель на его название и щелкает ЛКМ. Это действие приводит к выделению (подсветке) данного пункта и открытию связанного с ним выпадающего меню (рис.5.2).

Выпадающее меню отображается как панель с пунктами меню, размещенными в виде столбца. Хотя система поддерживает возможность вывода пунктов выпадающего меню в несколько столбцов, избегайте такой формы представления, поскольку она усложняет работу пользователя.

По мере того, как пользователь перемещает указатель по меню, каждый его пункт подсвечивается, отображая текущее положение указателя. Щелкая ЛКМ на пункте меню, пользователь выбирает команду, связанную с этим пунктом; при этом выпадающее меню закрывается. Если пользователь решил отказаться от работы с выпадающим меню, он может закрыть его, повторно щелкнув ЛКМ на том же пункте главного меню. Аналогичный результат дает и щелчок ЛКМ на другом пункте главного меню (правда, в этом случае будет открыто выпадающее меню, связанное с этим пунктом).

Для работы с выпадающим меню с помощью клавиатуры используется клавиша <Alt>, которая активизирует главное меню. Когда пользователь нажимает текстовую клавишу, удерживая нажатой клавишу <Alt>, система отображает выпадающее меню, соответствующее нажатой текстовой клавише (как правило, это клавиша, сопоставленная первому символу в названии пункта меню).

Рис. 5.2. Выпадающее меню

Чтобы получить доступ с клавиатуры к выпадающему меню, пользователь может также использовать клавиши управления курсором (<Вправо> и <Влево>). Если пользователь уже выбрал (открыл) одно из выпадающих меню, эти клавиши позволяют последовательно переходить от одного пункта главного меню к другому, причем новое выпадающее меню будет открыто автоматически (вместо меню, открытого ранее); при этом переход из крайних пунктов выполняется «по кругу» (от последнего к первому и наоборот).

Нажатие клавиш <Вверх> или <Вниз> также приводит к открытию выпадающего меню (если в данный момент нет открытых меню). Нажатие этих клавиш при открытом выпадающем меню обеспечивает переход на следующий пункт меню в соответствующем направлении; переход между крайними пунктами, как и в главном меню, выполняется «по кругу». Если выпадающее меню имеет несколько столбцов, то выполняется переход из последнего пункта текущего столбца к первому в следующем столбце.

Пользователь может закрыть выпадающее меню, нажав клавишу <Alt>; при этом также деактивируется главное меню. Нажатие клавиши <Esc> также позволяет закрыть выпадающее меню, однако эта клавиша отменяет только меню текущего уровня и оставляет выделенным соответствующий пункт главного меню. Повторное нажатие клавиши <Esc> отменяет выделение пункта главного меню и деактивирует полосу меню, возвращая фокус ввода на информацию, отображаемую в окне.

Для ускоренного выбора команд в выпадающих меню могут быть назначены *клавиши быстрого доступа (клавиши-акселераторы)*. Когда пользователь нажимает клавишу-акселератор, связанную с командой в меню, команда немедленно выполняется (без открытия меню). Дополнительно вы можете также выделить соответствующий пункт главного меню, содержащий выполняемую команду, не отображая при этом выпадающее меню.

Ниже более подробно описаны основные соглашения для выпадающих меню наиболее часто используемых в приложениях

Меню *Файл* (File)

Данное меню обеспечивает интерфейс для выполнения основных операций с файлами. Обычно приложение должно включать такие команды, как *Открыть*, *Сохранить*, *Отправить*, *Печать*. Эти команды часто также включаются во всплывающее меню пиктограммы, отображаемой в полосе заголовка окна.

Если ваше приложение поддерживает команду *Выход*, поместите ее последней в меню *Файл*, отделив от других команд символом-разделителем. При выполнении этой команды закройте все открытые окна и файлы, и завершите работу приложения. Если какие-либо объекты при закрытии окна должны остаться активными (например, рабочая папка или принтер), включите в меню вместо команды *Выход* команду *Закрыть*.

Меню *Правка* (Edit)

В это меню входят универсальные команды редактирования, такие, как *Вырезать*, *Вставить*, *Копировать*, а также команды связывания и внедрения объектов (команды OLE); кроме них в этот раздел могут также включаться следующие команды, если они поддерживаются системой (табл. 5.1).

Целесообразно также включить эти команды и во всплывающее меню выбранного объекта.

Таблица 5.1

Команды меню *Правка*

Команда	Функция
Отменить (Undo)	Отмена последнего действия
Повторить (Redo)	Повторное выполнение последнего (в том числе отмененного) действия
Найти и заменить (Find and Replace)	Поиск и замена последовательности символов
Удалить (Delete)	Удаление текущего выбора (объекта, фрагмента текста и т.д.)
Дублировать (Duplicate)	Создание копии текущего выбора

Меню *Вид* (View)

В этом меню содержатся команды, которые изменяют вид (форму представления) данных, отображаемых в окне, а также формат самого окна. К

командам первого типа относится, в частности, команда Масштаб (Zoom), а к командам второго типа — Показать/Скрыть строку состояния. Эти команды рекомендуется также включать во всплывающее меню окна или подокна.

Меню Окно (Window)

Данный раздел используется в приложениях, реализующих многодокументный интерфейс (MDI); в него включаются команды, связанные с управлением окнами в пределах рабочей области MDI (рис. 5.3).

Рис. 5.3. Меню Окно

Эти команды рекомендуется включать также во всплывающее меню родительского окна MDI.

Меню Помощь (Help)

Команды помощи используются для обеспечения доступа пользователя к справочной информации и к встроенным средствам обучения (если таковые имеются). Одна из рекомендуемых команд этого раздела — Содержание, которая позволяет пользователю просмотреть перечень тем, включенных в справочную систему приложения. Кроме того, вы можете дополнительно включить в это меню такие команды, как Поиск по индексу и Найти тему, а также другие команды помощи пользователю, учитывающие специфику приложения.

Если вы считаете необходимым предоставить пользователям информацию об авторском праве и версии вашего приложения, включите в это меню команду About (О программе). Когда пользователь выбирает эту команду, отобразите окно, содержащее имя приложения, номер версии, авторскую информацию и другие дополнительные сведения, относящиеся к приложению.

5.1.2. ВСПЛЫВАЮЩИЕ МЕНЮ

Даже если вы используете главное меню в качестве основного средства взаимодействия пользователя с приложением, вы тем не менее должны также поддерживать **всплывающие** меню. Всплывающие меню предоставляют пользователю эффективный способ доступа к операциям над объектами (рис. 5.4). Поскольку всплывающие меню отображаются в текущей Позиции (соответствующей положению указателя), они избавляют пользователя от необходимости перемещать указатель в полосу меню или на панель инструментов. Кроме того, поскольку всплывающие меню содержат команды, учитывающие специфику выбранного объекта или текущей ситуации, они позволяют сократить число команд, среди которых пользователь должен сделать выбор. И, наконец, всплывающие меню позволяют минимизировать объем отображаемой на экране информации, поскольку они появляются только по требованию пользователя.

Рис. 5.4. Всплывающее меню объекта

Хотя всплывающее меню внешне похоже на выпадающее меню, между ними имеется принципиальное различие: всплывающее меню должно содержать

только те команды, которые относятся к выбранному объекту (или объектам) и текущей ситуации, а не команды, сгруппированные по выполняемым функциям. Например, всплывающее меню для выбранного фрагмента текста может включать команды перемещения и копирования текста, а также команды установки параметров шрифта. Старайтесь сократить размер всплывающего меню, ограничивая количество включаемых в него пунктов. С этой целью в него следует включать только специфические команды, позволяющие пользователю управлять индивидуальными свойствами выделенного объекта.

Контейнер или композиция, из которых выбирается некоторый элемент, обычно поддерживают создание всплывающего меню для выбранного объекта. Аналогично, перечень команд, включаемых во всплывающее меню объекта, определяется не только свойствами самого объекта, но и свойствами содержащего его контейнера. Например, всплывающее меню для файла в папке содержит команды пересылки файла. Эти команды «порождаются» папкой (контейнером), а не файлом, над которым они должны быть выполнены. То же самое относится и к всплывающим меню, содержащим средства OLE для выделенного объекта.

Избегайте использования всплывающего меню в качестве единственного доступного средства для выполнения пользователем тех или иных действий. Вместе с тем, команды всплывающего меню не должны дублировать содержимое одного из выпадающих меню.

Выбирая последовательность расположения команд во всплывающем меню, пользуйтесь следующими руководящими принципами:

- Первыми должны располагаться основные команды для работы с объектом (например, *Открыть*, *Исполнить*, *Печать*), другие команды, поддерживаемые объектом (определяемые непосредственно его свойствами или текущим контекстом), и команда *Что это?* (если она поддерживается системой).

- Во вторую группу должны быть включены команды, реализуемые через буфер обмена (*Вырезать*, *Копировать*, *Вставить*).

- Последними должны идти команды редактирования дополнительных атрибутов объекта (если таковые имеются).

Для открытия всплывающего меню с помощью мыши пользователь должен щелкнуть ПКМ на интересующем его объекте. При этом объект изображается как выбранный. Как правило, всплывающее меню выводится на экран таким образом, чтобы его левый верхний угол находился в позиции указателя; однако если при этом меню (или его часть) может оказаться за краем экрана, его положение должно быть скорректировано.

Если указатель находится над ранее выбранным объектом, всплывающее меню должно относиться к этому объекту. Если пользователь щелкает ПКМ за пределами выбранного объекта, но в пределах той же области выбора, установите новую область выбора и отобразите меню для этой области. Закройте всплывающее меню, когда пользователь щелкает ЛКМ за пределами меню, или если нажимает клавишу <Esc>.

Вы можете также использовать всплывающие меню для объектов, выбираемых косвенно; к таким объектам относятся, в частности, полоса прокрутки и элементы строки состояния. Команды работы с такими элементами должны включаться во всплывающее меню того объекта, для управления

которым предназначены эти элементы. Например, полоса прокрутки обеспечивает перемещение по документу, поэтому во всплывающее меню документа могут быть включены такие команды, как *Начало документа*, *Конец документа*, *Следующая страница*, *Предыдущая страница*. Но в тех случаях, когда элемент управления интерпретируется как самостоятельный объект, для которого могут изменяться формат или размещение в окне, в меню дополнительно могут быть включены команды, относящиеся непосредственно к элементу управления (например, команды для перемещения или копирования этого элемента). Пример такого меню показан на рис. 5.5.

Рис. 5.5. Всплывающее меню для панели инструментов

Для клавиатурного доступа к всплывающим меню используется комбинация клавиш <Shift>+F10 (или клавиша <Application> для клавиатур, которые поддерживают спецификацию Windows). Кроме того, для работы с ними могут быть использованы клавиши управления курсором, клавиши <Enter> и <Esc>; все они действуют так же, как и при работе с выпадающим меню. Избегайте использования клавиш-акселераторов для выполнения команд всплывающего меню, поскольку для их описания требуется дополнительное пространство на панели меню.

Особенности использования всплывающих меню, включаемых в приложение, зависят от объектов и контекста, создаваемых этим приложением. Тем не менее существует несколько типов всплывающих меню, являющихся достаточно общими для всех Windows-приложений. Ниже приведены рекомендации по формированию и применению таких меню.

Всплывающее меню окна

Данное меню содержит команды, предназначенные для управления форматом и состоянием окна. Например, всплывающее меню типового первичного окна включает команды *Заккрыть*, *Восстановить*, *Переместить*, *Масштаб*, *Свернуть*, *Развернуть*. Не следует путать всплывающее меню первичного окна с выпадающим меню *Окно*, используемым в приложениях MDI.

В меню окна могут быть включены и другие команды, относящиеся к работе с окном; например, команда *Разделить*, выполняющая разбиение окна на подокна, или команды, позволяющие открывать подчиненные вторичные окна (типа *Показать палитру*).

Вторичное окно также может иметь всплывающее меню. Однако, поскольку диапазон операций для него более ограничен по сравнению с первичным окном, меню вторичного окна содержит только команды *Переместить* и *Заккрыть*. Окно, отображающее цветовую палитру, может иметь всплывающее меню, содержащее единственную команду — *Расположить сверху*, которая определяет, что данное окно должно всегда выводиться поверх своего родительского окна и всех его вторичных окон.

Всплывающее меню окна отображается на экране по щелчку ПКМ, если указатель находится в любой точке полосы названия окна, за исключением пиктограммы окна. Щелчок ПКМ на пиктограмме окна приводит к появлению

на экране всплывающего меню для объекта, представленного этой пиктограммой.

Всплывающее меню пиктограммы

Всплывающее меню пиктограммы содержит операции над объектом, представленным этой пиктограммой. Доступ к всплывающему меню приложения или пиктограммы документа реализуется на основе стандартных соглашений для всплывающих меню, таких, например, как открытие меню по щелчку ПКМ.

Всплывающее меню, как правило, поддерживается и для пиктограмм контейнерных приложений. Например, системой автоматически предусматриваются всплывающие меню для пиктограмм стандартных папок или для Рабочего стола системы. Тем не менее, если ваше приложение должно поддерживать всплывающие меню для технологии OLE, определите для обрабатываемых документов или файлов данных вашего приложения соответствующие средства работы с меню.

Всплывающее меню для контейнера включает команды, определяемые его содержимым, в первую очередь это команды пересылки и регистрации объектов соответствующего типа. Примером одной из таких команд является команда *Создать*, которая автоматически генерирует новый файл данных того типа, который поддерживается приложением.

Всплывающее меню пиктограммы Windows-приложения должно включать команды, указанные в табл. 5.2.

Таблица 5.2.

Команды всплывающего меню для файла приложения

Команда	Значение
Открыть (Open)	Открывает файл приложения
Отправить (Send To)	Отображает подменю направлений, по которым может быть передан файл. Содержание подменю определяется особенностями используемой системы
Вырезать (Cut)	Помечает перемещаемый файл (файл помещается в буфер обмена)
Копировать (Copy)	Помечает копируемый файл (копия файла помещается в буфер обмена)
Вставить (Paste)	Пытается открыть файл, находящийся в буфере обмена, с помощью активного приложения
Создать ярлык (Create Shortcut)	Создает ярлык файла
Удалить (Delete)	Удаляет файл
Переименовать (Rename)	Позволяет пользователю редактировать имя файла
Свойства (Properties)	Открывает панель свойств файла

Всплывающее меню, связанное с пиктограммой, представляющей документ или файл данных, обычно включает следующие команды (табл. 5.3)

За исключением команд *Открыть* и *Печать*, система автоматически обеспечивает выполнение перечисленных выше команд для пиктограмм, используемых в системных контейнерах (таких, например, как Рабочий стол или папки с *фатами*). Если ваше приложение создает свои собственные контейнеры для файлов, необходимо обеспечить поддержку соответствующих команд для работы с ними через всплывающее меню.

Для использования в меню команд *Открыть* и *Печать* ваше приложение должно зарегистрировать эти команды в системном реестре. Вы можете также зарегистрировать дополнительные или заменяющие их команды (например, команду *Быстрый просмотр*, которая отображает содержимое файла, не открывая приложение, или команду *Что это?*, используемую для вывода справки по файлам данных).

Таблица 5.3.

Команды всплывающего меню для пиктограммы файла данных или документа

Команда	Значение
Открыть (Open)	Открывает первичное окно файла
Печать (Print)	Выводит файл на принтер, заданный по умолчанию
Быстрый просмотр (Quick View)	Открывает файл, используя специальные средства просмотра
Отправить (Send To)	Отображает подменю направлений, по которым может быть передан файл. Содержание подменю определяется особенностями используемой системы
Вырезать (Cut)	Помечает перемещаемый файл (файл помещается в буфер обмена)
Копировать (Copy)	Помечает копируемый файл (копия файла помещается в буфер обмена)
Удалить (Delete)	Удаляет файл
Переименовать (Rename)	Позволяет пользователю редактировать имя файла
Свойства (Properties)	Открывает панель свойств файла

Пиктограмма, расположенная в полосе заголовка окна, и пиктограмма, посредством которой пользователь открывает окно, представляют один и тот же объект. Поэтому обе указанные пиктограммы должны быть связаны с однотипными выпадающими меню. За исключением того, что для пиктограммы, находящейся в полосе заголовка окна, команду *Открыть* следует заменить командой *Закрыть* и дополнительно ввести команду *Сохранить* (если вносимые в документ изменения требуют явного выполнения операции записи в файл).

Для MDI-приложения сопоставьте всплывающее меню пиктограмме приложения в родительском окне, следуя рассмотренным ранее соглашениям по использованию пиктограммы в полосе заголовка окна. Рассмотрите также возможность включения во всплывающее меню такого приложения следующих дополнительных команд (там, где они уместны).

Кроме того, рекомендуется сопоставить соответствующие всплывающие меню и для пиктограмм, отображаемых в полосе заголовка дочерних окон.

Таблица 5.4.

**Дополнительные команды всплывающего меню пиктограммы
родительского окна**

Команда	Значение
Создать (New)	Создает новый файл данных или отображает список типов файлов данных, поддерживаемых приложением
Сохранить Все (Save All)	Сохраняет все файлы данных, открытые в рабочей области MDI, и состояние окон MDI-приложения
Найти (Find)	Открывает окно, которое позволяет пользователю задавать критерии поиска файлов

При необходимости указанные дополнительные команды могут быть включены во всплывающие меню обычных (не MDI) приложений.

5.1.3. КАСКАДНЫЕ МЕНЮ

Каскадное меню (называемое также *иерархическим* меню или *дочерним* меню) — это подменю, на которое «распадается» пункт меню более высокого уровня. Визуально на наличие каскадного меню указывает треугольник, выводимый рядом с родительским пунктом меню (рис. 5.6).

Каскадное меню может использоваться для того, чтобы предоставить пользователю возможность дополнительного выбора и при этом не занимать дополнительное пространство в родительском меню. Оно может также быть полезным для отображения иерархически связанных объектов.

Следует помнить, что использование каскадных меню ведет к увеличению сложности интерфейса, поскольку предполагает последовательное прохождение нескольких меню. Это, в свою очередь, требует от пользователя большей координированности при выборе пунктов меню. В силу указанных причин старайтесь ограничивать применение каскадных меню только теми ситуациями, где они действительно необходимы. При этом минимизируйте количество уровней для любого пункта данного меню, в идеале ограничивая его единственным подменю. Избегайте использования каскадных меню для доступа к распространенным, часто используемым командам.

В качестве альтернативного решения реализуйте дополнительный выбор с помощью вторичных окон, особенно в тех случаях, когда выбор представляет собой установку не связанных между собой параметров; это позволяет пользователю выполнять установку нескольких параметров за один сеанс работы с окном. Вы можете также реализовать установку наиболее часто изменяемых параметров как выбор соответствующих элементов на панели инструментов окна.

Рис. 5.6. Каскадное меню

Взаимодействие пользователя с каскадным меню подобно взаимодействию с выпадающим меню, за исключением того, что каскадное меню отображается после некоторой задержки. Это позволяет избежать вывода подменю, если пользователь просто просматривает родительское меню или перемещает указатель к другому пункту меню. Если после открытия каскадного меню пользователь перемещает указатель с другого пункта родительского меню, то каскадное меню после короткой задержки закрывается. Эта задержка позволяет пользователю непосредственно перейти из родительского меню в соответствующее каскадное меню, не нажимая кнопку мыши.

5.1.4. ЗАГОЛОВОК МЕНЮ

Каждое выпадающее и каскадное меню имеет собственный заголовок. Для выпадающего меню его заголовок, который отображается в полосе меню окна, служит входом в меню. Для каскадного меню заголовком является имя родительского пункта меню. Заголовок меню представляет соответствующее меню в целом и должно отражать его предназначение, так же, как и названия пунктов в меню должны пояснять назначение каждого из них.

Старайтесь обозначать каждое меню одним словом. Многословные названия или названия, содержащие пробелы, могут быть восприняты пользователем как два (или более) однословных названия. Вместе с тем, рекомендуется избегать необычных сложных слов, таких например, как *Шрифтразмер*.

Определите для каждого меню один символ, которому будет соответствовать клавиша быстрого доступа. Этот символ отображается в заголовке меню как подчеркнутый символ (рис.5.7.). Каждое меню должно иметь уникальную клавишу быстрого доступа. В противном случае непосредственный доступ к требуемому меню окажется невозможен.

Рис. 5.7. Обозначение клавиш быстрого доступа к выпадающим меню

5.1.5. ПУНКТЫ МЕНЮ

Каждый пункт меню соответствует определенному действию, которое может выполнить пользователь в данной ситуации. Пункты меню могут быть представлены в текстовой или в графической форме (Например, в виде пиктограмм), либо как комбинация графики и текста. При использовании графического интерфейса формат пункта меню должен обеспечивать визуальное отображение выбора пользователя, как показано на рис. 5.8.

Рис. 5.8. Различные форматы пунктов меню

Если меню содержит большое количество пунктов, и они могут быть сгруппированы по некоторому признаку, целесообразно визуально разделить

такие группы. В качестве стандартного разделителя используется горизонтальная прямая линия. Не рекомендуется использовать в качестве разделителей сами пункты меню, либо текстовые подзаголовки, которые могут быть восприняты пользователями как название пункта меню (рис. 5.9).

Рис. 5.9. Пример неудачного деления пунктов меню на группы

Всегда обеспечивайте пользователя визуальным указанием тех пунктов, которые в данной ситуации не доступны. Обычно такие пункты меню либо «обесцвечиваются», либо вообще не отображаются на панели, причем первый способ является более предпочтительным с точки зрения обеспечения обратной связи с пользователем. Из этих же соображений целесообразно сохранить для недоступных пунктов возможность выдачи подсказки. В частности, если окно содержит строку состояния, то в ней можно отобразить сообщение, указывающее, что собой представляет данная команда, и почему она недоступна.

Если в некоторой ситуации ни один из пунктов меню не является доступным, сделайте недоступным данное меню в целом.

Система обеспечивает стандартный способ отображения недоступных пунктов меню. Если вы хотите использовать собственный способ визуализации таких пунктов, помните о том, что он должен быть одинаков для всех однотипных меню создаваемого приложения.

Форматы пунктов меню

Многие пункты меню вступают в силу сразу же, как только они были выбраны. Если пункт меню является командой, которая требует ввести дополнительную информацию, имя команды должно сопровождаться многоточием (...). Многоточие (эллипсис) сообщает пользователю, что информация неполна. Выполнение таких команд обычно сопровождается выводом на экран диалоговой панели, обеспечивающей ввод пользователем требуемой информации. Например, команда *Сохранить как...* содержит эллипсис, поскольку она не может быть выполнена до тех пор, пока пользователь не укажет или не подтвердит имя файла, в который будет произведена запись.

Вместе с тем, не каждая команда, которая приводит к открытию диалоговой панели или другого вторичного окна, должна содержать эллипсис. Например, не следует его использовать для команды *Свойства*, поскольку выполнение данной команды само по себе предполагает открытие окна, отображающего свойства

соответствующего объекта. Не следует также его использовать для команд, выполнение которых может завершаться выводом на экран окна сообщения.

Наряду с использованием пунктов меню для выполнения команд, с их помощью можно также реализовать переключение режимов или установку требуемого состояния (или свойств) объекта без запуска соответствующего процесса. Например, выбор одного из пунктов меню, которое содержит список панелей инструментов, подразумевает соответствующее изменение формата окна. Если меню обеспечивает установку значений свойств объекта (например, размера шрифта), то выбор одного из пунктов такого меню означает изменение этого значения.

Пункты меню для установки параметров состояния объектов могут быть независимыми или взаимозависимыми.

Независимые пункты меню эквивалентны флажкам. Например, если меню содержит список параметров текста, таких как начертание шрифта (*Полужирный*, *Курсив*), то они образуют группу независимо устанавливаемых параметров. Пользователь может изменить каждый из них независимо от значений других, даже если все они относятся к одному фрагменту текста. Для визуального отображения выбора таких пунктов меню рекомендуется использовать **маркер флажка** (в виде «птички»), отображаемый на панели меню слева от выбранного пункта.

Взаимозависимые пункты меню аналогичны переключателям. Например, если меню содержит способы выравнивания текста (*По центру*, *По правому краю*, *По ширине*), то они образуют группу взаимозависимых параметров. Поскольку к выбранному участку текста может быть применен только один тип выравнивания, выбор одного пункта из группы автоматически отменяет выбор любого другого. Для визуального отображения выбора таких пунктов меню рекомендуется использовать **маркер переключателя** (в виде кружка), также отображаемый слева от выбранного пункта меню.

При использовании меню для выбора одного из двух альтернативных вариантов, как например, присутствие или отсутствие некоторого свойства объекта, для указания выбранного пункта также может использоваться маркер флажка. Например, если состояние текстового фрагмента соответствует пункту меню *Полужирный*, отобразите возле этого пункта маркер и уберите его, когда в выбранном фрагменте используется не полужирный шрифт. Если фрагмент содержит смешанный шрифт, пункт *Полужирный* также отображается без маркера.

Если же два какие-либо состояния не являются очевидными для пользователя альтернативами, целесообразно использовать два замещаемых названия для одного пункта меню, каждое из которых соответствует одному из альтернативных состояний. Например, не очень подготовленный пользователь может предположить, что альтернативой пункту меню с названием *Полное дублирование* является вариант *Пустое дублирование* (вместо *Дублирование фрагмента*). При выборе такого пункта меню его название изменяется на альтернативное.

Выбирая способ отображения в меню альтернативных вариантов, следует придерживаться следующих принципов:

- Если на панели меню имеется достаточное свободное пространство, лучше включить в меню альтернативные варианты в качестве отдельных пунктов; это

позволит пользователю видеть оба варианта одновременно; если таких альтернативных пар в меню несколько, они могут быть отделены друг от друга разделителем.

Избегайте такого определения пунктов меню, при котором их названия изменяются в зависимости от состояния клавиши-модификатора. При таком подходе большинство пользователей не смогут уяснить функциональные возможности приложения.

- При отсутствии в меню пространства для одновременного отображения альтернативных вариантов может быть использован единственный пункт меню, название которого изменяется при выборе на альтернативное. В этом случае название пункта меню отражает не текущее, а альтернативное состояние (т.е. состояние после следующего выбора данного пункта). Где это возможно, старайтесь подбирать такие названия для альтернативных вариантов, для которых клавиша-акселератор будет связана с одним и тем же символом. Например, буква *Д* могла бы использоваться для пункта меню; который переключается между *Полным дублированием* и *Дублированием фрагмента*.

Меню может также содержать пункт, выбираемый **по умолчанию**. Для такого пункта обязательно должна поддерживаться техника ускоренного доступа (например, посредством двойного щелчка ЛКМ) или техника drag-and-drop (*перетаски и оставь*). Так, если для пиктограммы по умолчанию используется команда *Открыть*, включите в меню соответствующий пункт, который также будет выбираться по умолчанию. Аналогично, если для операции drag-and-drop команда, выполняемая по умолчанию, — *Копировать*, отобразите ее во всплывающем меню в качестве пункта, выбираемого по умолчанию.

В соответствии с системными соглашениями пункты, выбираемые по умолчанию, отображаются на панели меню как полужирный текст.

Метка пункта меню

Каждый пункт меню целесообразно снабдить текстовым комментарием или графической меткой, либо комбинацией того и другого. При выборе формата метки следует помнить, что для текста значительно проще назначить клавишу-акселератор.

При выборе текстовых имен для обозначения пунктов меню придерживайтесь следующих правил:

- Используйте уникальные имена для пунктов в пределах одного меню; пункт с тем же названием может повторяться в других меню, чтобы представить аналогичные или другие действия.
- Название пункта может состоять из одного слова или из нескольких, но в любом случае оно должно в сжатой форме отражать смысл выполняемых действий. Пространные названия пунктов меню затрудняют его быстрый просмотр пользователем.
- Определяйте уникальные клавиши быстрого доступа для каждого пункта в пределах меню. Основные правила выбора клавиши-акселератора для пунктов меню — такие же, как и для меню в целом; кроме того, клавиша-акселератор для пункта меню может быть обозначена порядковым номером пункта меню (если таковой отображается на панели меню). Это полезно для тех пунктов меню, названия которых изменяются (например, содержащих имя файла). Где это

возможно, определяйте одни и те же клавиши доступа для одноименных пунктов, используемых в разных меню.

- Старайтесь придерживаться норм используемого языка (как русского, так и английского, или любого другого) по использованию заглавных букв в названии пункта меню. В частности, для англоязычных версий рекомендуется делать заглавной первую букву каждого слова, за исключением артиклей, союзов и предлогов, которые встречаются «внутри» многословных названий. Например, правильными являются следующие названия пунктов меню: *New Folder* (Новая папка), *Go To* (Перейти), *Select All* (Выделить все), *Go to Page* (Перейти на страницу).

- Избегайте использования индивидуальных текстовых стилей для каждого пункта меню. Это не только может затруднить визуальное восприятие меню пользователем, но и сделает невозможным однотипное выделение тех символов, которым сопоставлены клавиши-акселераторы.

Описание клавиш-акселераторов в пунктах меню

Если для выполнения команд выпадающего меню используются клавиши-акселераторы, отобразите на панели меню их сокращенное обозначение. Обозначение каждой клавиши выводится в соответствующей строке меню, но при этом все они выравниваются относительно некоторой общей позиции. Обычно такой позицией является первая свободная после самого длинного пункта в меню, который имеет клавишу-акселератор. Не применяйте пробелы для выравнивания обозначений клавиш, поскольку для некоторых шрифтов они могут отображаться непропорционально.

Для описания в меню клавиш-акселераторов лучше всего использовать те же стандартные имена, которые указаны на клавиатуре. Для описания сочетаний клавиш Ctrl и Shift с другими клавишами используются обозначения в виде Ctrl+<клавиша> (а не Control+<клавиша> или CONTROL+<клавиша> или ^+<клавиша>) и Shift+<клавиша>. Если для быстрого доступа к пункту меню используется функциональная клавиша, отобразите ее имя как Fn, где n — номер функциональной клавиши (например, F3).

Старайтесь не использовать клавиши-акселераторы для работы с всплывающими меню. Такие меню уже сами по себе являются формой ускоренного взаимодействия пользователя с программой и работа с ними изначально ориентирована на использование мыши.

5.2. КНОПКИ

Кнопки - это элементы интерфейса, которые служат для инициирования каких-либо действий или для изменения свойств объектов. Существует три основных типа кнопок: кнопки управления (*Command Buttons*), кнопки установки параметров (*Option Buttons, Radio buttons*), или переключатели, и кнопки независимого выбора, или флажки (*Check boxes*).

5.2.1. КНОПКИ УПРАВЛЕНИЯ

Кнопка управления, обычно именуемая в русскоязычных изданиях просто *кнопкой*, предназначена для запуска связанной с ней команды или операции;

такая кнопка имеет, как правило, прямоугольную форму и снабжается поясняющей меткой, в качестве которой может использоваться текст, графический символ, либо то и другое, как показано на рис. 5.10.



Рис. 5.10. Примеры реализации кнопок

Когда пользователь выбирает кнопку с помощью мыши (щелкает на ней ЛКМ), выполняется команда, связанная с кнопкой. Если пользователь нажимает кнопку мыши (не отпуская ее), фокус ввода переходит на кнопку; при этом изменяется состояние кнопки (она отображается как «нажатая»). При перемещении указателя за пределы кнопки она возвращается в свое исходное состояние. Перемещение указателя обратно на кнопку вновь переводит ее в «нажатое» состояние.

Если пользователь отпускает кнопку мыши, когда указатель находится над кнопкой, выполняется связанная с ней команда; если же указатель расположен за пределами кнопки, фокус ввода «снимается» с кнопки без выполнения команды.

Вы можете определить для кнопок клавиши доступа и клавиши-акселераторы. Кроме того, целесообразно обеспечить пользователю возможность навигации между кнопками с помощью клавиатуры. Для этого рекомендуется использовать клавишу <Tab> в сочетании с клавишами управления курсором. Клавиша <Spacebar> по умолчанию используется для активизации кнопки при установке на нее фокуса ввода.

Результат команды, выполняемой при «нажатии» кнопки, проявляется немедленно и непосредственно влияет на текущую ситуацию. Например, при «нажатии» кнопки, расположенной на панели инструментов приложения, тут же выполняется связанное с ней действие; «нажатие» кнопки, отображенной во вторичном окне, обеспечивает прием вводимых данных и, возможно, закрытие окна.

Как было указано выше, кнопка управления снабжается метком, обозначающей связанное с ней действие. Если в качестве метки используется текст, то он должен отвечать тем же требованиям, которые были рассмотрены ранее применительно к пунктам меню. Если в какой-либо ситуации связанная с кнопкой команда является недоступной, то это должно быть отображено визуально (как правило, в этом случае обесцвечивается метка кнопки).

Для кнопок, связанных с командами, которые требуют ввода дополнительной информации, в качестве соответствующего визуального признака используется эллипсис (...). Для некоторых кнопок при их «нажатии» может выводиться окно сообщения, но это не означает, что метка такой кнопки должна содержать эллипсис.

Кнопки могут также использоваться для расширения вторичного окна с целью отображения в нем дополнительных параметров; такие кнопки называются *кнопками развертывания* и представляют собой частный случай применения кнопок для выполнения специфической функции. Если кнопка используется с указанной целью, ее метка должна содержать пару символов >> (рис. 5.11).

Рис. 5.11. Пример использования кнопки развертывания








В некоторых случаях кнопка может представлять объект и связанное с ним по умолчанию действие. Например, кнопки входа, отображаемые на Панели задач,

представляют первичные окна объектов и команду *Восстановить*. Когда пользователь щелкает на такой кнопке ЛКМ, для выбранного объекта выполняется указанная команда; щелчок на кнопке Входа ПКМ приводит к появлению всплывающего меню для соответствующего объекта.

Кроме того, кнопки могут также использоваться для *выбора* режима работы или значения некоторой величины; в этом случае их применение аналогично применению кнопок двух других типов — переключателей и флажков. Если кнопка используется подобным образом, то для визуального представления состояния выбранной кнопки применяется изображение, несколько отличающееся от стандартного изображения «нажатой» кнопки. Изображения кнопок, рекомендуемые для визуального представления их возможных состояний, приведены в табл. 5.5.

Таблица 5.5.

Рекомендуемые изображения кнопок

Изображение	Состояние кнопки
	Нормальное состояние (кнопка не выбрана)
	Кнопка «нажата»
	Кнопка используется как переключатель находится в состоянии "выбрана"
	Кнопка недоступна
	Кнопка используется как переключатель, но в данный момент не доступна
	Кнопка находится в состоянии "не определено"
	Кнопка находится в фокусе ввода

В некоторых случаях кнопка управления может быть использована для открытия выпадающего меню; такую кнопку называют **кнопкой меню**. Хотя это достаточно специфическое использование кнопок, не предусмотренное системой, вы можете реализовать такой элемент интерфейса, используя стандартные компоненты.

Кнопка меню выглядит как стандартная кнопка управления, за исключением того, что ее метка содержит треугольную стрелку, подобную используемым в каскадных меню (рис. 5.12.)

Кнопка меню поддерживает тот же тип взаимодействия, что и выпадающее меню: меню отображается при нажатии кнопки; выбор пункта меню осуществляется посредством перемещения указателя по панели меню. Подобно любому другому меню, чтобы проследить перемещение указателя, в данном случае также используется подсветка пунктов меню.

Рис 5.12. Кнопка меню

Щелчок ЛКМ за пределами меню или повторное нажатие кнопки приводит к закрытию панели меню.

5.2.2. ПЕРЕКЛЮЧАТЕЛИ

Переключатели, иногда называемы также *радио-кнопками* (что, на наш взгляд, не очень корректно по отношению к А.С. Попову), предоставляют пользователю возможность выбрать единственный вариант из предлагаемого множества взаимоисключающих альтернатив. Другими словами, в любой группе переключателей может быть установлен (выбран) только один. Поэтому если окно содержит переключатели, относящиеся к разным объектам или к различным свойствам одного объекта, они обязательно должны быть явным образом разделены на соответствующие группы (рис. 5.13).

Рис. 5.13. Группа переключателей

Переключатели изображаются в виде небольших кружков. Если кнопка выбрана, в середине кружка появляется точка (маркер). Не рекомендуется использовать переключатели для запуска команд (т.е. в качестве кнопок управления). Единственным исключением является случай, когда двойной щелчок ЛКМ на переключателе используется как средство быстрого выбора значения некоторой величины и выполнения встроенной команды окна, которая является основным (или единственно возможным) действием пользователя для данного окна.

Как правило, переключатели используются для выбора одного из возможных значений некоторого свойства объекта. При этом желательно ограничить число различных вариантов (обычно используют не более семи), но вместе с тем их должно быть *не менее двух*. Если необходимо предложить пользователю большее количество вариантов, используйте другие элементы управления, например меню. На рис. 5.14 приведен пример не вполне корректного использования переключателя (если маркер установлен, то убрать его невозможно, не закрывая диалоговой панели).

Каждый переключатель снабжается текстовой меткой. Если же для пояснения назначения кнопок в группе более удобны графические символы, лучше использовать кнопки управления. Стандартные переключатели позволяют создавать метки, состоящие из нескольких строк. В таких случаях целесообразно использовать верхнее выравнивание (если ситуация не требует какой-либо другой ориентации текста).

Рис. 5.14. Пример некорректного использования переключателя

Как и для кнопок управления, для выбора переключателя используется щелчок ЛКМ — или в кружке, или на метке кнопки. Фокус ввода перемещается на метку кнопки, если указатель находится над ней или над кружком, когда пользователь нажимает кнопку мыши. Если пользователь перемещает указатель за пределы переключателя, не отпуская кнопку мыши, он возвращается в исходное состояние. Выбор не будет зафиксирован до тех пор, пока пользователь не отпустит кнопку мыши (при этом указатель должен находиться над переключателем). Повторный щелчок ЛКМ на том же переключателе не изменяет его состояние; пользователю нужно явно выбрать альтернативный вариант в группе, чтобы изменить или восстановить прежний выбор.

Для работы с переключателями с помощью клавиатуры могут быть определены соответствующие клавиши доступа. Выбор переключателя в группе может также выполняться клавишей <Tab> и клавишами перемещения курсора.

5.2.3. ФЛАЖКИ

Подобно переключателям, флажок может находиться в одном из двух состояний: "включено" или "выключено"; применительно к флажкам эти состояния обычно именуются "флажок установлен" или "флажок снят". Основное отличие флажков от переключателей заключается в том, что они используются для отображения независимых вариантов выбора. Другими словами, пользователь может одновременно выбрать (установить) несколько флажков, входящих в одну группу.

Флажок изображается в виде небольшого квадратного окошечка с сопутствующей меткой. Когда флажок установлен, в окошечке появляется маркер (рис.5.15).

Рис. 5.15. Группы флажков

Стандартный флажок имеет текстовую метку. Если же для пояснения предназначения флажка требуется графический символ, следует использовать вместо флажка кнопку управления. При выборе текста для метки пользуйтесь все тем же золотым правилом: чем короче и содержательнее надпись, тем лучше; если текст не удалось уместить в одну строку, используйте верхнее выравнивание строк (если по контексту не требуется какой-то другой вариант).

Рис.5.16. Применение флажка для управления элементами интерфейса:

- а) состояние диалоговой панели, когда флажок «В виде значка» снят;
- б) состояние диалоговой панели, когда флажок установлен

В некоторых случаях флажок может использоваться для управления другими элементами интерфейса. В частности, от состояния флажка может зависеть доступность элементов более низкого уровня иерархии (рис. 5.16).

Если список установок содержит большое число пунктов, или если их количество изменяется, вместо флажков удобнее использовать список множественного выбора.

Когда пользователь щелкает ЛКМ в окошечке флажка или на его метке, состояние флажка переключается на противоположное. При нажатии кнопки мыши фокус ввода переходит на флажок. Как и при работе с другими элементами управления, если пользователь, удерживая кнопку мыши, перемещает указатель за пределы флажка, элемент возвращается в исходное состояние. Состояние флажка не изменится, пока кнопка мыши не будет отпущена. Для изменения состояния указатель должен находиться в этот момент над флажком или его меткой.

Чтобы предоставить пользователю клавиатурный интерфейс для управления флажками, определите для каждой из них клавишу доступа. Кроме того, для навигации между кнопками могут поддерживаться клавиша <Tab> и клавиши управления курсором. Если фокус ввода находится на одном из флажков, отображаемых на диалоговой панели, то для его перемещения между флажками может использоваться клавиша <Spacebar>.

В некоторых случаях оказывается весьма полезным использование дополнительного, третьего состояния флажка, которое мы в дальнейшем будем называть *промежуточным*. На рис. 5.17 показано визуальное изображение этого состояния.

Рис. 5.17. Изображение флажка в промежуточном состоянии (увеличенный масштаб)

Примером такой ситуации, в которой целесообразно использовать промежуточное состояние флажка, может служить работа с фрагментом текста, содержащего несколько различных типов шрифта. Если для установки каждого типа шрифта используется отдельная кнопка-флажок, то при выборе «разнородного» фрагмента текста все они должны быть установлены в промежуточное состояние. Если в такой ситуации пользователь выберет (установит) один из флажков, то для всего фрагмента текста будет применен тип шрифта, соответствующий данному флажку. Повторный выбор флажка приводит к изменению его состояния на противоположное (флажок будет снят). Если же флажок будет выбран в третий раз, то он вновь вернется в исходное (промежуточное) состояние; то же самое произойдет и с редактируемым фрагментом текста.

5.3. СПИСКИ

Список предоставляет пользователю удобное средство управления выбором требуемых объектов или их свойств. Элементы списков могут быть представлены как в текстовой, так и в графической форме. Любой список как элемент интерфейса должен обеспечивать не только визуальное отображение сделанного пользователем выбора, но и поддержку связанных с выбранными пунктами (или пунктом) действий.

Использование списка является наиболее целесообразным в тех случаях, когда число возможных вариантов выбора велико, либо когда их перечень может

изменяться. Если какой-либо элемент списка не может быть выбран в данной ситуации, следует удалить его из списка. Например, если определенный размер не доступен для установленного шрифта, не отображайте этот размер в списке.

Порядок расположения элементов в списке определяется, как правило, его содержанием, и должен облегчать просмотр списка пользователем. Например, если список содержит имена, то их целесообразно расположить по алфавиту; список дат лучше составить в хронологическом порядке.

Стандартный список не снабжается текстовой меткой. Тем не менее, вы можете создать для списка метку в виде статической текстовой области, которая будет использоваться в качестве ссылки для клавиатурного доступа к списку.

Если в текущей ситуации список недоступен, отобразите на экране его метку в соответствующем виде.

Ширина поля списка должна быть достаточной, чтобы разместить в списке название пункта средней длины. Если заранее невозможно подобрать таковую, используйте один из следующих подходов:

- сделайте поле списка достаточно широким, чтобы разместить в нем наиболее «длинный» элемент;
- используйте эллипсис (...) в середине или в конце длинных названий, чтобы сократить их, сохранив при этом те символы, по которым их следует различать. Например, при указании маршрутов доступа наиболее важными являются обычно начало и конец маршрута, поэтому промежуточные каталоги могут быть заменены эллипсисом: *\Образец \... \Пример;*

• включите в панель списка горизонтальную полосу прокрутки; правда, этот вариант имеет некоторые ограничения по использованию, поскольку наличие полосы прокрутки приводит к сокращению объема данных, которые пользователь может просмотреть одновременно.

Если количество пунктов в списке превышает видимую область, то в поле списка включается вертикальная полоса прокрутки. При этом должны соблюдаться те же правила использования полосы прокрутки, которые были рассмотрены ранее.

Для выбора и прокрутки списка могут быть также использованы клавиши управления курсором. Кроме того, для списка обеспечивается поддержка клавиатурного выбора с помощью текстовых клавиш. Когда пользователь нажимает текстовую клавишу, в списке производится выбор соответствующего пункта, при этом список перемещается таким образом, чтобы выбор был виден пользователю. Последующие нажатия текстовых клавиш обеспечивают продолжение процесса сопоставления слова, набираемого на клавиатуре, с названиями пунктов списка. При этом каждое последующее нажатие клавиши должно быть выполнено в пределах установленной системной задержки. Если задержка истекает, сопоставление возобновляется с первого символа.

Включая список в состав окна приложения, продумайте возможность использования при работе со списком команд редактирования (*Вырезать*, *Копировать*, *Вставить*) и операций прямого манипулирования. Если список содержит пиктограммы или величины, которые пользователь может переместить или скопировать в другие позиции окна (например, в другой список), доступность таких операций может существенно ускорить его работу. Для текущего (просматриваемого) списка эти операции поддерживаются

автоматически; тем не менее, система обеспечивает возможность реализации указанных операции и для других (не активных) списков.

Списки могут различаться способом отображения содержимого и типом выбора, который они поддерживают.

Ниже рассмотрены особенности использования различных видов списков.

5.3.1. СПИСОК ЕДИНИЧНОГО ВЫБОРА

Список единичного выбора (*Single Selection List Box*) используется для выбора только одного пункта в списке. Следовательно, такой элемент управления реализует функцию взаимного исключения, подобно группе переключателей, за исключением того, что список позволяет более эффективно оперировать большим количеством пунктов.

Размер (высота) панели списка единичного выбора обычно выбирается таким образом, чтобы в нем можно было показать от трех до восьми пунктов (рис. 5.18). Всегда включайте в окно списка вертикальную полосу прокрутки. Если все пункты в списке видимы, то стрелки полосы прокрутки следует заблокировать; в остальном правила использования полосы прокрутки остаются прежними.

Рис. 5.18. Пример использования списка единичного выбора

Пользователь может выбрать требуемый пункт в списке единичного выбора, щелкнув на нем ЛКМ. Фокус ввода также устанавливается на этом пункте.

Поскольку список рассматриваемого типа поддерживает выбор только единственного пункта, при изменении выбора любой другой пункт в списке становится *невывбранным*.

Для реализации клавиатурного интерфейса при работе со списком используются клавиши навигации (клавиши управления курсором, <Home>, <End>, <PgUp> и <PgDn>). Могут также использоваться текстовые клавиши, обеспечивающие синхронное сопоставление вводимых символов с символами пунктов списка. Эти клавиши не только управляют прокруткой списка, но также выполняют выбор соответствующего пункта. Если в результате поиска ни один пункт в списке оказался не выбран, то выбранным считается пункт в списке, определяемый действием текущей клавиши навигации. Например, если пользователь нажимает клавишу <Home>, то выбранным становится первый элемент в списке.

Если пункты списка представляют собой значения некоторого свойства объекта, то в списке выделяется текущее значение этой характеристики для выбранного объекта. Если же выбрана группа объектов, имеющих различные значения этой характеристики, то ни один пункт в списке не должен быть выделен.

5.3.2. ВЫПАДАЮЩИЙ СПИСОК

Подобно списку единичного выбора, выпадающий список (*Drop-down List Box*) предусматривает возможность выбора единственного пункта; различие заключается в том, что выпадающий список отображается на экране только по требованию пользователя. Когда список закрыт (точнее, *свернут*), в его окне

отображается только выбранный пункт. Чтобы изменить выбор, пользователю необходимо открыть список. Рис. 5.19 показывает выпадающий список в закрытом и открытом состоянии.

Хотя выпадающий список представляет собой эффективный способ экономного использования пространства окна, он вместе с тем требует от пользователя дополнительных действий для просмотра и выбора пункта по сравнению со списком единичного выбора.

Рис. 5.19. Выпадающий список в свернутом и в открытом состоянии

Ширина окна свернутого выпадающего списка должна быть на несколько пробелов больше средней ширины пунктов в списке. В открытом состоянии окно списка должно быть достаточно высоким для того, чтобы показать три - восемь пунктов. Вместе с тем, свободное пространство в окне должно позволять не только отобразить выбор в списке, но также выполнять пользователю операции прямого манипулирования.

Взаимодействие пользователя с выпадающим списком подобно работе с выпадающим меню. Например, чтобы отобразить список, пользователь может щелкнуть ЛКМ на кнопке меню, а выбор пункта приводит к автоматическому закрытию списка; выбор пункта, как и в меню, выполняется щелчком ЛКМ в соответствующей строке.

Пользователь может также работать с выпадающим списком, используя клавишу доступа и клавиши управления курсором. При нажатии клавиши доступа, клавиши <Tab> или клавиш <Вверх>, <Вниз>, а также двух последних клавиш в сочетании с клавишей <Alt>, выпадающий список открывается. Клавиши управления курсором и текстовые клавиши обеспечивают одновременно навигацию и выбор пункта в списке. Если пользователь нажимает клавиши <Вверх>+<Alt>, <Вниз>+<Alt>, клавишу <Tab>, или клавишу доступа, чтобы перейти на другой элемент интерфейса, список автоматически закрывается. Клавиша <Esc> также закрывает список. При закрытии списка сделанный в нём выбор должен сохраняться.

Если пункты списка представляют собой значения некоторой характеристики объекта, то в списке выделяется текущее значение этой характеристики для выбранного объекта. Если же выбрана группа объектов, имеющих различные значения этой характеристики, то ни один пункт в списке не должен быть выделен.

5.3.3. РАСШИРЕННЫЙ СПИСОК И МНОЖЕСТВЕННОГО ВЫБОРА

Хотя большинство списков обеспечивают единичный выбор, в некоторых ситуациях пользователю требуется выбрать более чем один пункт. Расширенный список (*Extended List Box*) и список множественного выбора (*Multiple Selection List Box*) позволяют реализовать такую операцию.

Для названных списков остаются справедливыми те же соглашения относительно высоты и ширины окна, как и для рассмотренных ранее.

Расширенный поддерживает стандартную навигацию, а также технику непрерывного и непересекающегося выбора (рис. 5.20). Это означает, что список расширенного выбора может быть использован для выбора единственного пункта или единственной области, хотя по умолчанию предназначен для выполнения непересекающегося выбора.

Если вы хотите обеспечить пользователю возможность выбора из списка различных непересекающихся данных, но список расширенного выбора кажется вам слишком громоздким, вы можете создать список множественного выбора. Тогда как список расширенного выбора ориентирован на выбор отдельного пункта или области, список множественного выбора предназначен в первую очередь для реализации независимого выбора нескольких пунктов. Тем не менее, визуально различить два эти вида списков достаточно сложно. Поэтому старайтесь оформлять список множественного выбора таким образом, чтобы внешне он был подобен прокручиваемому списку флажков, как показано на рис. 5.21.

Рис. 5.20. Пример использования расширенных списков

Правда, это требует создания вашей собственной графики для представления пунктов в списке. Такой подход поможет пользователю увидеть отличие интерфейса списка множественного выбора от списков других типов. Он также способствует проявлению различий в использовании клавиатурной навигации.

Рис. 5.21. Список множественного выбора

5.3.4. МОДИФИЦИРУЕМЫЙ СПИСОК

Модифицируемый список (*List View Control*) представляет собой особую форму расширенного списка, который отображает набор пунктов, каждый из которых содержит пиктограмму и текстовую метку. Содержимое модифицируемого списка может быть представлено в одном из четырех видов (табл. 5.6).

Таблица 5.6.

Форматы модифицируемого списка

Формат	Описание
Пиктограмм а	Каждый пункт отображается как полноразмерная пиктограмма с расположенной под ней меткой. Пользователь может перемещать (перетаскивать мышью) пиктограммы в любую позицию в пределах видимой области списка

Маленькая пиктограмма	Каждый пункт отображается как пиктограмма маленького формата со своей меткой, расположенной справа. Пользователь может перемещать (перетаскивать мышью) пиктограммы в любую позицию в пределах видимой области списка
Список	Каждый пункт отображается как пиктограмма маленького формата со своей меткой, расположенной справа. Пиктограммы упорядочены в виде столбца заданного формата
Отчет	Каждый пункт отображается в виде строки, содержащей несколько столбцов; самый левый столбец содержит иконку и метку. Последующие столбцы содержат информацию, предоставляемую приложением, формирующим данный модифицируемый список

Примеры использования различных форматов модифицируемого списка приведены на рис. 5.22.

Рис. 5.22. Форматы модифицируемого списка:

- а) Пиктограмма («Крупные значки»);
- б) Маленькая пиктограмма («Мелкие значки»);
- в) Список;
- г) Отчет («Таблица»)

Для модифицируемого списка поддерживаются также операции выравнивания, выбора и упорядочивания пиктограмм, редактирования меток, а также операции прямого манипулирования.

Данный вид списка рекомендуется использовать в тех случаях, когда выбираемые объекты могут быть представлены с помощью пиктограмм. Кроме того, следует реализовать всплывающие меню для пиктограмм, отображаемых в списке. Это обеспечивает последовательность интерфейса при взаимодействии пользователя с пиктограммами.

Таким образом, работа с модифицируемым списком аналогична работе с файлами в окне каталога. Например, щелчок *Л* КМ на пиктограмме приводит к ее выбору. После выбора пиктограммы пользователь может включить в область выбора другие элементы списка.

Для модифицируемого списка поддерживаются клавиатурная навигация и выбор.

В качестве дополнительного параметра для стандартных модифицируемых списков также реализована возможность отображения вспомогательных графических элементов, используемая, в частности, для вывода информации о состоянии объектов, фигурирующих в списке. Такими элементами, например, могут быть флажки, помещаемые около пунктов списка.

5.3.5. МОДИФИЦИРУЕМОЕ ДЕРЕВО

Модифицируемое дерево (*Tree View Control*) является, в свою очередь, частным случаем модифицируемого списка, в котором содержимое отображается с учетом логического и иерархического соотношения между пунктами списка. В таком списке имеются кнопки, которые позволяют изменять форму представления структуры списка в целом и/или отдельных пунктов: они могут отображаться либо в развернутом, либо в свернутом виде (рис. 5.23). Модифицируемое дерево обычно используется в тех случаях, когда необходимо отобразить отношение между набором контейнеров или других иерархических элементов.

Для каждого узла дерева вы можете дополнительно включить пиктограмму с текстовой меткой. При изменении пользователем формы представления элемента списка (свертывании или развертывании) его пиктограмма может изменяться. В некоторых случаях полезно включить в обозначение пункта графический символ (например, все тот же флажок), чтобы отразить и информацию о состоянии данного пункта списка.

При создании модифицируемого дерева допускается также рисование линии, указывающих иерархическое соотношение между элементами в списке, а также между кнопками, используемыми для свертывания и развертывания списка.

Клавиатурная навигация реализуется с помощью клавиш управления курсором: переход между пунктами выполняется при нажатии клавиш <Вниз> и <Вверх>, а клавиши <Влево> и <Вправо> используются для перемещения вдоль конкретной ветви дерева.

Рис.5.23. Модифицируемое дерево

При нажатии клавиши <Вправо> может также выполняться развертывание пункта, если он отображен в свернутом виде. Для навигации по списку и выбора пунктов иногда используются текстовые клавиши; их работа, как и в других списках, основана на использовании техники синхронного сопоставления символов.

При использовании модифицируемого дерева в диалоговой панели следует дополнительно учитывать следующую рекомендацию. Если для пункта в списке определена встроенная команда, которая выполняется при нажатии клавиши <Enter> или по двойному щелчку ЛКМ, то на диалоговой панели должна существовать кнопка, запускающая ту же команду. Например, если вы используете двойной щелчок ЛКМ на элементе списка, чтобы отобразить его свойства, то на диалоговой панели должна существовать кнопка *Свойства*, запускающая соответствующую команду, когда фокус ввода находится на интересующем пользователя элементе.

5.4. ТЕКСТОВЫЕ ОБЛАСТИ

Как правило, окна содержат различные элементы интерфейса, которые обеспечивают отображение, ввод и редактирование текстовых величин. Некоторые из них представляют собой комбинацию области текстового ввода с элементами управления других типов.

В текстовой области не используются текстовые метки в качестве элементов управления. Тем не менее, вы можете дополнить ее статическим текстом, поясняющим назначение текстовой области и обеспечивающим индикацию ситуации, когда область недоступна. Если статический текст состоит из нескольких слов, при его написании целесообразно использовать рассмотренные ранее правила использования заглавных букв. Статический текст может также быть использован для определения клавиши доступа к текстовой области. При этом необходимо помнить, что при нажатии клавиши доступа фокус ввода должен устанавливаться на текстовую область, с которой метка связана, а не на сам статический текст. Вы можете также обеспечить клавиатурную навигацию в текстовой области, используя клавишу <Tab> и клавиши перемещения курсора.

В тех случаях, когда текстовая область используется для ввода ограниченного множества возможных значений (например, только чисел из заданного диапазона), следует тут же подтверждать правильность ввода или, наоборот, игнорировать неподходящие символы (возможно, выдавая пользователю соответствующее сообщение).

5.4.1. ТЕКСТОВЫЕ ПОЛЯ

Текстовое поле (*TextBox*) иногда называемое также *полем редактирования* или *полем ввода* — это прямоугольная область, в которой пользователь может вводить или редактировать текст (рис. 5.24). Она может содержать одну или несколько строк. Как правило, граница текстового поля обозначается явным образом, хотя на самом деле это необходимо только в тех случаях, когда текстовое поле расположено на панели инструментов или во вторичном окне.

Для стандартного текстового поля поддерживаются такие операции редактирования, как вставка и удаление символов, а также выделение фрагмента текста. Хотя индивидуальный шрифт для текстового поля обычно не поддерживается, при необходимости такая настройка может быть реализована.

В некоторых случаях текстовое поле может использоваться для вывода на экран текста, который не подлежит редактированию и предназначен только для чтения. Если такое ограничение устанавливается для стандартного текстового поля, система автоматически изменяет цвет его фона, чтобы указать пользователю на данную особенность.

Рис. 5.24. Стандартное текстовое поле

Для текстового поля поддерживается стандартная техника навигации и непрерывного выбора. Если текстовое поле содержит только одну строку, то для него отображается только горизонтальная полоса прокрутки; если же оно содержит более одной строки, то поле снабжается как горизонтальной, так и вертикальной полосами прокрутки.

Вы можете произвольным образом ограничить множество символов, разрешенных для ввода в данное текстовое поле. Кроме того, вы можете реализовать *автовыход*, установив фиксированную длину вводимых строк; это означает, что при вводе последнего символа в текстовое поле фокус ввода автоматически переходит на другой элемент управления. Например, вы можете определить шестисимвольный автовыход, чтобы облегчить ввод почтового индекса, или три текстовых поля с двухсимвольным автовыходом каждое, предназначенные для ввода даты (число, месяц, год). Хотя авто-выход обладает очевидными достоинствами, не рекомендуется злоупотреблять им, поскольку автоматический перенос фокуса ввода может вызвать замешательство у начинающего пользователя. Ограничьте применение автовыхода ситуациями, связанными с вводом большого объема данных (например, при заполнении экранных форм).

5.4.2. МНОГОСТРОЧНОЕ ТЕКСТОВОЕ ПОЛЕ

Многострочное текстовое поле (*Rich-Text Box*) (рис. 5.25) обеспечивает те же основные операции по работе с текстом, что и стандартное текстовое поле. Дополнительно для многострочного текстового поля поддерживается возможность индивидуальной настройки шрифта (тип шрифта, размер, цвет, начертание) для каждого символа, а также выбор формата параграфа (способ выравнивания, отступы, нумерация и т.д.). Кроме того, для него реализованы функции печати содержимого и вставки объектов с использованием технологии OLE.

Рис. 5.25. Многострочное текстовое поле

5.4.3. КОМБИНИРОВАННЫЙ СПИСОК

Комбинированный список (*Combo Box*) представляет собой объединение текстового поля и списка, как показано на рис. 5.26.

Текстовое поле и связанный с ним список имеют подчиненное отношение. По мере того, как текст набирается в текстовом поле, список перемещается в соответствии с вводимыми символами. И наоборот, когда пользователь выбирает пункт в списке, он автоматически помещается в текстовое поле и может быть изменен.

Интерфейс комбинированного списка представляет собой сочетание интерфейсов каждого из образующих его компонентов (стандартного текстового поля и списка), за исключением того, что клавиши <Вверх> и <Вниз> обеспечивают перемещение только по списку (от одного пункта к другому), а клавиши <Влево> и <Вправо> действуют исключительно в текстовом поле.

Рис. 5.26. Комбинированный список

5.4.4. ВЫПАДАЮЩИЙ КОМБИНИРОВАННЫЙ СПИСОК

Выпадающий комбинированный список (*Drop-down Combo Box*) объединяет свойства текстового поля и выпадающего списка (рис. 5.27). Данный элемент интерфейса является более компактным по сравнению с предыдущим и может использоваться в целях экономии пространства окна, но требует от пользователя дополнительных действий, чтобы открыть список.

В закрытом состоянии выпадающий комбинированный список похож на выпадающий список, за исключением того, что текстовое поле является интерактивным. Это означает, что пользователь может изменить содержимое текстового поля, и при нажатии клавиши <Enter> новое значение будет воспринято приложением.

Список открывается, когда пользователь нажимает кнопку *меню*, расположенную возле текстового поля. Повторное нажатие кнопки приводит к выбору пункта и закрытию списка.

Для повышения эффективности работы пользователей с выпадающим комбинированным списком рекомендуется создать для него метку в виде статической текстовой области и определить соответствующую клавишу доступа. Вы можете также разрешить использование клавиши <Tab> и клавиш управления курсором для навигации по блоку.

Рис. 5.27. Выпадающий комбинированный список (в закрытом и открытом состоянии)

Если фокус ввода находится на блоке (либо в текстовом поле, либо на панели списка), то при нажатии клавиш <Вверх> или <Вниз> (а также при использовании комбинаций клавиш <Вверх>+<Alt> и <Вниз>+<Alt>), список должен открываться.

Если список открыт и фокус ввода находится либо на нем, либо в текстовом поле, то при нажатии клавиши <Tab>, клавиши доступа, или комбинаций клавиш <Вверх>+<Alt> и <Вниз>+<Alt> список закрывается и фокус ввода переходит на другой элемент управления. Когда список закрывается, сохраните сделанный в нем выбор, если пользователь не нажимает кнопку *Отменить*. Клавиша <Esc> также закрывает список без сохранения нового выбора.

Если список открыт, взаимосвязь между текстовым полем и списком реализуется так же, как и в рассмотренном ранее комбинированном списке: когда пользователь набирает текст в текстовом поле, происходит перемещение списка в направлении соответствующего пункта; если же пользователь выбирает пункт в списке, то выбранный пункт отображается в текстовом поле и становится доступным для редактирования.

5.4.5. ДИСКРЕТНОЕ ТЕКСТОВОЕ ПОЛЕ

Дискретное текстовое поле (*Spin Box*) представляет собой текстовое поле, которое может принимать только ограниченный набор дискретных упорядоченных значений, образующих замкнутую последовательность. Данный элемент интерфейса является комбинацией текстового поля и специального элемента управления, который состоит из двух взаимосвязанных кнопок (он известен также как *элемент реверсивного управления— Up-Down Control*). В русскоязычных изданиях этот комбинированный элемент именуется по-разному: и «спин», и «спиннер», и просто «счетчик». Последний вариант является наименее удачным, поскольку его текстовое поле может содержать не только числовую, но и тестовую информацию (рис. 5.28.). Однако и пополнять все удлиняющиеся ряды "броузеров", "баннеров" и "дигитайзеров" тоже не очень хочется. Кроме того, необходимо учитывать, что элемент реверсивного управления может использоваться в сочетании с окном любого типа.

Рис 5.28. Дискретное текстовое поле

Когда пользователь щелкает ЛКМ в текстовом поле или на одной из кнопок, фокус ввода устанавливается в текстовом поле. После этого пользователь может либо непосредственно набрать текстовое значение, либо использовать кнопки, чтобы увеличить или уменьшить отображаемое в поле значение. Способ изменения зависит от конкретной реализации данного элемента.

При использовании дискретного текстового поля следует учитывать два основных момента.

Во-первых, далеко не во всех ситуациях шаг изменения значения, отображаемого в текстовом поле, будет очевидным для пользователя (например, при установке даты, представленной в форме <год/месяц/число>).

Во-вторых, изменение значения должно выполняться в обе стороны (и при уменьшении, и при увеличении) с одинаковым шагом.

Полезно также предусмотреть возможность циклического перехода между граничными значениями диапазона.

Вообще лучше всего сопроводить дискретное текстовое поле дополнительной информацией относительно особенностей его использования в каждом конкретном случае.

Чтобы предоставить пользователю возможность прямого клавиатурного доступа к дискретному текстовому полю, создайте для него (поля) метку в виде статической текстовой области и определите связанную с ней клавишу доступа. Вы можете также поддержать клавиатурный доступ на основе клавиши <Tab> и клавиш управления курсором. Если фокус ввода установлен на дискретном текстовом поле, пользователь может изменять текущее значение, нажимая клавиши <Вверх> или <Вниз>.

С помощью единственного элемента реверсивного управления может выполняться редактирование нескольких связанных текстовых полей, например, для установки времени, выраженного через часы, минуты и секунды. Кнопки влияют только на то текстовое поле, на котором в данный момент установлен фокус ввода.

5.4.6. СТАТИЧЕСКИЕ ТЕКСТОВЫЕ ОБЛАСТИ

Статические текстовые области (*Static Text Fields*) используются, как правило, для отображения информации, предназначенной только для чтения. В отличие от редактируемого текста, который может отображаться в обычном текстовом поле, статический текст нельзя выбрать (выделить цветом). Тем не менее, ваше приложение может изменить статический текст, чтобы отразить изменение состояния связанного с ним элемента интерфейса. Например, вы можете использовать статический текст, чтобы отобразить текущий маршрут директория или информацию о состоянии объекта (номер страницы, время и дату создания и т.д.).

Рис. 5.29 иллюстрирует применение редактируемой статической текстовой области.

Рис. 5.29. Статическая текстовая область

Вы можете также использовать статические текстовые области в качестве меток или пояснительной информации для других элементов интерфейса. Использование статической текстовой области в качестве метки позволяет определить клавишу доступа для элемента интерфейса, с которым она связана. В этом случае необходимо, чтобы фокус ввода переходил на этот элемент интерфейса, а не на статическую область. Не забывайте также включать двоеточие в конце текста. Этот символ не только помогает пользователю увязать текст с элементом интерфейса, он также используется экранными утилитами обзора.

5.4.7. ПОЛЕ НАЗНАЧЕНИЯ ГОРЯЧИХ КЛАВИШ

Поле назначения *горячих клавиш*, известных также как *клавиши-акселераторы* (*Shortcut key Input Control*) — это специальный тип текстового поля, предназначенный для того, чтобы обеспечить пользователю возможность назначения (переопределения) горячих клавиш или комбинации таких клавиш (рис. 5.30). Используйте это поле только в тех случаях, когда вы предоставляете пользователю право модифицировать по его усмотрению назначение клавиш-акселераторов, поддерживаемых вашим приложением. Поскольку горячие клавиши непосредственно иницируют команду, они обеспечивают более эффективный интерфейс для общих или часто используемых действий.

Поле назначения горячих

Рис. 5.30. Поле назначения горячих клавиш

Для данного типа текстового поля существует возможность определить запрещенные клавиши или комбинации клавиш, с тем чтобы гарантировать корректное назначение клавиш пользователем. Предусмотрена также возможность автоматической модификации вводимой информации в тех случаях, когда пользователь пытается использовать запрещенную клавишу.

Подобно большинству текстовых полей, данный элемент по умолчанию не имеет собственной метки, поэтому целесообразно дополнять его статической текстовой областью, чтобы обеспечить его идентификацию и назначить соответствующую клавишу доступа. С целью поддержки клавиатурного доступа может также использоваться клавиша <Tab>.

5.5. ПАНЕЛЬ ИНСТРУМЕНТОВ И СТРОКА СОСТОЯНИЯ

Панель инструментов (*Toolbar*) и строка состояния (*Status Bar*) — это специальные компоненты пользовательского интерфейса, предназначенные для создания функционально-ориентированных наборов элементов управления. Панель инструментов содержит, как правило, элементы управления, обеспечивающие быстрый доступ к наиболее часто используемым командам или свойствам объектов. Различают несколько типов панелей инструментов, ориентированных на применение в специфических приложениях. Такие специализированные панели иногда имеют собственные названия (рис. 5.31).

Рис. 5.31. Примеры панелей инструментов

Строка состояния — это специальная область внутри первичного окна (обычно в нижней его части), предназначенная для вывода информации о текущем состоянии объектов или процессов, представленных в окне, а также любой другой контекстной информации, например, о состоянии клавиатуры (рис. 5.32). Вы можете также использовать строку состояния, чтобы обеспечить вывод справочных сообщений о выбранном пункте меню или кнопке панели инструментов. Подобно панели инструментов, строка состояния может содержать элементы управления; тем не менее, рекомендуется включать в нее информацию «только для чтения» или не интерактивные элементы.

Доступ к элементам управления, включенным в панель инструментов или строку состояния, реализуется либо с помощью мыши, либо через другие стандартные средства взаимодействия с этими элементами. Может быть также реализовано клавиатурное взаимодействие, основанное на использовании горячих клавиш или клавиш доступа. Если элемент панели инструментов или строки состояния не имеет текстовой метки, средства клавиатурного доступа для него оказываются неэффективными. Кроме того, если какая-либо клавиша доступа уже используется в первичном окне, она не может применяться для работы с панелью инструментов. Например, если полоса меню первичного окна уже использует некоторую клавишу доступа, то все события, связанные с этой клавишей, будут обрабатываться применительно к полосе меню.

Рис. 5.32. Варианты реализации строки состояния

Когда пользователь взаимодействует с элементом управления, расположенном на панели инструментов или в строке состояния, любое изменение непосредственно относится к текущему выбору (например, если кнопка панели инструментов предназначена для изменения начертания шрифта в тексте, то при нажатии этой кнопки текст немедленно изменяется); никакое дополнительное подтверждение не требуется. Единственное исключение составляет случай, когда использование элемента управления (например, кнопки), требует от пользователя ввода дополнительной информации; соответствующая операция не может быть реализована, пока пользователь не введет требуемую информацию. Примером такого исключения является выбор объекта или установка значения параметра с помощью диалоговой панели.

Для тех элементов панели инструментов или строки состояния, которые не имеют текстовой метки, обязательно должна использоваться всплывающая подсказка. Система поддерживает создание всплывающей подсказки для элементов стандартной панели инструментов.

С целью обеспечения максимальной гибкости в работе пользователей панели инструментов и строки состояния должны допускать индивидуальную настройку. Простейший вариант такой настройки — возможность скрыть (не отображать) эти элементы интерфейса. В более сложных приложениях пользователю предоставляется право изменять или перестраивать элементы, включенные в панели инструментов или строки состояния.

Рекомендуется также обеспечивать возможность отображения каждой кнопки на панели инструментов по крайней мере двух размеров: 24 на 22 и 32 на 30 пикселей. Если для обозначения кнопок используются графические метки, они должны иметь размеры не более чем 16 на 16 и 24 на 24 пикселей соответственно.

Продумайте также возможность создания перемещаемой пользователем панели инструментов. Хотя панель инструментов по умолчанию отображается как **пристыкованная** к определенному краю окна или панели, в которых она используется, разрешите пользователю перемещать ее к другому краю или отображать в виде окна палитры (рис. 5.33).

Рис. 5.33. Перемещаемая панель инструментов

Для перемещения панели инструментов необходимо установить указатель на любом свободном участке панели и, нажав ЛКМ, перетащить ее на новое место. Если новая позиция находится в пределах **горячей зоны края окна**, приложение должно автоматически пристыковать панель к краю окна (после того, как пользователь отпустит кнопку мыши). Если же новая позиция находится вне горячей зоны края окна, панель инструментов преобразуется в окно палитры. Чтобы вновь пристыковать панель к краю окна, пользователь должен установить указатель на полосе заголовка окна палитры и, нажав ЛКМ, перемещать это окно, пока указатель не окажется в горячей зоне; когда пользователь отпустит кнопку мыши, панель инструментов отображается в пристыкованном состоянии.

Во время перемещения панели инструментов обеспечьте визуальную обратную связь, отобразив, например, перемещение контура панели инструментов. Когда указатель входит в горячую зону окна, измените соответствующим образом контур панели инструментов, чтобы пользователь мог визуально контролировать операцию (рис. 5.34).

Рис. 5.34. Визуализация перемещения панели инструментов

Вы можете также поддержать для панели инструментов настройку пользователем других параметров, таких например, как изменение размеров панели инструментов, стыковку нескольких панелей, размещение их в требуемом порядке и т.п.

Не рекомендуется включать в состав панели инструментов и строки состояния элементы, к которым пользователь не может получить доступ другим путем. Кроме того, всегда сохраняйте текущую позицию, размер и другую информацию о панели инструментов и строке состояния, с тем чтобы они могли быть восстановлены в прежнем виде, когда пользователь вновь открывает окно приложения.

Для строки состояния поддерживается также возможность включения в нее *регулятора* для калибровки размера окна, описанного в главе 4. При реализации этой возможности следует иметь в виду следующее.



















Нельзя одновременно отображать регулятор в двух позициях: в строке состояния и на пересечении полос прокрутки окна (в его стандартной позиции). Когда на экране отображается строка состояния, содержащая регулятор, то из стандартной позиции он должен быть убран; если же пользователь убирает строку состояния, восстановите регулятор на стандартной позиции.































СТАНДАРТНЫЕ КНОПКИ ПАНЕЛИ ИНСТРУМЕНТОВ

Табл. 5.6 иллюстрирует стандартные форматы кнопок «общего назначения» которые могут быть использованы в любом приложении.

Таблица 5.7.

Форматы стандартных кнопок панели инструментов

Форм ат 16x16	Форм ат 24x24	Функция
		Создать (New)
		Открыть (Open)
		Сохранить (Save)
		Печать (Print)
		Предварительный просмотр (Print Preview)
		Отменить (Undo)
		Повторить (Redo)
		Вырезать (Cut)
		Вставить (Paste)

		Копировать (Copy)
		Удалить (Delete)
		Найти (Find)
		Заменить (Replace)
		Полужирный [курсив] (Bold)
		Курсив (Italic)
		Подчёркнутый [шрифт] (Underline)
		Свойства (Properties)
		Что Это? [режим контекстно-зависимой подсказки] (What's This)
		Открыть родительскую папку (Open parent folder)
		Отобразить как крупные пиктограммы (View as large icons)
		Отобразить как мелкие пиктограммы (View as small icons)
		Отобразить в виде списка (View as list)
		Отобразить в виде таблицы (View as details)
		Выделить (выбрать) область (Region selection tool)

Применяйте приведенные выше изображения только для обозначения описанных функций. Последовательность в их использовании позволяет пользователю применять знания и навыки, полученные при работе с одним программным продуктом, в любом другом. Если же одна из стандартных пиктограмм используется для обозначения другой функции, это может сбить пользователя. Разрабатывая собственные кнопки для панели инструментов, следуйте стандартным системным соглашениям, рассмотренным ранее.

5.6. ДРУГИЕ ЭЛЕМЕНТЫ ГРАФИЧЕСКОГО ИНТЕРФЕЙСА

Система поддерживает наряду с рассмотренными выше и другие элементы графического интерфейса пользователя, которые могут быть полезны при разработке приложений.

5.6.1. ГРУППИРУЮЩИЙ БЛОК

Группирующий блок (*Group Box*) — это специальный элемент, который применяется для визуального объединения нескольких элементов интерфейса (в том числе разнотипных). Он представляет собой прямоугольную рамку, снабженную текстовой меткой, которая окружает группу элементов интерфейса, как показано на рис. 5.35. Группирующий блок обычно не предусматривает возможность непосредственного взаимодействия с пользователем, тем не менее вы можете предоставить навигационный доступ к элементам в группе, используя клавишу <Tab>, или назначая клавишу доступа к метке группирующего блока.

Рис. 5.35. Группирующий список

Для элементов группы могут быть созданы индивидуальные метки, связанные с меткой группирующего блока. Например, группирующий блок, имеющий метку *Выравнивание*, может содержать переключатели с метками *Влево*, *Вправо* и *По центру*. Если метка группирующего блока состоит из нескольких слов, при ее написании применяются стандартные правила использования заглавных букв.

5.6.2. ЗАГОЛОВКИ СТОЛБЦОВ

Заголовки столбцов (*Column Headings*), иногда называемые также управляемыми заголовками, используются для обозначения столбцов данных, содержащих формируемую вашим приложением текстовую или числовую информацию (например, при создании электронной таблицы). При этом стандартный заголовок может быть разделен на произвольное количество частей (элементов), как показано на рис. 5.36. Управляемые заголовки применяются, в частности, в модифицируемых списках, рассмотренных выше.

Рис. 5.36. Заголовок, состоящий из четырёх элементов

Метка каждого элемента заголовка может содержать текст и графический образ. Графический образ служит, как правило, для вывода дополнительной информации, например, для указания направления сортировки.

Элемент заголовка может быть определен таким образом, что по внешнему виду и по поведению он будет аналогичен кнопке управления, при нажатии которой выполняется связанная с ней функция. Например, для столбца, содержащего список данных, примером такой функции является сортировка списка по заданному признаку. Кроме того, для заголовка поддерживается использование правой кнопки мыши: с ее помощью вызывается всплывающее меню, содержащее специфические команды для работы с данными, такие, например, как *Сортировка по возрастанию* и *Сортировка по убыванию*.

Для заголовка столбца также поддерживается возможность перемещения пользователем границ элементов заголовка с целью выбора наиболее подходящей в

данный момент ширины каждого столбца. В качестве дополнительного средства настройки вы можете использовать двойной щелчок ЛКМ на границе заголовка столбца, с помощью которого будет запускаться команда форматирования столбца, например, автоматическая калибровка столбца по наиболее широкому элементу в этом столбце.

5.6.3. ЭТИКЕТКА ВКЛАДКИ

Этикетка вкладки (*Tab*) по форме и по назначению аналогична разделителю в картотеке или в записной книжке (рис. 5.37). Вы можете использовать этот элемент, чтобы создать несколько логически законченных страниц или секций в пределах одного окна (такие страницы обычно называют **вкладками**).

Рис. 5.37. Этикетки вкладок

Этикетка вкладки может содержать текстовую или графическую информацию, либо их комбинацию. Обычно размер этикетки устанавливается автоматически, в соответствии с размером отображаемой на ней метки, однако вы можете определить фиксированную ширину для создаваемых этикеток. Для вывода текстовой информации на этикетке рекомендуется использовать системный шрифт и соблюдать те же правила применения заглавных букв, которые были описаны выше. Если для обозначения этикетки используется только графический образ, создайте для такой этикетки всплывающую подсказку.

По умолчанию система поддерживает создание только одного ряда (линейки) этикеток, хотя при необходимости в пределах одного окна можно создать несколько линеек этикеток, либо реализовать одну прокручивающуюся линейку. Несмотря на то, что эти средства позволяют экономить пространство экрана, следует ограничивать их применение, поскольку они существенно усложняют работу пользователя.

Когда пользователь щелкает ЛКМ на этикетке вкладки, фокус ввода перемещается на эту этикетку. Для перехода с одной этикетки на другую могут использоваться клавиши <Влево> или <Вправо>, либо комбинация клавиш <Ctrl>+<Tab>. Дополнительно вы можете определить клавиши доступа для каждой этикетки. Если пользователь переходит с одной вкладки на другую с помощью этикеток, вы можете устанавливать фокус ввода на конкретном элементе вкладки, с которого пользователю следует начать работу. Если же в этом нет необходимости, оставьте фокус ввода на самой этикетке.

5.6.4. ПОЛОСЫ ПРОКРУТКИ

Полоса прокрутки (*ScrollBar*) представляет собой прямоугольную область, содержащую стрелки, которые указывают разрешенное направление прокрутки, и ползунок, величина и положение которого отражают размер невидимой части объекта (рис. 5.38). Как стрелки, так и ползунок являются интерактивными элементами (то есть они реагируют на воздействие пользователя). Стрелка полосы прокрутки представляет собой кнопку, при нажатии которой информация в окне перемещается в соответствующем направлении на одну дискрету;



Рис. 5.38. Компоненты полосы прокрутки

величина шага прокрутки устанавливается разработчиком приложения.

Общая характеристика полосы прокрутки как средства взаимодействия пользователя с окном была приведена в разделе «Первичное окно». Теперь поговорим о том, какие факторы следует учитывать при проектировании этого элемента.

Управление полосами прокрутки должно обеспечивать плавное и непрерывное перемещение информации в окне. Если она не может быть перемещена более в данном направлении, следует сделать недоступной стрелку, соответствующую этом. направлению. Кроме того, при достижении границы просматриваемой информации ползунок должен находиться непосредственно возле этой стрелки.

Если при выполнении прокрутки пользователь удерживает указатель на стрелке при нажатой кнопке мыши, перемещение информации должно выполняться до и пор, пока кнопка не будет отпущена.

Ползунок выполняет две основные функции:

- отображает размер и расположение просматриваемого в окне фрагмента относительно границ всего документа;
- служит средством ускоренной прокрутки информации.

Для реализации первой функции размер ползунка может изменяться. При этом его первоначальный размер определяется соотношением между объемом видимого в окне фрагмента и объемом всего документа (рис. 5.39).

Рис. 5.39. Выбор размера ползунка

Для ускоренного перемещения информации с помощью ползунка необходимо определить соотношение между величиной смещения ползунка вдоль полосы прокрутки и величиной перемещаемого фрагмента.

Еще один способ ускоренной прокрутки информации основан на использовании чувствительных областей полосы прокрутки. **Чувствительная область** — это часть полосы прокрутки между ползунком и стрелкой. При реализации этого способа щелчок ЛКМ на чувствительной области означает перемещение отображаемой информации в направлении, задаваемом ближайшей стрелкой. При этом величина перемещения устанавливается равной размеру видимой области по соответствующему измерению. Например, щелчок ЛКМ в нижней части чувствительной области вертикальной полосы прокрутки приводит к пролистыванию отображаемой информации на один экран вниз (рис. 5.40).

Рис. 5.40. Прокрутка при воздействии на чувствительную область

При выполнении такой прокрутки ползунок перемещается на расстояние, пропорциональное величине шага прокрутки. Если пользователь удерживает кнопку мыши в нажатом состоянии, выполняется непрерывная прокрутка на произвольное количество шагов, вплоть до границы документа.

В некоторых случаях удобнее интерпретировать перемещение ползунка в крайнюю позицию полосы прокрутки как переход на конец данных, а не в конец обрабатываемого документа. Например, стандартная электронная таблица может иметь 65000 строк, из них данные могут быть введены только в первые 50 строк. В такой ситуации пользователю значительно важнее иметь возможность быстро

перейти к последней строке, содержащей данные, а не к последней строке электронной таблицы.

Горизонтальные и вертикальные полосы прокрутки могут использоваться для создания собственных перемещаемых областей (для первичного окна и списков полосы прокрутки создаются автоматически). Используйте полосы прокрутки только для перемещения содержимого соответствующей области. Если же требуется предоставить пользователю средства для установки или регулировки каких-либо величин, применяйте другие элементы интерфейса, например, ползунковый регулятор или дискретную текстовую область.

При реализации взаимодействия пользователя с полосами прокрутки придерживайтесь приведенных ранее рекомендаций относительно блокировки стрелок полосы прокрутки: заблокируйте соответствующую стрелку, когда пользователь оказывается в начале или в конце просматриваемого документа.

Хотя фокус ввода может быть установлен на полосе прокрутки, старайтесь избегать этого, поскольку такая возможность не поддерживается для полос прокрутки первичных окон и списков, с которыми привык работать пользователь.

5.6.5. ПОЛЗУНКОВЫЙ РЕГУЛЯТОР

Ползунковый регулятор (*Slider*) используется для установки или изменения величин, имеющих непрерывный диапазон значений, например, таких как объем или яркость.

Для этого элемента имеют место практически те же терминологические трудности, что и для дискретной текстовой области. Различные авторы используют для его обозначения самые разные наименования: «слайдер», «движок», «ползунок» и даже «бегунок». Вместе с тем, в технике для подобных устройств уже давно используется термин «ползунковый регулятор».

Ползунковый регулятор состоит из **шкалы**, которая определяет диапазон возможных значений регулируемой величины, и **индикатора**, положение которого показывает текущее значение величины (рис. 5.41); индикатор используется также для установки нового значения.

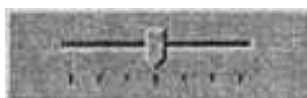


Рис. 5.41. Ползунковый регулятор

Поскольку ползунковый регулятор не имеет собственной метки, используйте в качестве нее статическую текстовую область. Вы можете также добавить текст и графику, чтобы помочь пользователю уяснить особенности установки и регулировки значений конкретной величины.

Для ползункового регулятора поддерживаются многие дополнительные параметры настройки: ориентация (вертикальная или горизонтальная), размеры индикатора, дискретность изменения величины и т.д.

Пользователь перемещает индикатор, перетаскивая его мышью в конкретную позицию, или щелкая ЛКМ в той точке шкалы, куда следует переместить индикатор.

Чтобы обеспечить клавиатурное взаимодействие, используйте клавишу <Tab> и определите клавишу доступа для статической текстовой области, применяемой в качестве метки. Когда фокус ввода установлен на ползунковом регуляторе, для пере-

мещения индикатора могут использоваться клавиши управления курсором; при этом направление движения индикатора соответствует направлению действия клавиши.

5.6.6. ИНДИКАТОР СОСТОЯНИЯ ПРОЦЕССА

Индикатор состояния процесса (*Progress Indicator*) - это элемент интерфейса, который обычно используется для того, чтобы отобразить ход выполнения какой-либо длительной операции (процесса). Он состоит из прямоугольной зоны, которая «заполняется» слева направо, как показано на рис. 5.42.

Рис. 5.42. Индикатор состояния процесса

Поскольку индикатор только отображает информацию, то он не является интерактивным элементом. Поэтому во многих случаях полезно снабдить индикатор статическим текстом, поясняющим его назначение. Текст, естественно, должен располагаться вне индикатора.

Используйте индикатор в качестве средства обратной связи для длинных операций или фоновых процессов (как дополнение к изменению формы курсора). Индикатор состояния процесса предоставляет пользователю информацию о состоянии процесса в более наглядной форме. Вы можете также использовать его, чтобы отразить протекание фонового процесса, освободив курсор для отображения состояния основного процесса и для взаимодействия с ним.

Если индикатор предполагается использовать в окне сообщения или в строке состояния, продумайте форму представления отображаемой операции или процесса.

5.6.7. ВСПЛЫВАЮЩАЯ ПОДСКАЗКА

Всплывающая подсказка (*Tooltip Control*) — это небольшое окно, содержащее пояснительный текст, которое появляется на экране, когда пользователь устанавливает указатель на один из элементов интерфейса, как показано на рис. 5.43. Всплывающая подсказка появляется после короткой задержки и автоматически удаляется, когда пользователь активизирует выбранный элемент или перемещает курсор в другую позицию экрана (либо по истечении установленного интервала времени).

Рис. 5.43. Всплывающая подсказка

Система отображает всплывающую подсказку ниже и правее курсора, по автоматически позиционирует ее, если она при этом может выйти за пределы экрана. Исключение составляют текстовые поля, для которых всплывающая подсказка отображается снизу и по центру поля, к которому она относится (рис. 5.44).

Рис. 5.44. Вывод всплывающей подсказки для текстового поля

Для реализации указанной особенности имеется специальный параметр настройки всплывающей подсказки.

5.6.8. КОЛЛЕКЦИИ

Коллекция (*Well*) — это специальный элемент интерфейса, подобный в использовании группе переключателей, с той лишь разницей, что обеспечивает выбор пользователем элементов графического оформления, таких например, как цвет, тип линии, пиктограмма и т.п. (рис. 5.45).

Рис. 5.45. Коллекция для выбора цвета

Подобно переключателям, Коллекция используется для представления величин, которые имеют два или более значений, и обеспечивает логически упорядоченное размещение этих значений в виде компактной группы. Если Коллекция используется в вашем приложении как интерактивный элемент, применяйте для разделения значений в группе тот же тин рамки, что и для флажков или текстовых полей. Когда пользователь выбирает конкретное значение в группе, отметьте его специальной выделяющей рамкой.

Как уже было сказано, техника взаимодействия пользователя с Коллекцией аналогична его работе с переключателями. Когда пользователь щелкает ЛКМ на одном из элементов Коллекции, соответствующее свойство редактируемого графического объекта принимает выбранное значение. Чтобы идентифицировать Коллекцию и определить для нее клавишу доступа, используйте группирующий блок или статический текст. Для перемещения фокуса ввода внутри коллекции используются клавиши управления курсором.

5.6.9. ОБЛАСТЬ СООБЩЕНИЙ

На противоположной от кнопки *Пуск* стороне Панели задач отображается специальная область, называемая **областью сообщений** (*Status Notification* или *System Tray*). Приложение может поместить здесь специальный индикатор или сообщение, уведомляющее пользователя о какой-либо ситуации; выведенное сообщение сохраняется в области сообщений даже тогда, когда приложение находится в неактивном состоянии.

Поскольку Панель задач — коллективный ресурс, используемый всеми активными приложениями, в области сообщений следует выводить только такую информацию, которая носит «глобальный» характер или необходима пользователям при работе с другими приложениями.

Для отображения информации в области сообщений рекомендуется использовать графические символы, поставляемые вашим приложением (рис. 5.46).

Рис. 5.46. Отображение информации в области сообщений

При отображении информации в области сообщений следует придерживаться приведенных ниже рекомендаций:

- Обеспечьте появление на экране всплывающего окна, которое содержит дополнительную информацию или средства управления для объекта, представленного индикатором в области сообщений; для вызова окна обычно используется щелчок ЛКМ на изображении индикатора. Всплывающее окно должно быть расположено таким образом, чтобы пользователю было удобно управлять им (рис. 5.466). Избегайте

применения других типов вторичных окон, поскольку они требуют явного действия пользователя для их закрытия. Если нет необходимости в выводе дополнительной информации, всплывающее окно создавать не следует.

- Для объекта, представленного индикатором, должно поддерживаться всплывающее меню, появляющееся на экране по щелчку ПКМ на изображении индикатора (рис. 5.46в). Это меню должно содержать основные команды панели свойств соответствующего объекта (или других связанных с ним окон). Например, если индикатор в области сообщений связан с *Лазерным проигрывателем*, то его всплывающее меню может содержать команду *Изменить уровень громкости*.

- Всплывающее меню может содержать predetermined команду, которая выполняется по двойному щелчку Л КМ на изображении индикатора.

- Создайте всплывающую подсказку для индикатора, поясняющую его назначение.

- Предоставьте пользователю возможность не отображать индикатор в области сообщений, включив соответствующее свойство в панель свойств объекта, представленного индикатором. Это позволит пользователю самому определять, какой индикатор вынести в область сообщений. При этом следует обеспечить альтернативные средства представления информации, отображаемой в области сообщений.

5.7. ВЫБОР ВИЗУАЛЬНЫХ АТТРИБУТОВ ОТОБРАЖАЕМОЙ ИНФОРМАЦИИ

Удачный (в смысле *продуманный*) выбор визуальных атрибутов отображаемой на экране информации—это значительно больше, чем просто красивое внешнее оформление приложения. Качество визуального проектирования в значительной степени влияет и на психофизиологическое состояние пользователя, и на эффективность его работы в целом. Вследствие этого даже достаточно мощный по своим возможностям программный продукт может оказаться недостаточно функциональным, если визуальное представление его интерфейса не удовлетворяет соответствующим требованиям. Именно поэтому при реализации больших и ответственных проектов программисты в последнее время все чаще прибегают к услугам профессиональных дизайнеров. В частности, уже получил практически официальный статус такой термин, как Web-дизайн.

Разумеется, качество визуального проектирования в наибольшей степени проявляется при использовании графического пользовательского интерфейса. И хотя общая концепция внешнего облика приложения во многом определяется его предназначением, для всех GUI-приложений справедливы следующие положения.

- Все графические элементы приложения создают единую визуальную среду; поэтому обязательным этапом визуального проектирования является выбор упомянутой выше общей концепции внешнего облика приложения;

- При использовании графического интерфейса каждый графический элемент и реализуемая им функция тесно взаимосвязаны и эта связь должна быть интуитивно понятна пользователю.

- Исходя из этого, в рамках визуального проектирования (дизайна) интерфейса приложения должны быть решены такие проблемы, как эффективное использование пространства экрана, выбор адекватной формы представления объектов, цветовая палитра и композиция графических элементов, а также выбор средств привлечения внимания пользователя к тем или иным элементам информации, отображаемой на экране.

5.7.1. КОМПОЗИЦИЯ И ОРГАНИЗАЦИЯ

Исследования в данной области (как, впрочем, и практический опыт) свидетельствуют о том, что человек значительно лучше воспринимает визуальную информацию, если она соответствующим образом организована в пространственном отношении. Мы просматриваем информацию на экране так же, как и любые другие формы информации. Взгляд всегда привлекают в первую очередь цветные элементы, а не черно-белые, изолированные (отдельно стоящие), а не сгруппированные, и, наконец, графические, а не текстовые.

Исходя из этого, при проектировании визуальных элементов интерфейса приложения целесообразно опираться на определенные принципы, основными из которых являются следующие:

- иерархическая организация отображаемой информации;
- визуальное выделение наиболее важных элементов;
- сбалансированность структуры экрана;
- визуальное объединение логически взаимосвязанных элементов;
- обеспечение удобочитаемости и логической согласованности отображаемой информации;
- использование единых подходов к визуализации отображаемой информации не только в рамках приложения, но и рабочей среды в целом (принцип интеграции).

Иерархическая организация информации

Принцип иерархической организации информации означает размещение информации с учетом ее значения относительно других визуальных элементов приложения. Результат этого упорядочения влияет на реализацию всех других принципов визуального представления информации. С точки зрения работы пользователя иерархическая организация информации определяет доступность тех или иных данных и последовательность выполнения решаемой задачи. Для успешной реализации этого принципа необходимо получить ответы на следующие вопросы:

1. Какая информация наиболее важна для пользователя?

Другими словами, что является наиболее важным с точки зрения пользователя при работе с вашим приложением. Например, наиболее важными аспектами для него могут быть создание или поиск документа.

2. Что пользователь хочет или должен делать в первую, во вторую, в третью очередь?

3. Что пользователь должен видеть на экране в первую, во вторую, в третью очередь?

Визуальное выделение наиболее важных элементов

При реализации принципа привлечения внимания пользователя должны решаться две задачи: во-первых, выбор на каждом шаге работы некоторой основной идеи, наиболее важной для выполнения этого шага; во-вторых, соответствующее представление и размещение реализующих эту идею элементов.

В силу указанных выше психофизиологических особенностей люди обращают взгляд в первую очередь на верхний левый угол просматриваемой области или на ту ее часть, которая визуально отличается от других. Исходя из этого, имеет смысл

размещать наиболее важную информацию (или узловой элемент) либо в верхнем левом углу экрана, либо в окне, снабженном специальными атрибутами.

Сбалансированность структуры экрана

Принцип сбалансированности структуры экрана — один из наиболее важных принципов визуального проектирования. Он предполагает, с одной стороны, рациональное использование пространства экрана, а с другой — такое размещение информации, при котором на экране в каждый момент времени представлена только та ее часть, которая действительно необходима для выполнения очередного шага задания пользователя.

Визуальное объединение логически взаимосвязанных элементов

Данный принцип можно рассматривать как развитие предыдущего. Визуальное объединение логически взаимосвязанных элементов способствует уяснению пользователем того, как именно представленная на экране информация и элементы управления связаны с выполняемым шагом задания и друг с другом. Например, если в диалоговой панели имеется кнопка, которая влияет на содержимое списка, целесообразно поместить их рядом.

Удобочитаемость и логическая согласованность отображаемой информации

Имеется в виду, что любая информация (не только текстовая) должна быть выражена в компактной и вместе с тем в доступной форме; кроме того, пользователь должен быть способен уяснить, как она связана с предыдущими и последующими шагами задания.

При реализации данного принципа полезно получить ответы на следующие вопросы:

- Можно ли представленную на экране идею или понятие выразить более просто?
- Насколько легко сможет пользователь выполнить данный шаг?
- Действительно ли все отображаемые элементы необходимы для выполнения этого шага?

Интеграция

Если интерфейс приложения визуально согласуется с интерфейсом системы и интерфейсом других приложений, значительно легче обеспечить пользователю последовательную и предсказуемую рабочую среду. Для того, чтобы осуществить этот принцип, продумайте следующие вопросы:

- Насколько оказывается согласованным экранное представление различных компонентов приложения в процессе работы пользователя?
- Как соотносятся визуальные параметры приложения с системным интерфейсом и интерфейсом других приложений?

5.7.2. ЦВЕТ

Цвет является одним из важнейших визуальных атрибутов интерфейса. Поскольку цвет имеет способность привлекать взгляд, используйте его для идентификации тех элементов интерфейса, на которые вы хотите обратить

внимание пользователя (например, для выделения текущего выбора). Цвет также имеет ассоциативный аспект; во многих случаях объекты одного цвета воспринимаются людьми как взаимосвязанные. Цветовая гамма может также оказывать определенное эмоциональное или психологическое воздействие; не зря, например, разделяют цвета на «холодные» и «теплые».

Области, фон которых представлен более теплыми оттенками красной части спектра, кажутся крупнее, чем области, цвет фона которых смещен к голубой части. Объекты экрана на белом фоне или на фоне цвета из средней части спектра кажутся ярче и легче воспринимаются при разном внешнем освещении. Наибольший контраст между двумя областями экрана достигается в том случае, если цвет фона одной из них — черный или близок к любой из границ спектра, а цвет фона другой — белый или взят из средней части спектра. Данные рекомендации справедливы и для соотношения между цветами символов и фона. Следует учитывать, что некоторые комбинации цвета, например голубой цвет символов на красном фоне, неприятны для глаз. Различные психологические состояния человека характеризуются различными предпочитаемыми (наиболее приятными) цветами. Помимо этого отмечена и обратная зависимость — влияние цвета на настроение и работоспособность. В работах, посвященных исследованию психологического воздействия различных цветов, приводятся такие данные:

голубой — успокаивает;
 красный — волнует и утомляет;
 зеленый — настраивает на добродушный и безынициативный лад;
 желтый — веселый, оптимистичный, вызывает легкомысленный настрой;
 оранжевый — раскрепощает фантазию;
 фиолетовый — губителен для глаз, цвет зависти, тревоги, неудовлетворенности;
 коричневый — угнетает умственную активность;
 черный — мрачный, способствует возникновению головных болей, но снижает число ошибок.

Таким образом, непродуманное использование цвета может вызвать у пользователя негативное эмоциональное состояние или даже отвлекать его от работы, мешая сосредоточиться на выполняемом задании.

Кроме того, при выборе цветовой палитры следует учитывать целый ряд дополнительных факторов:

- Хотя вы можете использовать цвет, чтобы отразить связь между теми или иными объектами, ассоциация между выбранным вами цветом и конкретным значением не всегда будет очевидна или легко узнаваема для пользователя.
- Цвет является очень субъективной характеристикой (помните народную мудрость: «На вкус и на цвет товарищей нет?»); поэтому то, что нравится вам, совсем не обязательно будет приятно пользователям.
- Некоторые пользователи могут иметь проблемы с цветовосприятием (около 9 процентов взрослого мужского населения имеют то или иное отклонение в восприятии цвета).

В силу приведенных выше обстоятельств цвет должен использоваться только как дополнительная форма передачи информации.

Еще один вывод состоит в том, что целесообразно в рамках одного приложения использовать ограниченное множество цветов, причем предпочтение следует отдавать приглушенным, пастельным тонам.

Поскольку цвет является субъективным фактором, разрешите пользователю самому настраивать цветовую гамму, где это возможно. Для стандартных элементов

интерфейса система обеспечивает стандартные цветовые схемы. Если при создании приложения вы будете их использовать в качестве основы, то это позволит избежать включения в интерфейс дополнительных элементов управления; кроме того, указанный подход гарантирует, что ваши собственные визуальные элементы не будут «выпадать» из общего ряда, когда пользователь изменяет системные цвета. Создавая ваш собственный интерфейс для изменения цветов, учитывайте сложность выполняемого задания и уровень подготовленности пользователя.

5.7.3. ШРИФТ

Подобно другим визуальным элементам, шрифты способствуют организации информации и созданию определенного настроения. Изменяя размер и плотность шрифта, вы можете указать пользователю на степень важности той или иной информации и порядок, в котором она должна быть прочитана.

На экранах стандартных мониторов шрифты обычно менее разборчивы, чем на отпечатанной странице. Старайтесь не использовать курсив (Italic) и рубленый шрифт (Serif), поскольку они трудны для чтения, особенно при низком разрешении монитора.

Ограничьте количество применяемых шрифтов и стилей. По возможности, используйте стандартный системный шрифт для общих элементов интерфейса. Это обеспечивает визуальную согласованность между интерфейсом вашего приложения и интерфейсом рабочей среды и, кроме того, делает ваш интерфейс легче масштабируемым. Поскольку многие элементы интерфейса могут модифицироваться пользователем, проверьте системные параметры для встроенного системного шрифта и установите аналогичные значения для вашего приложения.

5.7.4. «МНОГОМЕРНОСТЬ» ЭКРАНА

При изображении многих элементов интерфейса полезно использовать перспективу, подсветку и затенения с целью обеспечения эффекта трехмерного образа. Это способствует повышению функциональности интерфейса и наглядности обратной связи при работе пользователя с компьютерными аналогами реального мира. Например, кнопки изображаются таким образом, чтобы при их выборе у пользователя появлялась иллюзия, что кнопка действительно нажата.

При создании эффектов освещенности/затенения по умолчанию принимается, что гипотетический источник света находится в верхнем левом углу экрана.

Разрабатывая ваши собственные визуальные элементы, будьте осторожны: не переусердствуйте в использовании «объемных» образов, поскольку изображение каждого «трехмерного» объекта занимает на экране значительно больше места, чем его «плоский» аналог. Учитывая это обстоятельство, и следуя все тому же принципу согласованности интерфейса, используйте трехмерные эффекты только для изображения интерактивных элементов. При этом вводите лишь те детали, которые действительно необходимы для идентификации образа пользователем.

5.7.5. ПРОСТРАНСТВЕННОЕ РАЗМЕЩЕНИЕ ВИЗУАЛЬНЫХ ЭЛЕМЕНТОВ

Размер и взаимное расположение визуальных элементов очень важны для создания визуально последовательной и предсказуемой среды. Визуальная структура важна также с точки зрения передачи назначения элементов, отображенных в окне. В

общем случае при выборе варианта размещения элементов следует придерживаться тех же правил, которые используются при компоновке печатной страницы.

Группирование

Группирование предполагает компактное размещение взаимосвязанных элементов. Для реализации группирования может использоваться либо специальный элемент — группирующий блок, либо просто размещение элементов на соответствующем расстоянии друг от друга.

С целью обеспечения единого подхода к пространственному размещению графических элементов введена специальная единица измерения — *dialog base unit* (единица площади диалоговой панели), которую мы в дальнейшем для краткости будем называть «дискретой окна».

Дискрета окна — это аппаратно-независимая величина; в горизонтальном направлении она равна одной четверти средней ширины символов текущего системного шрифта; по вертикали — одной восьмой средней высоты символов текущего системного шрифта.

Рекомендуется оставлять между элементами в группе промежутки, равный по крайней мере четырем дискретам, а расстояние до края окна и между группами должно быть не менее семи дискрет (рис. 5.47). Хотя вы можете использовать цвет, чтобы визуально сгруппировать объекты, такой подход нельзя признать удачным, поскольку он может привести к нежелательным эффектам, когда пользователь изменит цветовую схему.

Элементы управления в панели инструментов старайтесь располагать так, чтобы от края панели до края окна оставался промежуток, равный по меньшей мере ширине рамки окна. В свою очередь, между элементами самой панели должно быть не менее четырех дискрет высоты (если вы не хотите сгруппировать набор связанных кнопок). В некоторых случаях, например, когда кнопки панели инструментов используются подобно набору переключателей, они могут располагаться слитно (без промежутка).

Основные кнопки управления вторичного окна целесообразно сгруппировать в верхнем правом углу окна или расположить в виде линейки вдоль нижнего края окна, как показано на рис. 5.48. Если в окне имеется предопределенная кнопка, то она, как правило должна стоять первой. Кнопки *ОК* и *Отменить* должны располагаться рядом. «Закрывать» группу должна кнопка *Справка* (если она поддерживается приложением). Если кнопка *ОК* не используется в данном окне, но имеются другие кнопки управления, то лучше всего установить кнопку *Отменить* в конце набора кнопок управления, но перед кнопкой *Справка*. Если же какая-либо кнопка применяется только для конкретной области окна, включите ее в эту область.

Рис. 5.47. Рекомендуемый формат размещения элементов управления во вторичном окне

Рис. 5.48. Пример размещения кнопок

Если кнопки (или другие элементы управления) вынесены на вкладку, предполагается, что их область действия распространяется только на эту вкладку; соответственно, кнопки, не входящие в состав ни одной из вкладок окна, относятся к окну в целом.

Выравнивание

Выравнивание представляет собой еще один дополнительный способ визуального отображения взаимосвязанной информации (или элементов управления). Как правило, различают три способа выравнивания информации:

- вертикальное (по левому или правому краю выравниваемых элементов);
- горизонтальное (по верхней строке или по верхнему краю элемента);
- смежное выравнивание (когда элементы смыкаются краями, рис. 5.49).

Рис. 5.49. Смежное выравнивание в группах

Если информация расположена вертикально, целесообразно выровнять ее элементы по левому краю соответствующей области. Это обычно облегчает пользователю быстрый просмотр информации. Вместе с тем, если в виде столбца выводятся числовые данные, значения которых могут изменяться, их лучше выровнять по правому краю (рис. 5.50).

Рис. 5.50. Вертикальное выравнивание

5.7.6. ВИЗУАЛИЗАЦИЯ ВЫПОЛНЯЕМЫХ ОПЕРАЦИЙ

Визуализация выполняемых операций является одним из способов предоставления пользователю обратной связи с приложением. Продуманный способ визуализации не только способствует лучшему уяснению пользователем сущности выполняемой операции, но и обеспечивает своевременную и корректную реакцию пользователя в случае ошибочных или неудачных действий. Это особенно важно для приложений, используемых в системах управления, принятия решений и других, работающих в режиме реального времени.

Визуализация операций выбора

Визуальная обратная связь при выполнении операций выбора должна позволить пользователю однозначно идентифицировать выбранный объект на фоне остальных. Способ отображения выбора обычно зависит от сущности объекта и текущей ситуации.

Изображение выбираемого объекта должно изменяться непосредственно в процессе выполнения операции выбора. То же самое относится и к выбираемой области. При этом состояние выбора должно отображаться только для активной области или уровня иерархии (например, для активного окна или подокна). Это поможет пользователю определить, к какому из имеющихся выборов относится выполняемое действие.

Тем не менее, в некоторых ситуациях может оказаться полезным одновременное визуальное выделение нескольких объектов, относящихся к разным компонентам приложения. Например, когда пользователь выбирает объект в первичном окне и затем выбирает пункт меню, относящегося к этому объекту, выбор должен отображаться как для объекта, так и для пункта меню. В некоторых случаях

требуется указать, что один из существующих выборов является «первичным», а другой (или другие) — «вторичным». В таких ситуациях целесообразно использовать цвет: область вокруг «вторичного» выбора может быть окрашена цветом, отличающимся от цвета выделения активного выбора.

Для многих типов объектов для указания состояния выбора может применяться системный цвет выделения.

Во вторичных окнах выделение текущего выбора цветом применяется, как правило, только для текстовых полей (в том числе входящих в состав комбинированных списков) и элементов списков. Для указания выбора других элементов управления целесообразно использовать визуальное отображение фокуса ввода.

В качестве средства выделения выбранного графического объекта могут использоваться зоны управления, рассмотренные в разделе «Взаимодействие пользователя с приложением». В этом случае для их изображения должен использоваться системный цвет выделения.

Визуализация операций пересылки

При выполнении операций пересылки средством визуализации является перемещение образа пересылаемого объекта (одновременное с перемещением указателя). При этом перемещаемое изображение объекта должно визуально отличаться от его изображения в исходной позиции: в качестве него используется либо полупрозрачное, либо контурное изображение объекта (рис. 5.51). Для создания полупрозрачного изображения рекомендуется использовать пиксели с коэффициентом прозрачности, равным 0.5.

Рис. 5.51. Изображение пересылаемого объекта при выполнении операций пересылки

Представление пересылаемого объекта всегда определяется свойствами объекта (позиции-) приемника. Другими словами, когда перемещаемое изображение оказывается над объектом-приемником, оно должно отражать результат пересылки. Например, если пересылаемый объект будет вставлен в объект-приемник в качестве пиктограммы, то изображение должно принять соответствующий вид.

Анимация

Анимация во многих случаях может оказаться весьма эффективным средством передачи визуальной информации (например, в качестве иллюстрации функционирования какого-либо устройства или выполнения операции, рис. 5.52). Иногда ее применение может просто оживить интерфейс приложения и сделать общение пользователя с ним более приятным.

Рис. 5.52. Пример использования анимации

Эффективное использование анимации предполагает учет многих факторов из тех, что были рассмотрены выше применительно к разработке других графических элементов интерфейса.

Одно из важнейших требований к использованию анимации заключается в том, что она не должна влиять на интерактивность интерфейса. Не заставляйте пользователя

ожидать завершения «мультика». Если анимация не является частью выполняемого процесса, разрешите пользователю либо прервать ее, либо параллельно продолжать работу.

Избегайте необоснованного использования анимации. Если анимация используется только для создания декоративного эффекта, она может отвлекать или даже раздражать пользователя (вспомните, например, телевизионные заставки перед рекламными блоками).

5.8. ТРИ СЛУЧАЯ ИЗ ЖИЗНИ GUI

Один из классиков научного мировоззрения в свое время сказал примерно следующее: «Ни одну теорию нельзя доказать с помощью примеров, но любую теорию с помощью примеров можно опровергнуть». Тем не менее мы решили прервать ненадолго изложение Windows-стандартов на построение GUI-приложений и продемонстрировать читателю, к чему может привести пренебрежительное отношение к детальному проектированию и реализации пользовательского интерфейса. Те примеры, о которых пойдет речь ниже, относятся к свободно распространяемому в Интернете программному обеспечению и взяты практически наугад; то есть авторы совершенно не знакомы с разработчиками и никоим образом не хотели их обидеть. Более того, по своему функциональному предназначению эти утилиты представляются нам весьма полезными и работают достаточно корректно.

Итак, *случай первый* (самый тяжелый). На рис. 5.53 показано окно утилиты TimeBack, обеспечивающей управление таймером компьютера. Плохая различимость надписей в окне объясняется не низким полиграфическим качеством рисунка, а цветовой гаммой, выбранной разработчиком (черные символы на темно-зеленом фоне). Вместе с тем, взгляд потенциального пользователя сразу привлекает надпись в центре окна «Запустить приложение», на которой непроизвольно хочется Щелкнуть мышью. На самом деле это статический текст, который визуальнo не связан ни с одним из элементов управления, находящихся в окне. Впрочем, как и надпись «Таймер», расположенная ниже. Вообще, ко всем надписям, имеющимся в окне, хотелось бы поставить дополнительный вопрос: «Дата» — какая?; «Продолжительность» — чего?; «Авто Заккрытие» — чего? (в последнем случае, кроме того, налицо также и отклонение от норм русского языка).

Рис. 5.53. Окно утилиты Time Back

Для установки даты используется выпадающий список; хотя этот элемент управления реализован весьма оригинально и интересно, при работе с ним возникают определенные проблемы. Во-первых, при попытке закрыть список без установки новой даты появляется сообщение об ошибке (рис. 5.54), текст которого вызывает недоумение и не может ничем помочь пользователю (оказывается, обязательно следует установить какую-либо новую дату).

Во-вторых, при установке даты соответствующий элемент отображается как выбранный только после щелчка ЛКМ; однако при этом список тут же закрывается, так что если пользователь случайно щелкнет не там, где хотел, ему снова придется открывать список и повторять процедуру.

Для установки продолжительности работы утилиты используются два элемента управления: дискретная текстовая область и выпадающий комбинированный

список. Продолжительность может измеряться в минутах или в секундах; диапазон изменения для обеих единиц одинаков — от нуля до 3001 (как вы думаете, сколько это в часах?); при попытке ввести вручную любое дробное значение оно автоматически преобразуется в нуль. Использование комбинированного выпадающего списка в первый момент создает иллюзию того, что пользователь может задать собственные единицы измерения продолжительности, однако это не так: его «свобода выбора» ограничивается возможностью «подредактировать» название пункта списка в текстовом поле, что, собственно, ни на что не влияет.

Вместе с тем, необходимо отметить, что операции работы с окном реализованы в рассматриваемой утилите корректно: состав доступных команд всплывающего меню окна согласован с кнопками управления окном. Этого как раз нельзя сказать о следующей утилите, которую мы, кстати, неоднократно успешно использовали (вот она, людская благодарность!).

Случай второй. Утилита называется «Конвертор Лексикон—»RTF» и обеспечивает преобразование текста (вполне корректное) в любой из указанных форматов (рис. 5.55).

Рис. 5.54. Реакция утилиты на работу с календарем

Рис. 5.55. Окно утилиты «Конвертор Лексикон—»RTF»

Однако состав команд всплывающего меню окна утилиты не согласован с кнопками управления окном. Судя по тому, что кнопка *Восстановить/Развернуть* отображается как недоступная, разработчики хотели запретить пользователям изменение размеров окна. Вместе с тем, во всплывающем меню окна остались доступны все команды изменения размера окна, в результате чего при желании исходное окно приложения можно изменить произвольным образом (рис. 5.56).

Рис. 5.56. «Запрещенное» разработчиками изменение размеров окна

Кроме того, в окне приложения присутствуют элементы управления (флажки «Включая подкаталоги» и «Все файлы в один каталог»), взаимосвязь между которыми можно обнаружить только при счастливом стечении обстоятельств. Попытки получить пояснения по их использованию с помощью кнопки *Справка* приводят к появлению на экране сообщения «Не удастся найти файл справки».

Случай третий. Свидетельствующий о том, что пробелами в организации интерфейса грешат не только «русскоязычные» разработчики. Утилита «Message Manager» предназначена для создания окон сообщений стандартных типов (подробному обсуждению самих окон посвящен один из следующих разделов книги). Пользовательский интерфейс утилиты реализован на основе первичного окна (рис. 5.57), которое стараниями разработчиков превратилось в «почти-вторичное» окно: кнопка входа для него на Панели задач появляется почему-то только после создания первого сообщения, но и в этом случае свернуть окно невозможно, и оно постоянно присутствует на экране.

Правда, необходимо отдать должное последовательности разработчиков данной, утилиты: состав доступных команд во всплывающем меню окна совпадает с доступными кнопками управления окном. Но вернемся к пробелам интерфейса. Разработчиками не была создана пиктограмма приложения и поэтому она,

естественно, отсутствует в полосе заголовка окна; тем не менее, двойной щелчок ЛКМ в той позиции, где должна находиться пиктограмма, приводит к закрытию окна без каких бы то ни было комментариев. Не менее впечатляющий эффект дает и нажатие кнопки *ОК*, расположенной в правой верхней части окна. Независимо от того, какие манипуляции перед этим выполнял пользователь, при нажатии этой кнопки утилита безмолвно исчезает, оставляя вместо себя на экране созданное вами окно сообщения, которое вы, может быть, и не хотели бы больше видеть.

Рис. 5.57. Окно утилиты «Message Manager»

Завершая это мини-обозрение, хотелось бы еще раз отметить, что приведенные здесь в качестве примеров утилиты далеко не худшее из того, что «гуляет» по Интернету, все они обеспечивают решение тех задач, для которых создавались и, по большому счету, указанные недостатки не имеют принципиального значения. Казалось бы... Но представьте себе, что эти три утилиты являются компонентами одного более сложного приложения. .. Представили? В этом случае недостатки, свойственные каждой из них, будут не просто суммироваться, а перемножаться с недостатками двух других.

Вероятно, в какой-то степени издержки интерфейса могут быть компенсированы средствами помощи пользователю, но и в этом отношении рассмотренные изделия не могут служить образцом; правда, в первом и третьем примерах используется всплывающая подсказка, но, как мы видели, этого оказалось недостаточно.

6. ПРОЕКТИРОВАНИЕ СРЕДСТВ ПОДДЕРЖКИ ПОЛЬЗОВАТЕЛЯ

Средства оперативной поддержки пользователя являются важной частью приложения и могут быть реализованы различными способами, от использования команд явного вызова помощи до автоматического отображения справочной информации, соответствующей текущей ситуации. Содержание выводимых сообщений также может носить разнообразный характер и представлять собой либо краткое пояснение, либо ссылку на другой источник информации, либо своеобразный электронный учебник. Но помощь пользователю должна быть всегда простой, эффективной и своевременной, чтобы пользователь мог получить ее до того, как будет вынужден прекратить работу.

6.1. ОКНО СООБЩЕНИЕ

Окно *СООБЩЕНИЕ* (*Message Box*) — это вторичное окно, используемое для вывода на экран сообщений пользователю; как правило, сообщения содержат информация о конкретной ситуации или условиях выполнения операций. Сообщения являются важной частью пользовательского интерфейса любого программного продукта. В связи с этим разработке сообщений должно быть уделено самое пристальное внимание. Вообще же лучше избегать ситуаций, которые требуют вывода сообщений. Например, если может возникнуть ситуация, связанная с нехваткой свободного дискового пространства для продолжения работы, следует ее предотвратить, вместо того, чтобы потом сообщать о ней пользователю.

Заголовок окна **СООБЩЕНИЕ**

Заголовок окна должен идентифицировать объект или процесс, с которым связано сообщение; поэтому в нем обычно фигурирует имя объекта. Например, если сообщение обусловлено редактированием документа, то название окна должно содержать имя этого документа, а также имя используемого для работы с ним приложения. Если объект, вызвавший появление сообщения, не является документом, то используйте в заголовке окна только имя приложения. Обеспечение корректной идентификации сообщения особенно важно в мультизадачной среде, поскольку в этом случае сообщения не всегда являются результатом действий пользователя. Кроме того, поскольку технология OLE допускает внедрение объектов, то выбор объекта пользователем может привести к активизации другого, связанного с ним, приложения (*сервера*). Следовательно, заголовок окна в этом случае играет важную роль как средство коммуникации с источником сообщения.




Не рекомендуется использовать в заголовке окна сообщения такие слова, как например, «предупреждение» или «предостережение», поскольку символ сообщения сам по себе уже отражает цель сообщения. И никогда не используйте в заголовке окна слово «ошибка».

Форматы окна **СООБЩЕНИЕ**

Как правило, окно **СООБЩЕНИЕ** содержит графический символ, который указывает на тип выводимого сообщения. Большинство используемых на практике сообщений могут быть отнесены к одной из следующих категорий (табл. 6.1).

Таблица 6.1

Графические обозначения типов сообщений

Символ	Тип сообщения	Описание
	Информация	Предоставляет пользователю информацию о результатах выполнения команды. Пользователю не предлагается возможность выбора; считается,
	Предупреждение	Предупреждает пользователя о возникновении ситуации, которая требует от него выбора одного из возможных вариантов последующих действий приложения или системы; используется в тех случаях, когда предстоящая операция является
	Критическая ситуация	Информирует пользователя о серьезной проблеме, которая требует его вмешательства или внесения каких-либо изменений

На рис. 6.1 показаны варианты использования различных типов сообщений.

Поскольку вывод на экран окна сообщения приостанавливает выполнение текущего задания пользователя, следует отображать его только в тех случаях,

когда окно соответствующего приложения является активным. Если оно не активно, то приложение должно использовать свою кнопку входа на Панели задач, чтобы предупредить пользователя о поступившем сообщении (например, посредством цветовой индикации, как показано на рис. 6.3). Само же окно сообщения отображается только тогда, когда пользователь активизирует приложение.

Если для индикации наличия сообщения применяется подсветка кнопки входа, то лучше не подсвечивать кнопку непрерывно, а использовать для привлечения внимания пользователя мерцание (например, трехкратное), и только после этого оставить кнопку в выделенном состоянии.

Рис. 6.1. Примеры использования различных типов сообщений

Не рекомендуется снабжать окно сообщения символом в виде знака вопроса, использовавшимся в ранних версиях ОС Windows, поскольку предполагается, что пользователи могут воспринять такое сообщение как справочную информацию, выдаваемую при запросе помощи. Тем не менее, практика показывает, что такая ассоциация встречается все реже и реже, и во многих приложениях используются окна сообщений, содержащие знак вопроса; как правило, такие сообщения применяются в тех случаях, когда пользователь должен уточнить тот или иной момент, влияющий на дальнейшее выполнение задания (рис. 6.2).

Рис. 6.2. Использование окна *СООБЩЕНИЕ* для подтверждения намерений пользователя

Вы можете включить в окно *СООБЩЕНИЕ* *собственные* графические символы или анимацию, но рекомендуется не злоупотреблять этой возможностью. Кроме того, не используйте собственные графические символы для идентификации приведенных выше стандартных типов сообщений.

Рис. 6.3. Индикация наличия сообщения в неактивном приложении

Указанный способ уведомления предпочтителен в том случае, если сообщение не связано с сохранностью данных пользователя; в этом случае ваше приложение может немедленно вывести на экран системно-зависимое окно сообщения.

Для каждой конкретной ситуации должно формироваться только одно окно *СООБЩЕНИЕ*, так как большое число сообщений может запутать пользователя.

Окно *СООБЩЕНИЕ* *может* быть реализовано таким образом, чтобы после предоставления пользователю необходимой информации оно закрывалось автоматически, не требуя каких-либо действий пользователя. Например, окно, обеспечивающее визуальное представление состояния некоторого процесса, автоматически исчезает при его завершении. Аналогично, стартовое окно программного продукта («заставка»), которое идентифицирует имя продукта и содержит информацию об авторском праве (рис. 6.4), может быть автоматически Удалено с экрана после того, как приложение загрузилось; в этом случае окно *СООБЩЕНИЕ* не должно содержать символ типа.

Однако такой подход может быть использован только для информационных сообщений, но ни в коем случае не для предупреждающих сообщений или сообщений о критических ситуациях, поскольку некоторые пользователи могут не успеть прочитать и уяснить текст сообщения.

Рис. 6.4. Применение окна *СООБЩЕНИЕ* в качестве «заставки» при запуске приложения

Использование кнопок в окне *СООБЩЕНИЕ*

Кнопки обеспечивают простой и эффективный способ взаимодействия пользователя с окном *СООБЩЕНИЕ*. В большинстве случаев оно содержит только такие кнопки, которые обеспечивают выбор пользователем одного из возможных вариантов ответа (или действий). При этом в качестве предопределенной кнопки целесообразно использовать такую, которая представляет наиболее вероятный или наименее опасный вариант.

Если сообщение не требует от пользователя ввода никакой информации, окно должно содержать только кнопку *ОК* и, возможно, кнопку *Справка*. Если же реакция на сообщение предполагает выбор пользователем одного из вариантов, для каждого из них в окне должна иметься соответствующая кнопка. При этом лучше всего постараться сформулировать вопрос таким образом, чтобы пользователь мог ответить на него «Да» или «Нет». Если это сложно, следует использовать в качестве меток кнопок наименование связанных с ними действий, например, *Сохранить* и *Удалить*. В окне сообщения могут также использоваться кнопки, управляющие состоянием окна. Например, если сообщение говорит о том, что пользователь должен переключиться на другое окно приложения, чтобы скорректировать выполняемую операцию, окно сообщения может содержать кнопку, которая переключает пользователя на это окно.

Некоторые ситуации могут потребовать от пользователя не только сделать выбор между выполнением или невыполнением действия, но и вообще отменить процесс, вызвавший появление сообщения. Для таких случаев в окне сообщения необходимо предусмотреть кнопку *Отменить*, как показано на рис. 6.5. Следует иметь в виду, что применение такой кнопки требует от разработчика особой аккуратности.

Если в окне Сообщение используется кнопка *Отменить*, напомните пользователю, что отмена подразумевает восстановление того состояния процесса или задания, которое имело место до выдачи сообщения. Если же кнопка *Отменить* используется для того, чтобы прервать процесс, и его начальное состояние при этом не восстанавливается, лучше заменить ее кнопкой *Останов*.

Если есть необходимость включить в окно сообщения другие элементы интерфейса, всегда следует учитывать потенциальное увеличение сложности работы с окном.

Рис. 6.5. Пример использования кнопки *Отменить* в окне Сообщение

Выбор текста сообщений

Текст сообщения должен быть ясным, кратким, и использовать терминологию, понятную пользователю. Не рекомендуется применять в сообщениях технический жаргон или системную информацию (особенно в приложениях, ориентированных на непрограммирующих пользователей).

Рекомендации по составлению сообщений можно сформулировать в виде следующих руководящих принципов:

- В тексте сообщения должны содержаться: краткая формулировка проблемной ситуации, ее вероятная причина (если она известна), и рекомендации относительно возможных действий пользователя (даже если решение представляется разработчику очевидным). Например, вместо сообщения «Недостаточно места на диске» следует использовать, например, такое: «Невозможно сохранить файл, поскольку диск заполнен. Попробуйте сохранить его на другом диске или освободите место на этом же диске».

- Если у пользователя имеется несколько вариантов решения проблемы, изложите сообщение так, чтобы максимально облегчить пользователю выбор. Например, вместо сообщения «Одна или более строк превышают допустимую длину (60 символов)» лучше использовать такое: «Одна или более строк превышают допустимую длину. Строка может содержать максимум 60 символов при вертикальной ориентации бумаги или 90 символов при горизонтальной. Вы хотите переключиться на горизонтальную ориентацию сейчас?» В качестве вариантов ответа предложите пользователю «Да» и «Нет».

- Избегайте использования необязательных технических терминов и чрезмерно сложных предложений. В первую очередь это относится к использованию в сообщениях англоязычных терминов в русской транскрипции, таких как «спиннер», «баннер» и т.д.

- Текст сообщения не должен содержать обвинений в адрес пользователя или носить нравоучительный характер. Например, вместо «Ошибка при вводе имени файла» лучше написать «Не могу найти файл с таким именем, проверьте правильность ввода». Вообще крайне нежелательно появление в сообщении слова «ошибка»

- Сообщение должно быть как можно более конкретным. Не используйте одно и то же окно сообщения для описания нескольких проблемных ситуаций одновременно. Например, если существуют различные причины, по которым файл не может быть открыт, сформируйте отдельное сообщение для каждого случая.

- Старайтесь избегать использования стандартных системных сообщений типа «Ошибка ядра ОС INT 244», заменяя их по возможности собственными сообщениями, ориентированными на пользователя.

- Текст сообщения должен быть хотя и достаточно полным, но в то же время лаконичным. Практика показывает, что хорошо продуманное сообщение уместается в двух — трех строках. Если необходимо более подробное пояснение, включите в окно сообщения кнопку *Справка*.

При необходимости каждое сообщение может быть снабжено идентификационным номером, однако чтобы он не мешал пользователям воспринимать текст сообщения, помещайте его в конце текста сообщения; не рекомендуется также включать номер сообщения в заголовок окна.

6.2. КОНТЕКСТНАЯ ПОМОЩЬ

Контекстная помощь обеспечивает предоставление пользователю информации о конкретном объекте или ситуации. Она реализуется в виде ответов на вопросы типа «Что это?» и «Как я должен использовать это?» В данном разделе представлены

некоторые основные способы использования контекстной помощи пользователю в создаваемом приложении.

6.2.1. КОМАНДА ЧТО ЭТО?

Команда *Что это?* (What's This?) обеспечивает пользователя контекстной информацией относительно любого объекта, представленного на экране, включая элементы управления на панелях свойств и других диалоговых панелях. Эта форма контекстной помощи пользователю называется **контекстно-зависимой подсказкой**. Как показано на рис. 6.6, доступ пользователя к этой команде может быть реализован одним из следующих способов:

- Через выпадающее меню *Справка* первичного окна.
- С помощью кнопки на панели инструментов.
- С помощью кнопки, расположенной в полосе заголовка вторичного окна.
- Через всплывающее меню конкретного объекта.

Рис. 6.6. Различные методы доступа к команде *Что это?*

Когда пользователь выбирает команду *Что это?* из меню *Справка* или нажимает соответствующую кнопку на панели инструментов, система переходит во временный режим (**режим подсказки**). Визуальным признаком этого режима служит изменение формы указателя (рис.6.7). Альтернативным средством перехода в режим подсказки является комбинация клавиш <Shift>+F1.

Рис. 6.7. Вид указателя в режиме контекстно-зависимой подсказки

Указатель должен принимать соответствующий вид только над тем окном, в котором пользователь хочет получить контекстно-зависимую подсказку, то есть над тем активным окном, которое находится в режиме подсказки

Если в этом режиме пользователь щелкает ЛКМ на интересующем его объекте, отображается всплывающее окно контекстно-зависимой подсказки для данного объекта. Выводимая подсказка должна обеспечивать краткое пояснение относительно назначения объекта и способов его использования (рис. 6.8). Как только окно подсказки появится на экране, указатель должен быть восстановлен в его обычном состоянии.

Рис. 6.8. Всплывающее окно контекстно-зависимой подсказки

Если пользователь нажимает какую-либо клавишу-акселератор, которая относится к окну, находящемуся в режиме подсказки, вы можете отобразить окно подсказки для команды, связанной с этой клавишей.

Описанный выше механизм носит достаточно общий характер, поэтому при его реализации возможны некоторые исключения.

Во-первых, если пользователь в режиме подсказки выбирает заголовок меню, расположенный в полосе меню, или пункт каскадного меню, сохраните для них основной режим до тех пор, пока пользователь не выберет один из пунктов меню, и только после этого отобразите контекстно-зависимую подсказку для этого пункта.

Во-вторых, если пользователь щелкает на каком-либо объекте ПКМ и этот объект поддерживает всплывающее меню, сохраните режим подсказки, пока пользователь не выберет один из пунктов меню или не закроет его. Если объект не поддерживает всплывающее меню, то эффект должен быть таким же, как при использовании ЛКМ.

Наконец, если выбранный объект или позиция не поддерживает контекстно-зависимую подсказку или вывод подсказки неуместен, следует запретить переход в режим подсказки.

Если пользователь повторно выбирает команду *Что это?*, в том числе используя комбинацию клавиш <Shift>+F1, или нажимает клавишу <Esc>, режим подсказки отменяется. При этом указатель восстанавливается в его обычном состоянии.

Когда пользователь выбирает команду *Что это ?* из всплывающего меню объекта, порядок взаимодействия немного отличается от описанного выше. Поскольку пользователь в этом случае идентифицирует объект, нажимая ПКМ, нет необходимости устанавливать режим подсказки. Вместо этого можно сразу отобразить всплывающее окно контекстно-зависимой подсказки для выбранного объекта.

В этом случае в качестве быстрой клавиши используется клавиша F1; это означает, что при нажатии F1 отображается окно контекстно-зависимой подсказки для того объекта, на котором установлен фокус ввода.

Правила создания контекстно-зависимой подсказки

Контекстно-зависимая подсказка может быть сформулирована в виде ответа на вопросы «Что это?», «Почему?» или «Каким образом?», но если процедура требует выполнения нескольких шагов, продумайте возможность предоставления пользователю соответствующей информации в форме помощи, **ориентированной на задание** (подробнее этот подход будет рассмотрен ниже). Содержание подсказки должно быть кратким и вместе с тем достаточно полным, чтобы пользователь мог быстро прочитать ее и уяснить смысл.

Как один из допустимых вариантов, может быть реализована контекстно-зависимая справка для типов файлов, поддерживаемых вашим приложением. Это позволит пользователю, выбрав команду *Что это?* из всплывающего меню пиктограммы файла, получить о нем необходимую информацию (рис. 6.9).

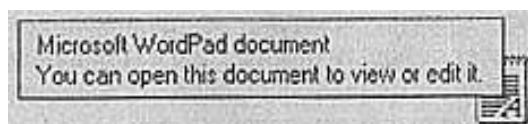


Рис. 6.9. Контекстно-зависимая справка для пиктограммы файла

6.2.2. ВСПЛЫВАЮЩАЯ ПОДСКАЗКА

Другой формой контекстной помощи пользователю является всплывающая подсказка (*tooltips*). **Всплывающая подсказка** — это небольшое всплывающее окно, которое содержит название элемента управления, не имеющего текстовой метки. Наиболее распространенный вариант использования такой подсказки — пояснения для кнопок панели инструментов, которые имеют только графическое обозначение (рис. 6.10).

Рис. 6.10. Всплывающая подсказка для кнопки панели инструментов

Всплывающая подсказка появляется возле указателя, если он находится над кнопкой в течение некоторого (достаточно короткого) интервала времени и остается на экране, пока пользователь не нажмет кнопку мыши или не переместит указатель, либо в течение установленного периода времени. Если пользователь перемещает указатель на другой элемент управления, поддерживающий всплывающую подсказку, задержка игнорируется, и новая подсказка отображается немедленно, заменяя предыдущую.

Для стандартных элементов управления система автоматически обеспечивает всплывающую подсказку. Если вы создаете собственные окна всплывающей подсказки, обеспечьте их согласованность с существующими системными окнами подсказки.

6.2.3. ВЫВОД СООБЩЕНИЙ В СТРОКЕ СОСТОЯНИЯ

Контекстная помощь пользователю может также быть реализована на основе строки состояния. Однако, если пользователю предоставлено право выбора, отображать или нет строку состояния, то лучше ее не использовать в этих целях (если альтернативные средства доступа к выводимой в ней информации отсутствуют). Кроме того, поскольку строка состояния не всегда находится в центре внимания пользователя, то он может не обратить внимание на появившееся там сообщение. Поэтому следует рассматривать вывод сообщений в строке состояния лишь как дополнительную форму помощи пользователю.

Как уже было сказано, в строке состояния наряду с информацией о текущей ситуации могут отображаться краткие сведения о выбираемых пунктах меню и кнопках панели инструментов, как показано на рис. 6.11. Как и при использовании всплывающей подсказки, соответствующее окно должно быть активным. Описание интересующего элемента отображается в строке состояния, когда пользователь перемещает курсор на кнопку панели инструментов или на один из пунктов меню.

Рис. 6.11. Вывод справки о пункте меню в строке состояния

В качестве сообщения, отображаемого в строке состояния, может также использоваться индикатор процесса или какие-либо другие средства информирования пользователя о выполняемых операциях, таких, например, как печать или сохранение файла. В некоторых случаях строка состояния может использоваться для отражения состояния фоновых процессов, чтобы окно основного процесса не перекрывалось окном сообщения.

При составлении текста сообщения в строке состояния следует придерживаться тех же правил, которые были рассмотрены ранее применительно к другим текстовым сообщениям. Описывая команду со специфической функцией, используйте ключевые слова, отражающие эту специфику. Если диапазон применения команды весьма широк, попытайтесь тем не менее дать ее обобщенную характеристику. Например, назначение команды *Статистика* можно пояснить следующим образом: «Сбор и отображение статистических сведений об активном документе».

Определяя сообщения для пунктов меню и кнопок панели инструментов, необходимо учитывать, что в некоторых ситуациях они могут быть недоступны. Если имеет место одна из таких ситуаций, предоставьте пользователю соответствующее пояснение. Например, когда пользователь выбирает команду *Вырезать*, которая в данный момент является недоступной, может быть выведено сообщение: «Команда недоступна, поскольку объект не выбран».

6.2.4. КНОПКА СПРАВКА

Выдача контекстной справочной информации можете также быть реализована для таких элементов интерфейса, как панель свойств, диалоговая панель или окно сообщения. С этой целью в них должна быть включена кнопка *Справка*, как показано на рис. 6.12. При «нажатии» пользователем этой кнопки должно открываться не всплывающее окно контекстно-зависимой подсказки, а вторичное окно, содержащее соответствующую справочную информацию.

Рис. 6.12. Использование кнопки *Справка* во вторичном окне

Помощь, предоставляемая пользователю по этой команде, отличается по форме от подсказки, выводимой по команде *Что это?* Она должна обеспечить пользователя более полной, развернутой пояснительной информацией по данному окну. Например, для окна Сообщение это может быть более подробная информации о причинах и способах устранения возникшей ситуации.

С развитием средств оперативной помощи пользователям через Интернет многие приложения обеспечивают доступ к ним также посредством кнопки *Справка*.

Вместе с тем, следует иметь в виду, что использование команды *Справка* является дополнительной, вспомогательной формой контекстной помощи пользователю, которая не должна подменять контекстно-зависимую подсказку, вызываемую по команде *Что это?* Нельзя также рассчитывать на то, что наличие подробной справочной информации может компенсировать недостатки неудачно спроектированного интерфейса вторичных окон.

6.3. ПРОБЛЕМНО-ОРИЕНТИРОВАННАЯ ПОМОЩЬ

Проблемно-ориентированная *помощь (Task-Oriented Help)* представляет собой описание последовательности шагов, необходимых для выполнения некоторого задания пользователя. Для предоставления пользователю проблемно-ориентированной помощи соответствующая справочная информация организуется в виде разделов, каждый из которых описывает отдельный шаг задания. В свою очередь, каждый такой раздел отображается на экране в виде отдельного окна, называемого окном *Раздел задания*.

Окно Раздел задания (Task Topic Window) — это первичное окно, для которого поддерживаются в основном те же правила взаимодействия, что и для любого другого первичного окна. В частности, пользователь может изменять размер этого окна, сворачивать его и т.п.

Классификация окна Раздел задания как первичного окна обусловлена спецификой его отображения и использования. Вместе с тем, в некоторых вариантах технической документации окно этого типа названо *вторичным окном помощи*.

Первоначальный доступ к любому из окон разделов задания пользователь получает через *браузер разделов*, описанный ниже в этой главе. Вместе с тем, доступ к некоторым разделам может быть реализован посредством *навигационных ссылок*, установленных в других окнах разделов задания.

Окно Раздел задания содержит в верхней части набор кнопок, которые обеспечивают пользователю доступ к браузеру разделов, к предыдущей выбранной теме, а также к

другим командам (например, копирования и вывода на печать содержимого окна), как показано на рис. 6.13. Разработчик приложения может сам определить, какие кнопки используются в окне Раздел задания.

Рис. 6.13. Окно Раздел задания

Размер и расположение окон Раздел задания зависит, как правило, от специфики создаваемого приложения, однако целесообразно выбирать их таким образом, чтобы окно занимало минимум пространства основного окна приложения, что позволит пользователю прочесть выведенную информацию, не перемещая окно. Вместе с тем, рекомендуется использовать для основных разделов такой размер окна, чтобы для просмотра информации не требовались полосы прокрутки. Это облегчит работу начинающему пользователю, который может быть незнаком с прокруткой.

Заголовок окна Раздел задания должен однозначно идентифицировать отображаемую в нем информацию. В качестве же названия темы, используемого непосредственно в окне, уместно взять ключевую фразу раздела. Кроме того, следует согласовать название темы с данными, включаемыми в браузер разделов, который предоставляет прямой доступ к разделу. Это не означает, что название темы должно дословно совпадать с обозначением раздела, используемого в браузере, но они должны быть достаточно близки, чтобы пользователь мог легко соотнести их друг с другом.

Подобно окну всплывающей подсказки, в качестве установленного по умолчанию цвета рабочей области окна Раздел задания должен использоваться системный Цвет, определенный для окон помощи. Это обеспечивает быстрое «узнавание» пользователем такого окна на фоне других окон. Тем не менее, для некоторых специфических разделов может быть установлен собственный цвет окна.

Использование кнопок в окне Раздел задания

Кнопки, отображаемые в верхней части окна Раздела задания, определяются **Help-файлом** вашего приложения. Как минимум, должны присутствовать следующие кнопки:

- кнопка, которая обеспечивает доступ к диалоговой панели браузера разделов;
- кнопка, позволяющая пользователю вернуться к предшествующей теме;
- кнопки, предоставляющие доступ к другим функциям (например, копирования и печати).

Первая из перечисленных выше кнопок обычно называется *Содержание (Contents)*; она обеспечивает вывод на экран одноименной вкладки окна браузера разделов, представленного в том же виде, в котором окно было отображено последний раз. Поскольку стандартное окно браузера содержит три вкладки: *Содержание (Contents)*, *Предметный указатель (Index)* и *Поиск (Find)*, то при необходимости для двух последних в окне Раздел задания могут быть созданы собственные кнопки; в этом случае пользователь будет обеспечен прямым доступом к указанным вкладкам браузера разделов.

Как и контекстно-зависимая подсказка, справочная информация по разделам задания должна быть написана лаконично, но доходчиво. При этом она должна быть сформулирована в виде ответа на вопрос «Каким образом?», а не «Что?» или «Почему?», поскольку призвана помочь пользователю в выполнении конкретного задания, а не просто расширить его знания по соответствующей теме. Если существует несколько различных способов достижения цели, выберите один из них (обычно

самый простой либо наиболее общий). Если вы считаете необходимым включить информацию об альтернативных методах, предоставьте пользователю доступ к ней каким-либо другим образом.

Дополнительно окно Раздел задания может содержать кнопку *Связанные разделы*, обеспечивающую непосредственный доступ к другим темам. Когда пользователь выбирает эту кнопку, на экране должна появляться диалоговая панель *Найденные разделы* (как показано на рис. 6.14).

Рис.6.14. Диалоговая панель Найденные Разделы

Использование кнопок-акселераторов

Окно Раздел задания может также содержать **кнопку-акселератор** («*Выполнить это*»), которая предоставляет пользователю возможность быстрого автоматизированного выполнения конкретного шага задания, как показано на рис. 6.15. Например, такая кнопка может использоваться, чтобы автоматически открыть необходимую диалоговую панель, панель свойств или выполнить какую-либо операцию, если пользователю не удастся это сделать самостоятельно.

Рис. 6.15. Окно Раздел задания с кнопкой-акселератором

Применение кнопок-акселераторов не только повышает эффективность работы пользователя, но также уменьшает объем информации, которую он должен усвоить. Тем не менее, кнопки-акселераторы не должны использоваться в качестве единственного средства выполнения задания (или отдельного его шага), особенно если вы хотите научить пользователя выполнять задание самостоятельно. Для наиболее общих заданий может быть использован компромиссный вариант, при котором окна проблемно-ориентированной помощи и предоставляют пользователю информацию о том, как выполнить задание, и содержат кнопки-акселераторы, облегчающие его выполнение. Например, окно Раздел задания может содержать текст «Чтобы открыть панель свойств, нажмите эту кнопку», и кнопку-акселератор, открывающую указанную панель.

6.4. СПРАВОЧНИК

Справочник (*Reference Help*) обеспечивает предоставление пользователю справочной информации в форме интерактивной документации. Использование Справочника помогает пользователю уяснить общие основные характеристики программного продукта.

В качестве основы для создания Справочника используется первичное окно, отличающееся по структуре от рассмотренного выше окна Раздел задания (рис. 6.16).

Рис. 6.16. Окно Справочника

Доступ к Справочнику может быть реализован несколькими способами. Наиболее распространенный из них — явный вызов посредством соответствующей команды из выпадающего меню *Справка (Help)*, но возможны также использование специальной кнопки на панели инструментов или вызов через пиктограмму конкретного объекта.

Окно Справочника содержит полосу меню с разделами *Файл (File)*, *Правка (Edit)*, *Закладка (Bookmark)*, *Параметры (Options)* и *Помощь (Help)*, а также панель

инструментов с кнопками *Содержание (Contents)*, *Указатель (Index)*, *Назад (Back)* и *Печать (Print)*. Для окна справочника система поддерживает перечисленные элементы по умолчанию. Они обеспечивают выполнение пользователем таких операций, как открытие конкретного Help-файла (используя браузер разделов), копирование и печать содержимого страницы Справочника, создание комментария и закладки для данной страницы, а также настройку параметров окна. Разработчик может добавить и другие кнопки, отвечающие конкретному варианту реализации Справочника.

Хотя Справочник может содержать информацию подобную той, которая выводится в окнах контекстной подсказки и проблемно-ориентированной помощи, эти формы помощи не исключают друг друга. Зачастую именно разумное сочетание всех этих средств оказывается наиболее эффективным с точки зрения помощи пользователю.

Каждое окно (страница) Справочника может включать текст, графику, элементы анимации, видеоклипы и даже звуковое сопровождение. Кроме того, система поддерживает некоторые специальные средства, которые рассматриваются ниже.

Дополнительные меню и инструментальные кнопки

Разработчик может как создать в окне Справочника дополнительные меню и кнопки, так и удалить существующие разделы меню.

Поскольку Help-файлы обычно содержат связанные темы, рекомендуется включить в панель инструментов окна Справочника кнопки *Предшествующая тема* и *Следующая тема*. (обычно они обозначаются символами » и « соответственно).

Еще одна кнопка, которая может во многих случаях оказаться полезной — *Смотри также (SeeAlso)*, позволяющая вывести на экран диалоговую панель *Найденные разделы* (рис. 6.14), содержащую список связанных тем. Если Справочник имеет достаточно сложную структуру, на панель инструментов окна могут быть выведены также кнопки *Обзор (Up)* (для перехода на корневой или обзорный раздел Справочника) и *История (History)*, которая позволяет отобразить список тем, просмотренных пользователем.

Рекомендуется делать кнопки панели инструментов контекстно-зависимыми (т.е. заи: ящими от просматриваемой пользователем страницы Справочника). Например, если текущая страница является последней в цепи просмотра, следует сделать недоступной кнопку *Следующая тема*, для исходной, корневой страницы, следует запретить использование кнопки *Обзор*.

Выбор названия темы (страницы) Справочника

Каждая тема или страница Справочника, отображаемая в новом окне, должна иметь собственное название. Название идентифицирует тему и помогает пользователю ориентироваться в содержании Справочника. Кроме того, название должно соответствовать наименованию темы, используемому в окне браузера разделов, а также имени Help-файла.

Неперемещаемые области

Если какой-либо раздел (страница) Справочника не помещается по вертикали на один экран, может оказаться полезным использование **неперемещаемой области**. Такая область позволяет оставить видимым название темы (и другую необходимую информацию), когда пользователь перемещает в окне остальное содержимое

страницы. Визуальной границей неперемищаемой области является горизонтальная линия, отображаемая в нижней ее части. При этом вертикальная полоса прокрутки для перемещаемой области страницы должна быть расположена так, чтобы ее верхняя стрелка находилась ниже границы неперемищаемой области, не перекрываясь этой областью (рис. 6.17).

Рис. 6.17. Неперемищаемая область в окне Справочника

Использование ссылок

Ссылка — это кнопка или интерактивная область, щелчок на которой активизирует некоторое событие. С помощью ссылок может быть реализована однонаправленная связь между страницами Справочника, а также переходы в пределах одного окна либо к другому Help-файлу.

Ссылка может также использоваться для вызова всплывающего окна, содержащего пояснительную информацию о каком-либо понятии или объекте, подобно всплывающему окну контекстно-зависимой подсказки.

Кроме того, ссылка может служить средством запуска некоторых команд, аналогично тому, как кнопки-акселераторы используются в окне Раздела задания.

Ссылка должна визуальнo отличаться от других, не интерактивных областей окна. Например, она может быть реализована в виде кнопки, либо может изменяться форма указателя, когда пользователь перемещает его на ссылку, либо для нее могут применяться специальные цвет или шрифт (или комбинация этих средств). По умолчанию для обозначения текстовых ссылок используется подчеркнутый шрифт зеленого цвета.

Броузер разделов

Броузер разделов представляет собой диалоговую панель, которая обеспечивает пользователю доступ к перечню разделов справочной информации. Для вызова этой диалоговой панели раздел меню *Справка* должен содержать соответствующий пункт (например, *Разделы справочника*), либо команды, позволяющие открывать конкретную вкладку панели (*Содержание*, *Предметный указатель* или *Поиск*). Кроме того, соответствующая кнопка вызова должна присутствовать на панели инструментов окна приложения или окна Справочника.

Когда пользователь вызывает браузер разделов, он должен быть открыт на той вкладке, с которой пользователь работал последний раз.

Как было указано ранее, стандартный браузер разделов содержит три вкладки (*Содержание*, *Предметный указатель* и *Поиск*). Однако разработчик имеет право создать собственную дополнительную вкладку.

Вкладка Содержание

На данной вкладке отображается список тем, упорядоченный по их содержанию, как показано на рис. 6.18. Пиктограмма в виде книги указывает на наличие нескольких связанных тем (разделов), а пиктограмма в виде страницы обозначает одну отдельную тему. Каждый раздел или группа разделов могут входить в раздел более высокого уровня, однако не рекомендуется использовать более трех уровней вложенности, поскольку это может затруднить пользователю поиск информации.

Кнопки, расположенные в нижней части вкладки, позволяют пользователю открывать или закрывать выбранную тему (группу тем), а также выводить на печать их содержимое. Открыть интересующий раздел пользователь также может с помощью двойного щелчка ЛКМ на его названии или пиктограмме.

Рис. 6.18. Вкладка Содержание броузера разделов

Порядок расположения тем в списке и их наименование определяет разработчик приложения, исходя из того, что пользователь должен уяснить взаимосвязь между темами и характер приведенной в них информации.

Вкладка Предметный указатель

Список, отображаемый на этой вкладке, упорядочен по ключевым словам, которые расположены в алфавитном порядке; перечень ключевых слов для каждой темы определяет разработчик (рис. 6.19).

Для получения справки по интересующему его понятию пользователь может либо ввести ключевое слово в текстовом поле, либо выбрать его из списка. Кнопка *Вывести (Display)* выводит на экран раздел Справочника, содержащий это ключевое слово. Если имеется несколько разделов, в которых используется данное ключевое слово, то открывается дополнительное вторичное окно, которое позволяет пользователю выбрать одну из этих тем, как показано на рис. 6.14. Вы можете также использовать это окно, чтобы отобразить связанные темы, когда пользователь выбирает кнопку *Смотри также (See Also)* в окне Справочника.

При выборе ключевых слов следует руководствоваться следующими основными принципами:

- Используйте одновременно два варианта ключевых слов: для начинающих и подготовленных пользователей.
- Используйте несколько синонимов для обозначения одного понятия.
- Придерживайтесь общепринятой терминологии для данной предметной области.
- Используйте как достаточно общие понятия, относящиеся к данной теме, так и более конкретные.

Рис. 6.19. Вкладка Предметный указатель броузера разделов

Вкладка Найти (Find)

Вкладка *Найти* обеспечивает поиск любого указанного пользователем слова (или фразы) по всему тексту Help-файла. Эта возможность требует наличия индексного файла, который либо создается разработчиком заранее, при создании Help-файла, либо динамически, когда пользователь выбирает соответствующую команду на вкладке *Найти* (рис. 6.20).

В связи с усилением ориентации разработчиков программного обеспечения на требования и особенности Интернета пользовательский интерфейс приложений в последнее время претерпевает существенные изменения. Анализ этих тенденций посвящен отдельный раздел книги. Тем не менее, поскольку указанные изменения относятся и к средствам помощи пользователю, здесь уместно привести новый формат Справочника, предложенный фирмой Microsoft, и дать его краткую характеристику.

Рис. 6.20. Окно Мастера для создания базы данных поиска

Рис. 6.21. Новый формат Справочника

С точки зрения пользовательского интерфейса новый Справочник представляет собой первичное окно, разделенное на два подокна; одно из них обеспечивает работу с браузером разделов, а второе предназначено для отображения содержимого выбранного раздела (рис. 6.21).

Новый формат Справочника можно условно назвать «Интернет-ориентированным», поскольку и по внешнему оформлению, и по имеющимся в окне элементам управления он согласован с «Интернет-обозревателем» MS Internet Explorer. В частности, через меню *Параметры* (команда *Параметры Интернета*) можно получить доступ к панели свойств Internet Explorer.

Броузер разделов в новом Справочнике имеет прежнюю структуру; при необходимости его окно можно скрыть, либо воспользовавшись соответствующей командой (или кнопкой на панели инструментов), либо переместив к левой границе окна полосу разделения подокон.

Еще одним важным признаком Интернет-ориентированности нового Справочника является отказ от двойного щелчка ЛКМ как средства ускоренного открытия (в данном случае — раздела Справочника). Теперь выбор раздела происходит при позиционировании на нем указателя мыши (при этом изменяется форма указателя — рис. 6.22), а для открытия выбранного раздела достаточно однократного щелчка ЛКМ.

Рис. 6.22. Выбор раздела Справочника

6.5. МАСТЕРА

Мастер (Wizard) представляет собой специальную форму помощи пользователю, которая позволяет автоматизировать выполнение задания посредством ведения диалога с пользователем. Мастера используются в тех случаях, когда выполняемое задание является достаточно сложным и требует значительного опыта в работе с приложением. Вообще же диапазон применения Мастеров весьма широк: с их помощью может быть автоматизировано практически любое задание, включая создание новых объектов (например, построение графика) или форматирование уже существующих (например, таблицы или параграфа). Тем не менее, Мастеров не следует использовать для обучения пользователя: хотя они и помогают пользователю в выполнении задания, Мастера должны разрабатываться таким образом, чтобы скрыть от него многие шаги задания и внешне упростить задание. Аналогично, Мастера не должны использоваться в качестве электронного учебника; Мастер должен оперировать с реальными данными. Для консультации пользователя следует применять либо рассмотренные выше средства помощи (в частности, проблемно-ориентированную помощь), либо специальные средства обучения, о которых будет сказано ниже. Не стоит также рассчитывать на то, что применение Мастеров позволит повысить качество непродуманного или сложного интерфейса приложения.

Более того, Мастеров следует рассматривать лишь как дополнение к основным инструментам приложения, предназначенным для выполнения конкретного задания.

Другими словами, Мастер — это средство помощи пользователям, позволяющее им наиболее эффективным образом выполнить достаточно сложное или редко встречающееся задание. В связи с этим доступ к Мастерам не обязательно должен быть реализован так же, как к другим средствам помощи. Для этого может быть использован целый ряд иных способов, например кнопки панели инструментов или специальные пиктограммы, обеспечивающие доступ к так называемым *шаблонам (templates)*.

С точки зрения пользовательского интерфейса Мастер — это набор диалоговых панелей, последовательно отображаемых на экране по мере выполнения пользователем очередного шага задания.

Система обеспечивает поддержку создания Мастеров на базе стандартных панелей свойств. Каждая такая панель содержит элементы интерфейса, позволяющие пользователю ввести (или выбрать) данные, необходимые для выполнения очередного шага задания.

При необходимости Мастер может быть реализован и на основе связанных вторичных окон «общего назначения», однако это может привести к неэффективному использованию пространства экрана и затруднит ориентацию пользователя. В связи с этим рекомендуется включать в состав Мастера не более одного вторичного окна.

В общем случае каждое окно Мастера должно содержать следующие кнопки (табл. 6.2).

Таблица 6.2.

Кнопки, используемые в окнах Мастера

Команда	Действие
< Назад (Back)	Возврат на предшествующую страницу (кнопка недоступна на первой странице)
Далее > (Next)	Переход на следующую страницу в последовательности; выполняется независимо от того, какие параметры установил пользователь на предшествующих страницах
Готово (Finish)	Применяет введенные пользователем или установленные по умолчанию значения параметров со всех страниц и иницирует
Отменить (Cancel)	Отменяет все установленные пользователем значения параметров, завершает процесс и закрывает окно Мастера

Перечисленные выше кнопки помещаются в нижней части окна, причем во всех окнах они должны располагаться в одной и той же последовательности.

Заголовок окна Мастера должен однозначно идентифицировать его назначение. Дополнительно в меню окна Мастера можно включить команду контекстно-зависимой помощи *Что это?*

На первой странице Мастера целесообразно поместить (слева) графический символ, отражающий назначение Мастера, либо ожидаемый результат его работы (рис. 6.23). В правой верхней части окна обычно отображается приветствие и краткое пояснение предназначения Мастера. Первая страница может также содержать элементы управления для ввода или редактирования данных, которые должны использоваться Мастером (если для этого достаточно места). Тем не менее, и на первой, и на остальных страницах

Мастера не рекомендуется предлагать пользователю слишком много различных вариантов выбора (их должно быть не более пяти-семи).

На последующих страницах также может присутствовать графика (что обеспечивает визуальную согласованность страниц); при этом необходимо иметь в виду, что используемые рисунки должны носить не просто иллюстративный, а пояснительный характер (как показано на рис. 6.24). Если же текст и элементы управления занимают все пространство окна, то графические иллюстрации могут отсутствовать. По возможности для всех устанавливаемых пользователем параметров должны быть предусмотрены значения, используемые по умолчанию.

Рис. 6.23. Первая страница Мастера

Кнопка *Готово (Finish)* может быть помещена в любой точке процесса (т.е. на любой странице), где Мастер может перейти к выполнению задания. Например, если приложение способно обеспечить приемлемый результат, используя значения параметров, установленные по умолчанию, то эта кнопка может присутствовать даже на первой странице. В любом случае кнопка *Готово* должна располагаться крайней справа, после кнопки *Далее*. Это позволит пользователю решить, стоит ли ему переходить на следующую страницу, или завершить ввод требуемых значений в данной точке. Если же от пользователя требуется последовательно пройти все страницы Мастера, то кнопка *Готово* должна присутствовать только на последней странице; при этом она заменяет кнопку *Далее*. На последней странице Мастера следует также сообщить пользователю, что Мастер подготовил задание к выполнению и для его завершения необходимо нажать кнопку *Готово*.

Рис. 6.24. Страница ввода данных для Мастера

При разработке Мастера следует помнить, что и его назначение в целом, и каждый шаг в отдельности должны быть понятны пользователю. Исходя из этого, лучше создать большое количество простых страниц с меньшим числом вариантов выбора, чем наоборот — несколько сложных страниц, насыщенных большим числом параметров или текстом. Кроме того, при проектировании Мастера целесообразно придерживаться следующих основных правил:

- Количество страниц, требующих отображения дополнительного вторичного окна, должно быть минимальным.
- Избегайте такой организации работы Мастера, при которой пользователь будет вынужден самостоятельно выполнять некоторые шаги задания.
- Графические элементы страниц Мастера должны быть реализованы таким образом, чтобы пользователю было понятно, какие из них носят иллюстративный характер, а какие являются интерактивными элементами интерфейса; как правило, иллюстрации имеют больший размер и более абстрактную форму.
- Избегайте автоматического «пролистывания» страниц Мастера, поскольку пользователь может не успеть прочитать информацию. Кроме того, использование Мастера предполагает возможность управления со стороны пользователя выполняемым заданием.
- При инициализации Мастера его окно должно появляться на экране таким образом, чтобы пользователь однозначно воспринимал его как начальную точку выполнения задания.

- Старайтесь убедить пользователя, что значения параметров, предлагаемые Мастером по умолчанию, обеспечивают достижение положительного результата; в некоторых случаях пользователю может быть предложено несколько различных вариантов с соответствующими пояснениями.

- Следует сделать очевидным для пользователя завершение ввода данных, необходимых для выполнения задания; в частности, на последней странице Мастера может быть помещено соответствующее пояснение.

При составлении текстовых комментариев, помещаемых на страницах Мастера, следует использовать диалоговый, а не «справочный» стиль. Практика показывает, что пользователи (особенно начинающие) лучше реагируют на вопросы типа «Как бы Вы хотели это сделать?», чем на сообщения типа «Сделайте то-то». Например, вопрос «В каком формате Вы хотите получить результат?» предпочтительнее, чем предложение «Выберите формат представления результата».

6.6. СРЕДСТВА ОБУЧЕНИЯ ПОЛЬЗОВАТЕЛЯ

Практика показывает, что даже при наличии полной и подробной документации на программный продукт, но выполненной в печатной форме, пользователи предпочитают осваивать его методом «проб и ошибок».

Рассмотренные в предыдущих подразделах средства предназначены для оказания пользователю оперативной помощи, то есть так или иначе эти средства являются контекстно-зависимыми и предоставляют ему «конкретный ответ на конкретный вопрос». Несколько особняком стоит в их ряду Справочник, разделы которого не связаны непосредственно с текущим контекстом выполняемого задания, но и он используется обычно в качестве развернутого «толкового словаря», а не «книги для чтения». Для тех приложений, которые реализуют небольшой перечень функций (причем одним определенным образом), средств оперативной поддержки пользователя вполне достаточно, чтобы помочь ему при возникновении затруднений. Однако для более сложных приложений, которые обычно именуются «интегрированными средами» или «пакетами», требуется создание специальных средств обучения пользователей технологии работы с этими приложениями. Их наличие необходимо и в том случае, если создаваемое приложение предназначено для определенной категории пользователей (либо не знакомых с той предметной областью, к которой относится приложение, либо имеющих недостаточный уровень «компьютерной грамотности»). Номенклатура и способ реализации используемых в приложении средств обучения зависит от целого ряда факторов:

- назначения и уровня сложности приложения,
- характеристик потенциальных пользователей,
- времени, отводимого на изучение приложения.

В рамках одного предложения могут использоваться различные варианты средств обучения, относящиеся как ко всему приложению, так и к отдельным его компонентам (либо предназначенные для различных групп пользователей). При этом пользователю должна быть предоставлена и свобода выбора средств обучения, и возможность отключения любого из них.

Для наиболее сложных программных продуктов средства обучения могут быть реализованы в виде относительно самостоятельных приложений, имеющих в своем составе как подсистему обучения, так и подсистему контроля уровня подготовленности пользователя. Такой подход характерен для систем управления и администрирования, работающих в реальном масштабе времени, ошибка в эксплуатации которых может привести к значительным потерям (в частности, экономическим).

Например, компанией Cisco Systems была создана обучающая программа CIM — Cisco Interactive Mentor («Интерактивный учитель»), которая помогает программистам и сетевым администраторам изучать работу продуктов компании в корпоративных сетях. CIM обеспечивает возможность моделирования поведения сети в различных ситуациях и способствует приобретению пользователями навыков в решении возможных проблем.

Создание подобных систем представляет весьма сложную самостоятельную инженерно-техническую и научную задачу, поэтому мы ограничимся рассмотрением тех средств обучения, которые могут быть реализованы в качестве компоненты пользовательского интерфейса приложения.

С их помощью может обеспечиваться достижение следующих уровней обучения:

1) «стимул-ответ» — вырабатывается точная реакция обучаемого на заданный стимул (например, для вызова всплывающего меню объекта требуется щелкнуть ПКМ на его пиктограмме);

2) обучение цепочкам событий — уяснение требуемой последовательности действий, необходимой для решения определенной задачи (например, для коррекции содержимого файла: открыть – отредактировать - записать);

3) концептуальное обучение — пользователь должен научиться определять общие свойства множества объектов (например, определять перечень разрешенных операций для графических объектов);

4) обучение правилам — предполагает обучение логическому увязыванию между собой концепций (например, пользователь должен уяснить правила применения технологии OLE для объектов различных типов);

5) обучение решению задач — означает формирование навыков в работе с приложением при решении конкретных задач (например, пользователь должен научиться выполнять распределение ресурсов в рамках имеющегося календарного графика).

Практически для всех перечисленных вариантов справедливы следующие общие требования к функционированию средств обучения:

- должны обеспечивать наличие обратной связи (обучаемый должен знать, какой результат дает каждое его действие);
- время реакции системы на действия пользователя должно быть минимальным;
- работа средств обучения в целом должна характеризоваться доброжелательным отношением к обучаемому.

К наиболее распространенным в настоящее время «встроенным» средствам обучения относятся:

- «Полезные советы» (*Tip of the Day*, в терминологии разработчиков локализованной версии MS Office — Мастер подсказок);
- Подборки примеров, иллюстрирующих результаты применения приложения для решения различных задач;
- Демонстрационные ролики;
- Электронные учебники.

«ПОЛЕЗНЫЕ СОВЕТЫ»

Данное средство обучения реализуется в виде последовательности советов по работе с приложением, относящихся, как правило, к вопросам организации пользовательского интерфейса. Для вывода «полезных советов» на экран могут использоваться либо вторичное окно (рис. 6.25 а), либо окно Палитра (рис. 6.25 б). По умолчанию очередной

совет отображается при каждом новом запуске приложения, однако, обычно пользователю предоставляется возможность «пролистать» всю подборку советов, а также право запретить отображать на экране соответствующее окно.

А)

Б)

Рис. 6.25. Средства вывода «полезных советов» на экран

Практика показывает, что данное средство обучения является наименее эффективным и в лучшем случае позволяет обеспечить достижение первого уровня обучения («стимул-ответ»). По нашему мнению, в подавляющем большинстве случаев лучше не включать данное средство в приложение, либо использовать его для знакомства пользователя с действительно полезными советами, позаимствованными, например, у Козьмы Пруtkова или Ларошфуко. Последний, кстати, даже не будучи знаком ни с одним из разработчиков программного обеспечения, как-то заметил: «У всех нас хватит сил, чтобы пережить несчастье ближнего» (читай — пользователя).

ПОДБОРКИ ПРИМЕРОВ И ДЕМОНСТРАЦИОННЫЕ РОЛИКИ

Указанные средства реализуют наиболее традиционный и проверенный принцип обучения — «делай как я». Примеры, иллюстрирующие работу приложения, могут использоваться для обучения пользователя двумя способами:

- в качестве прототипа, изучив и модифицировав который пользователь может получить то, что ему нужно (то есть примеры используются «в статике»); такой подход позволяет реализовать второй и третий уровни обучения;
- в качестве основы для создания демонстрационного ролика, проводящего пользователя по всем основным технологическим этапам работы с приложением (при решении какой-то определенной задачи); в данном случае примеры используются «в динамике» и это обеспечивает достижение четвертого-пятого уровня обучения.

В обоих случаях файлы, содержащие примеры, должны располагаться компактно, в папке с соответствующим названием (например, *Example* или *Sample*). Доступ пользователей к примерам может осуществляться либо стандартными средствами (с помощью команды *Открыть*), либо посредством специальных элементов интерфейса, как показано на рис. 6.26.

Рис. 6.26. Диалоговое окно для доступа к файлам примеров

Для запуска демонстрационных файлов должна быть предусмотрена специальная команда (например, *demo*). Эту команду целесообразно включить в раздел *Help* главного меню приложения, а также поместить в виде кнопки на панели инструментов. Если приложение предназначено для решения достаточно широкого класса задач, то пользователю должны быть предоставлены средства выбора того аспекта работы с приложением, который его интересует (например, с помощью специального окна, как показано на рис. 6.27).

Рис. 6.27. Диалоговое окно для выбора демонстрационного файла

Наилучшим вариантом построения демонстрационного файла является такой, при котором пользователю предоставляется право управлять процессом демонстрации (приостанавливать просмотр, пролистывать фрагменты в ускоренном темпе и т.д.). Указанный подход может быть реализован на основе многооконной технологии построения приложения (например, в виде Проекта, как показано на рис. 6.28).

Если по какой-либо причине сложно реализовать интерактивный режим просмотра, то пользователь должен по крайней мере иметь возможность в любой момент прервать демонстрацию.

Рис. 6.28. Просмотр демо-ролика в интерактивном режиме

ЭЛЕКТРОННЫЕ УЧЕБНИКИ

Основное отличие электронных учебников от демонстрационных примеров — это обязательное наличие интерактивности. В зависимости от сложности и способа реализации они позволяют обеспечить достижение любого из пяти перечисленных выше уровней обучения. Эффективность применения электронного учебника существенно повышается, если в нем используется анимация. Основное ее назначение — иллюстрация материала, изложенного в текущем разделе учебника (рис. 6.29).

Рис. 6.29. «Страница» электронного учебника, сопровождаемая анимацией

Доступ пользователя к электронному учебнику обычно реализуется с помощью соответствующей команды (например, *Tutorial*), входящей в меню *Help*; дополнительно на панели инструментов может присутствовать кнопка, обеспечивающая быстрый доступ к этой команде.

6.7. СРЕДСТВА АДАПТАЦИИ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

Способность приложения учитывать уровень подготовки и психофизиологическое состояние пользователя, а также характер выполняемого задания можно рассматривать как еще один элемент поддержки пользователя. При обсуждении этапов проектирования пользовательского интерфейса в главе 2 были рассмотрены три основных типа его адаптации: полная, фиксированная и косметическая. При использовании GUI реализация каждого из них имеет определенную специфику по сравнению с другими способами организации пользовательского интерфейса. Эта специфика обусловлена в основном тремя факторами:

- использованием концепции Рабочего стола;
- объектно-ориентированным подходом к реализации GUI;
- наличием графических элементов управления.

Благодаря указанным особенностям в GUI-приложении могут быть реализованы следующие виды адаптации пользовательского интерфейса:

- конфигурирование Рабочего стола;
- разграничение прав пользователя по работе с объектами приложения;
- разграничение прав пользователей по использованию элементов управления;
- изменение визуальных атрибутов отображаемой на экране информации.

Первые три пункта представляют собой различные варианты фиксированной адаптации. Например, в зависимости от того, к какой категории относится пользователь,

ему может быть запрещено, разрешено (или разрешено с некоторыми ограничениями) формирование собственной конфигурации Рабочего стола (рис. 6.30).

Рис. 6.30. Настройка параметров Рабочего стола

При необходимости приложение может автоматически выполнить конфигурирование Рабочего стола.

Аналогичным образом может производиться настройка интерфейса и относительно используемых в приложении объектов. В частности, определенным категориям пользователей может быть запрещено подключение принтера (рис. 6.31).

Рис. 6.31. Управление правами пользователей по работе с объектами

Третий вариант фиксированной адаптации — разграничение прав пользователей по использованию элементов управления — предполагает возможность дифференцированного формирования подмножества доступных пользователю элементов управления как в первичном окне приложения, так и во вторичных окнах. Например, в пакете RTWin, предназначенном для создания систем управления реального времени, предусмотрена возможность разграничения прав доступа к элементам интерфейса для различных категорий операторов.

Как правило, основное предназначение рассмотренных выше способов фиксированной адаптации — обеспечение требуемого уровня безопасности (в смысле защищенности) системы. Тем не менее, их использование во многих случаях способно существенно облегчить работу недостаточно опытным пользователям («меньше знаешь — спокойнее спишь»).

Изменение визуальных атрибутов отображаемой информации является для GUI-приложений одним из основных способов косметической адаптации. Если «инициатором» фиксированной адаптации выступает, как правило, приложение, то при выполнении косметической адаптации интерфейса ведущая роль отводится пользователю. Однако если в приложении предусмотрены дополнительные способы настройки интерфейса (по сравнению со стандартным минимумом), оно должно каким-то образом известить об этом пользователя. С этой целью может использоваться, например, окно «Полезные советы», отображаемое на экране при первом запуске приложения.

Стандартные методы косметической адаптации — выбор цветовой палитры, шрифтов, изменение расположения панелей инструментов и т.д. — были рассмотрены при описании свойств элементов интерфейса. В этом подразделе остановимся немного подробнее на тех способах косметической адаптации, которые значительно реже используются в приложениях (хотя также основаны на стандартных свойствах элементов интерфейса), но оказывают значительно большее влияние на эффективность работы пользователя.

В эту группу, в частности, входят:

- назначение (или переопределение) клавиш-акселераторов;
- изменение состава отображаемых пунктов меню;
- изменение состава кнопок панелей инструментов;
- изменение формы представления кнопок панелей инструментов.

Использование клавиш-акселераторов позволяет во многих случаях существенно повысить скорость работы пользователей, особенно тех, которые испытывают затруднения при работе с мышью. Их применение дает существенный эффект и при выполнении наиболее распространенных операций. Напомним, что назначение (или

переопределение) клавиш-акселераторов производится с помощью специального текстового поля, которое так и называется - поле назначения горячих клавиш. Это поле обычно помещается на панели свойств с соответствующим названием (например, *Параметры* или *Настройки*) и используется вместе со списком, отображающим текущее распределение горячих клавиш (рис. 6.32).

Если приложение может использоваться несколькими пользователями, целесообразно предусмотреть средства раздельного сохранения индивидуальных подмножеств горячих клавиш для каждого из них.

Рис. 6.32. Элементы управления, используемые для назначения (переопределения) горячих клавиш

В главе 2 было отмечено, что наличие на экране избыточной информации не только затрудняет пользователю отыскание требуемых ее элементов, но и способствует его повышенной утомляемости; это, в свою очередь, приводит к снижению эффективности работы пользователя.

Один из способов устранения избыточности информации - предоставление пользователю права изменять состав отображаемых пунктов (и разделов) меню и кнопок панелей инструментов. Реализация такой возможности является нетривиальной задачей, которая может решаться по-разному в зависимости от используемого языка и технологии программирования. Способ ее решения влияет на архитектуру приложения, поэтому вопросы об использовании указанных средств адаптации интерфейса должны быть решены на самых ранних стадиях проектирования приложения. На рис. 6.33 приведен пример удачной реализации средств настройки меню пользователем, а на рис. 6.34 ~ пример не менее удачного подхода к настройке панели инструментов приложения.

Диапазон настроек панели инструментов приложения может быть значительно расширен за счет предоставления пользователю права включать в нее кнопки, обеспечивающие выполнение его собственных команд, то есть исполняемых файлов (.exe, .com, .bat). Это позволяет использовать панель инструментов вашего приложения для запуска других приложений, с которыми потребуется взаимодействовать пользователю при выполнении того или другого задания. Такой подход позволяет сформировать на базе одного приложения устойчивую рабочую среду пользователя, которая будет восстанавливаться каждый раз при запуске приложения. На рис. 6.35 приведен пример включения в панель инструментом кнопки запуска текстового процессора MS Word.

Описанные выше варианты косметической адаптации ориентированы на «среднестатистического» пользователя. Вместе с тем, во многих случаях следует учитывать особенности специфических категорий пользователей, например, людей с ослабленным зрением или нарушенным цветовосприятием. Наиболее распространенный вариант решения этой проблемы - предоставить пользователям возможность выбора формы визуального представления элементов управления. С этой целью соответствующие команды могут быть включены во всплывающее меню элемента управления, подлежащего настройке (рис. 6.36).

Рис. 6.33. Панель свойств для настройки пользователем меню

Рис. 6.34. Панель свойств для настройки пользователем панели инструментов

Рис. 6.35. Включение в панель инструментов кнопки запуска текстового процессора

Реализация такого способа адаптации предполагает наличие нескольких вариантов визуального представления элементов управления. Для кнопок управления, например, могут использоваться различные способы их обозначения: крупные и мелкие рисунки, совмещение графики с текстовыми метками (либо замена графики текстом), как показано на рис. 6.37.

Возможны и другие способы адаптации. Например, компонента ОС MS Windows 2000, которая называется Microsoft Magnifier («Увеличительное стекло»). Она позволяет превратить часть экрана в своеобразную «лупу» с регулируемым увеличением, которая перемещается в нужную точку вместе с указателем мыши.

Рис. 6.36. Всплывающее меню для выбора формы представления кнопок панели инструментов

Рис. 6.37. Дополнительные форматы кнопок панели инструментов

Широкий спектр параметров косметической адаптации пользовательского интерфейса связан с использованием мультимедийных средств. Ориентация на применение этих средств проявляется все в большем числе приложений. Вместе с тем говорить о стандартных подходах к их проектированию и включению в качестве обязательной компоненты интерфейса нам представляется несколько преждевременным. Отметим лишь в качестве примера, что в состав ОС MS Windows 2000 входит компонента Microsoft Screen Reader, которая позволяет озвучивать изображение на экране (названия кнопок, пунктов меню и т.д.). К сожалению многих русскоязычных пользователей, эта компонента пока не локализована. В связи с этим уместно еще раз вспомнить о такой форме косметической адаптации, как поддержка родного языка пользователя. Эффективная реализация многоязычности GUI-приложений стала возможна благодаря появлению международного стандарта представления алфавитов Unicode UTF-8. Первый известный нам программный продукт, полностью реализующий такую возможность (правда, пока «только» для 30 языков) - это средство динамического анализа данных Gentia 4.0 (производитель — фирма Gentia Software). В Gentia 4.0 вся текстовая информация хранится в независимом от языка виде до тех пор, пока пользователь не запустит программу на исполнение. Преобразование данных в конкретный язык достигается за счет использования двух типов таблиц: таблиц приложения и таблиц сообщений. Первые переводят в UTF текстовые строки, применяемые в приложении для обозначения элементов управления (например, названия пунктов меню). Таблицы сообщений содержат переводы каждой такой строки на другие языки и позволяют пользователям запускать приложение на базе выбранного ими языка.

Полная адаптация интерфейса, как было отмечено в главе 2, предполагает наличие динамически корректируемой модели пользователя. При этом основное ее отличие от фиксированной и косметической форм адаптации заключается в том, что она инициируется не пользователем, а самим приложением. Для реализации такого вида адаптации пользовательский интерфейс приложения должен обладать как бы

двунаправленной обратной связью: информация о действиях пользователя должна не только визуализироваться, но и регистрироваться приложением с целью формирования модели пользователя. Первые удачные попытки реализации такого подхода связаны с появлением Интернет-технологий, а точнее - с появлением специфической формы GUI — Web-интерфейса, которая с одной стороны становится все более самостоятельной, а с другой - оказывает все большее влияние на своего «предка». Более подробно вопросы проектирования Web-интерфейса рассмотрены в следующей главе. Здесь лишь приведем один из примеров полной адаптации интерфейса, реализуемого при работе пользователей в Интернете. Речь идет о программном продукте под названием Learn Sesame 1.2, выпущенном в феврале 1998 года фирмой Charles River Analytics. Данный продукт отслеживает все действия посетителя в ходе его пребывания на Web-узле и строит «профиль» (набор характеристик) этого пользователя. После того, как в базе данных Learn Sesame 1.2 накопится достаточно сведений о посетителе, ему будет предложено индивидуально настроенное содержимое узла. Например, в канадской фирме Newstar Technologies программное обеспечение на основе Learn Sesame 1.2 используется для предоставления клиентам персонализированной финансовой информации; определив, что посетителя интересуют рискованные инвестиции, ему предлагают данные о ценных бумагах именно такого рода.

7. ПРОЕКТИРОВАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА: ОТ ОБЩЕГО К ЧАСТНОМУ

При всей общности рассмотренных в предыдущих главах принципов построения пользовательского интерфейса читатель, вероятно, заметил, что они излагались с ориентацией на среду Windows (а если точнее, то на Windows 95) и офисные приложения. В связи с этим вполне закономерен вопрос: насколько эти принципы применимы для реализации интерфейса на других платформах и в других предметных областях? Для ответа на этот вопрос мы рассмотрим особенности организации интерфейса в двух сферах применения информационных технологий, которые с точки зрения конечных пользователей являются антиподами: *Интернет и Системы управления реального времени*. При работе в Интернете действует «классический» принцип демократии — «разрешено все, что не запрещено», а работа в системах управления основана на принципе «запрещено все, что не разрешено». Тем не менее, и перед WEB-мастерами, и перед разработчиками автоматизированных рабочих мест (АРМ) операторов систем управления стоят одни и те же проблемы, и решают они их аналогичными способами. Читатель сможет в этом убедиться, ознакомившись с материалами данной главы.

7.1. ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС WEB-ПРИЛОЖЕНИЙ

Организация пользовательского интерфейса определяется решаемой задачей. Это положение неоднократно подчеркивалось, но в данной главе нелишне его повторить. Стремительное развитие Интернета, и главное, широкое распространение этого комплекса технологий в «пестрых» массах пользователей с самым разным уровнем подготовки, потребовали появления новых пользовательских интерфейсов.

Первые шаги Интернета были связаны с текстовым интерфейсом, ориентированным на профессионалов: программистов, ученых, военных и студентов. Для обмена файлами и пересылки сообщений по каналам с низкой пропускной способностью текстовый интерфейс был вполне пригоден. Да и на «посторонних» здесь явно не рассчитывали. Но «всемирная паутина» росла, менялась аппаратура, накапливались информационные ресурсы, доступ к которым хотели получить многие и многие. Потребность в новом пользовательском интерфейсе созрела и он появился.

Программа Mosaic одна из первых предоставила для работы в Интернете графический интерфейс с использованием мыши. С этого момента, собственно, и началось победоносное шествие WWW по странам и континентам. Родился новый класс программ, предназначенных для работы в Интернете и просмотра документов, полученных из сети и составленных по ее правилам. Эти программы называют **броузерами** (*browser*) или обозревателями сети (есть и другие варианты названий, на пример, навигаторы, гляделки, бродилки или смотрелки, но оставим их обсуждение лингвистам). Броузер позволяет соединиться через модем или сетевую карту с провайдером (продавцом сервисов Интернета) — открыть сеанс работы, а затем, следуя адресам, вводимым пользователем, обеспечивает переход к соответствующим ресурсам (серверам, порталам, сайтам, узлам, ftp-архивам или отдельным документам). Кроме того, броузер позволяет принимать и отправлять электронную почту, получать новости, вести беседу в чате (chat), работать на удаленном компьютере (telnet) или участвовать в телеконференциях. Функций у хороших броузеров так много, что им посвящены объемистые руководства. Тем не менее, их перечисление пригодится нам для дальнейшего изложения. Итак, вот перечень основных задач, решаемых броузером (а задачи, как вы помните, определяют требования к пользовательскому интерфейсу):

- работа с аппаратурой (конфигурирование модема, настройка портов компьютера, учет особенностей линии связи и т.д.);
- открытие и закрытие сеанса работы (дозвон до провайдера, ввод идентификационных данных, согласование параметров);
- ввод адресов информационных ресурсов Интернета (числового IP-адреса или алфавитно-цифрового — URL);
- ожидание получения документа (имитация перехода по ссылке или наоборот — подкачки материалов, что, собственно, на самом деле и происходит);
- визуализация документа;
- информирование о проблемах (уважительные и неуважительные причины отказов);
- поиск в документе;
- сохранение документа (перевод на локальные носители компьютера), его распечатка на принтере;
- редактирование локальных документов;
- настройка самого броузера.

Перечень заведомо не полон, ведь есть еще и упомянутые ранее задачи, связанные с реализацией многочисленных Интернет-технологий (почта, новости, chat, конференции и т.д.). Но и приведенного списка вполне достаточно, чтобы задаться вопросом: достаточно ли «традиционных» средств и методов организации пользовательского интерфейса для эффективного решения этих задач?

Здесь имеет смысл сделать небольшое отступление. Изыскания в области методов построения гибких интерфейсов приводили к интересным, но, как правило громоздким и неуклюжим решениям. Это, разумеется, было не следствием низкого профессионального уровня разработчиков, а проявлением «капиталистических»

отношений в программной индустрии. Борьба с монополизмом Microsoft, развернутая в последние годы, проходит под флагами Интернета. Именно в этой борьбе родилась новая парадигма, основанная на достаточно старой технике текстовой интерпретации. Любой из современных браузеров, помимо всего прочего, является интерпретатором текстов Web-страниц.

В браузере объединены два аспекта, два подхода к построению пользовательского интерфейса:

во-первых, браузер предоставляет пользователю свой собственный интерфейс — как и любое другое приложение какой-либо операционной системы;

во-вторых, браузер реализует пользовательский интерфейс, разработанный автором ресурса, к которому обратился пользователь.

Реализация первого аспекта подчинена, как правило, гласным и негласным правилам и традициям в организации интерфейса приложений для той операционной системы, на которой функционирует браузер. Имеющиеся отличия определяются дополнительными задачами, решаемыми браузером, но и они, чаще всего, соответствуют общим принципам организации взаимодействия с пользователем, принятым в соответствующей операционной среде.

Совсем другие концепции положены в основу интерфейса, реализуемого средствами создания Web-страниц. Исполняя роль интерпретатора текста, содержащегося в отображаемом документе, браузер *обязан* наиболее точно «воплотить в жизнь» замысел автора документа. И здесь почти вся ответственность за организацию взаимодействия с пользователем ложится на создателя ресурса. Именно этим объясняется многоликость Интернета, проявляющаяся в многообразии способов организации информации, выставляемой на всеобщее обозрение. С другой стороны, разные виды ресурсов и технологий требуют разных подходов к построению интерфейса. Добавьте к этому и разнообразие аппаратно-программных платформ, участвующих в создании, развитии и эксплуатации Интернета. В результате возможна ситуация, когда пользователь IBM PC, управляемого Windows 95, может просматривать в браузере Netscape Navigator страницу, полученную с сервера, функционирующего под управлением ОС UNIX, подготовленную на компьютере Macintosh. Вот где поистине начинаешь ценить те усилия, которые вкладываются в согласование и стандартизацию! Без этого Интернет просто перестал бы существовать (точнее, не появился бы). Но вернемся к пользовательскому интерфейсу. При всем нашем желании дать как можно более полную его характеристику, многие чрезвычайно интересные области останутся за рамками нашего обзора; как говаривал Козьма Прутков, «нельзя объять необъятного».

7.2. WEB-СТРАНИЦЫ И САЙТЫ

С развитием Интернета появилось множество новых областей деятельности и даже профессии, одна из которых — Web-дизайнер. Так называют специалиста, создающего основной вид информационных ресурсов сети — Web-страницы, а точнее, их системы. Страницы объединяются с помощью гипертекстовых ссылок (связей) и образуют тематические Web-узлы и сайты (граница между этими категориями весьма условна и касается только масштабы творения или претенциозности автора;

считается, что сайт заметно больше узла и полнее раскрывает основную тему). В последнее время появился еще один термин — портал, под которым в общем случае понимается объединение нескольких тематических направлений. Тем не менее, в некоторых источниках приводится и более развернутая трактовка этого термина:

портал - это Web-узел, предоставляющий посетителю персонализированную начальную страницу, бесплатные услуги электронной почты, новостной и развлекательный сервисы; как правило, портал является точкой входа пользователей в Интернет.

Поскольку и порталы, и сайты состоят из страниц, то именно о них и пойдет речь далее.

Вообще полный цикл создания Web-страниц и сайтов рассматривается в специальной и популярной литературе. В этих изданиях представлен широкий спектр проблем: от чисто эстетических до лингвистических и технологических. Мы рассмотрим те же проблемы, но под определенным углом зрения: как обеспечить единство восприятия документа его автором и пользователем. Для начала выделим основные компоненты Web-документа, формирующие его пользовательский интерфейс (важно отметить, что современные Web-документы, в отличие от привычных текстов с картинками, обладают интерактивностью, то есть в их отношении можно говорить о полноценном пользовательском интерфейсе).

Итак, интерфейс большинства Web-страниц определяют следующие компоненты:

- пассивные элементы страницы (фон, текст, графика, таблицы, разделители, фреймы);
- интерактивные элементы (списки, кнопки, сенсорные карты, формы);
- элементы эстетического оформления (фоновые изображения, звуковое сопровождение, анимационные эффекты);
- средства навигации по странице (документу) и в системе страниц;
- ссылки на внешние ресурсы Интернет.

Не следует также забывать, что Web-документы могут взаимодействовать и с браузером, что обуславливает многие особенности отображения их содержимого. Формами такого взаимодействия, например, могут быть указания о необходимости открытия нового окна, включение или отключение полос прокрутки, запрос на запуск программы, формирующей сложную сцену или выполняющей вычисления. К слову сказать, существуют и прямо противоположные возможности. Пользователь может так настроить браузер, что тот будет практически полностью игнорировать художественные изыскания автора страницы и выводить только текстовое наполнение, да еще и вполне определенным шрифтом. При нынешней стоимости доступа в Интернет, когда многие пользователи вынуждены стать аскетами, необходимо учитывать и эту, не совсем приятную для Web-дизайнера, ситуацию.

Не хочется, чтобы последующие замечания воспринимались как попытка ограничить чью-нибудь творческую свободу, но, заботясь все о тех же пользователях, мы считаем необходимым высказать некоторые наблюдения и рекомендации по использованию перечисленных выше элементов Web-страниц.

ПАССИВНЫЕ ЭЛЕМЕНТЫ

Старайтесь использовать преимущественно спокойные тона для фона; помните, что работа с текстом на белом фоне аналогична чтению надписей на горящей электролампе. Долго всматриваться в такую страницу пользователь вряд ли захочет и уж наверняка испытает внутреннее раздражение по отношению к ее автору. Другой крайностью можно считать применение светлых букв на темном фоне. Здесь есть две «западни». Во-первых, если в качестве фона выбрано темное

изображение, а пользователь отказался от загрузки изображений, (такая возможность есть в браузерах и часто используется для уменьшения трафика в целях экономии времени и денег), то пользователь получит светлые символы, неразличимые на светлом фоне. Во-вторых, психологи и специалисты в области эргономики установили, что цвет оказывает заметное влияние на настроение человека. И это влияние тем сильнее, чем интенсивнее цвет (о влиянии цвета на настроение пользователя подробно говорилось в главе 2).

Если вы испытываете затруднение в выборе цвета, воспользуйтесь личными предпочтениями пользователя и задайте для своей страницы установки цветов по умолчанию (*default*). В этом случае при отображении документа будут использоваться настройки браузера или операционной системы на компьютере пользователя.

ИНТЕРАКТИВНЫЕ ЭЛЕМЕНТЫ

Несомненно, наиболее важным элементом, обеспечивающим интерактивность Web-документов, являются гиперссылки, но именно по этой причине мы выделили их в особый класс и рассмотрим чуть ниже. Здесь речь пойдет о более привычных для пользователей современных персональных компьютеров элементах интерфейса, рассмотренных в предыдущих главах. Главный принцип разработчика должен быть тем же, что и при создании локальных приложений: никаких неожиданностей для пользователя. Продиктован он, как вы понимаете, требованием согласованности интерфейса. Его реализация частично возлагается на браузер, обязанностью которого является обеспечение привычного вида и поведения таких элементов (кнопок, списков и т.п.), а частично - на Web-дизайнера, задача которого — не переусердствовать в изобретении образов тех же кнопок, представленных на странице, например, в виде сенсорных карт.

Область применения интерактивных элементов на Web-страницах достаточно широка, перечислим лишь несколько примеров, которые помогут лучше представить варианты их реализации, согласующиеся с уже сформировавшимися представлениями пользователей.

- Ввод информации для ее пересылки на сервер (на основе заполнения форм);
- Выбор информации из имеющегося перечня (работа со списками);
- Навигация по сети или внутри данного сайта (работа с гиперссылками и сенсорными картами);
- Настройка визуальных атрибутов и содержимого страницы (списки или переключатели вариантов оформления).

ФОРМЫ НА WEB-СТРАНИЦАХ

Читатель, вероятно, помнит, что заполнение готовых форм — это один из вариантов ведения диалога с пользователем. Во времена текстовых интерфейсов он прекрасно себя зарекомендовал, так как сочетает высокую наглядность, поддержку пользователя и более эффективное использование ресурсов компьютера, чем, например, диалог в форме «вопрос-ответ». Современные графические интерфейсы несколько повысили наглядность форм, добавив флажки, переключатели и заменив текстовые указания типа «После заполнения всех полей нажмите клавишу [Enter]» кнопками с требуемым текстовым комментарием. Заполнение таких форм осуществляется заметно быстрее за счет применения мыши и техники прямого ма-

нипулирования. Да и сами формы практически ничем не отличаются от диалоговых панелей, работа с которыми понятна каждому пользователю современного ПК.

СЕНСОРНЫЕ КАРТЫ

Сенсорные карты — это графические объекты, содержащие специальные области, так называемые **активные зоны** (*hotspots*), которые позволяют пользователю перемещаться на связанный (ассоциированный) с картой URL или на другую страницу в пределах того же LJRL посредством щелчка ЛКМ на выбранной зоне. Сенсорная карта может иметь несколько активных зон, с каждой из которых связан собственный URL. Хотя в большинстве случаев сенсорные карты используются для перехода на другую страницу или на другой URL, с их помощью можно также вызывать файлы произвольного типа (например, звуковые).

Сенсорная карта может быть создана на основе любого графического изображения, представленного в одном из поддерживаемых браузерами форматов (.gif или .jpg). В связи с этим у некоторых авторов появляется искушение изобразить на создаваемой странице что-нибудь «этакое», способное, на их взгляд, произвести на потенциального посетителя незабываемое впечатление (например, использовать в качестве сенсорной карты собственную фотографию). Тем не менее, такой подход вряд ли поможет посетителям узла быстро сориентироваться в предлагаемой им информации и решить свои проблемы за минимальное время.

Графическое изображение сенсорной карты должно обеспечивать:

1. Наглядное представление структуры узла (или системы взаимосвязанных узлов);

2. Предоставление посетителю дополнительных сведений, способствующих выбору им собственного рационального маршрута перемещения по узлу; такими сведениями, в частности, могут быть:

- краткая характеристика содержащейся информации;
- относительный объем информации;
- периодичность (или последняя дата) обновления информации.

Пример возможного варианта реализации сенсорной карты показан на рис. 7.1.

Карта сайта "форт-технология"

Рис. 7.1. Вариант реализации сенсорной карты

НАСТРОЙКА ВИЗУАЛЬНЫХ АТТРИБУТОВ ОТОБРАЖАЕМОЙ ИНФОРМАЦИИ

Забота об удобстве пользователя должна стать одним из основных приоритетов Web-дизайнеров (сейчас, к сожалению, на первом месте чаще оказывается демонстрация собственных прикладных навыков или шокирующих эстетических изысков; но будем считать это «болезнью роста», в преодоление которой мы тоже пытаемся внести свою лепту).

Возможность настройки пользовательского интерфейса (эта тема обсуждалась в предыдущей главе) является отличительной чертой любой хорошей программы. На Web-страницах элементы настройки интерфейса хотя и нечасто, но встречаются. Наиболее популярным вариантом настройки является возможность выбора языка и способа кодировки символов для отображения текстового содержимого страниц. Реже предлагается выбрать размер и способ вывода графических изображений, и совсем редко предлагается выбрать один из нескольких вариантов общего дизайна

страницы (простейший случай: текстовый или графический). Большинство авторов ориентируются только на средства настройки, предоставляемые браузерами, либо просто указывают, в каком формате следует просматривать их творения.

Существует несколько веских причин «собственноручной» настройки страниц пользователем:

- необходимость учета характеристик аппаратуры и соединения с Интернетом;
- необходимость учета особенностей применяемого пользователем программного обеспечения;
- языковые предпочтения и индивидуальные психофизиологические характеристики людей, для которых эти страницы созданы.

Конечно, нереально удовлетворить пожелания всех пользователей, но знание аудитории, на которую ориентированы конкретные материалы, поможет в этом не легком деле. Мы не будем подробно освещать тактику и технику решения названных проблем, этому учат пособия по Web-дизайну. Остановимся на вопросе организации интерфейса для настройки визуальных атрибутов страницы.

Рекомендация первая. О возможности настройки страницы пользователь должен быть уведомлен сразу, как только он ее открывает; соответствующее сообщение может быть выведено в отдельном фрейме (лучше — в верхней части окна).

Рекомендация вторая. Если страница содержит много графики (которая загружается значительно медленнее текста и потому появляется на странице в последнюю очередь), следует предусмотреть текстовый вариант переключателя, реализуемого в форме меню или списка. Таким образом, оформляются, например, переключатели используемого языка и кодировки символов текста страницы. В некоторых случаях переключатели языка имеют вид небольших изображений с символикой соответствующих национальностей (например, флага государства). Этот вариант хорош своей наглядностью и компактностью, но предполагает определенный уровень образованности и сообразительности пользователя. Если в браузере отключен вывод графики, предлагаемый вариант должен предусматривать вывод альтернативного текста, что легко реализуемо в языке HTML (Hyper Text Markup Language - язык гипертекстовой разметки), но используется не всегда. На рис. 7.2 приведены некоторые варианты оформления переключателей вида страницы.

Б) Переключатель языка в виде RUS | USA
текстовых гиперссылок

В) Переключатель языка на ный WIN-1251 | KOI-8 | MAC
основе изображения переключателей для настройки страницы

УПРАВЛЕНИЕ СОСТАВОМ ОТОБРАЖАЕМОЙ ИНФОРМАЦИИ

Интересы пользователей, посещающих те или иные информационные ресурсы Интернета, достаточно разнообразны, и не всегда совпадают с тематикой и широтой охвата, предлагаемыми разработчиком. Идеальной можно считать ситуацию, когда пользователь, посетив тот или иной сайт, получил все необходимые ему сведения и ни разу не отвлекся на что-то лишнее, избыточное или не относящееся к делу. К сожалению, реальность еще очень далека от идеала. В стремлении наиболее полно удовлетворить информационные потребности

потенциальных пользователей, а часто просто в наивной надежде объять необъятное, некоторые разработчики наполняют свои сайты всевозможными сведениями, разобраться в которых бывает достаточно сложно. Существующие на данный момент инструменты не позволяют непосредственно встроить в Web-страницу средства фильтрации предоставляемой информации на основе предпочтений конкретного пользователя. В этой связи уместно еще раз привести в качестве примера программный продукт Learn Sesame, выполняющий индивидуальную настройку узла по набору характеристик пользователя. Правда, для уяснения интересов пользователя требуется достаточно длительное время (либо неоднократное посещение узла пользователем). Этого недостатка лишены так называемые **групповые фильтры**, которые обеспечивают практически мгновенную настройку узла на определенный класс (группу) посетителей, имеющих сходные характеристики. Примером такого фильтра является программа GroupLens фирмы Net Perceptions. Как групповые фильтры, так и инструменты типа Learn Sesame достаточно сложны, имеют высокую стоимость (порядка \$30 000) и ориентированы на применение на крупных Web-узлах. Поэтому для многих Web-дизайнеров одним из наиболее доступных средств индивидуализации предоставляемой информации являются рассмотренные выше сенсорные карты, помогающие пользователю выбрать собственный маршрут перемещения по узлу.

ПОИСКОВЫЕ СИСТЕМЫ

Поразительные темпы роста количества информационных ресурсов Интернета не могли не стимулировать появления специализированных сервисов, обеспечивающих систематизацию и поиск информации. В настоящее время можно с уверенностью сказать, что информационно-поисковые системы — ИПС (называемые также **поисковыми машинами**) являются наиболее популярными ресурсами, без которых не обходится ни один из пользователей Интернета.

Повторим еще раз, что хороший интерфейс должен отвечать двум главным требованиям: точное соответствие проблемной области и максимальная дружелюбность пользователю. Степень соблюдения этих требований отражается в такой трудно формализуемой характеристике, как удобство использования программной системы.

Пользовательский интерфейс ИПС должен обеспечивать удобство формирования информационных запросов, удобство просмотра и анализа результатов поиска и удобство доступа к найденным в результате поиска документам.

Удобство формирования информационных запросов обеспечивается:

- полнотой и наглядностью языка запросов;
- удобством ввода и редактирования запросов;
- доступностью и качеством составления справочной информации. Удобство просмотра и анализа результатов поиска определяется:

- структурой отчета о результатах поиска;
- удобством навигации (перемещения) по этому документу;
- составом и объемом сведений о найденных документах, включаемых в отчет.

Удобство доступа к найденным документам зависит от:

- полноты и качества описания документов в соответствующих пунктах отчета;
- наличия и корректности ссылок на документы;

- корректности реакции на ситуации, связанные с недоступностью документа (отключен сервер, изменен или удален документ с момента последнего индексирования и т.д.)

В качестве примера, иллюстрирующего принципы построения и функционирования поисковых машин, рассмотрим систему отечественного производства под названием Яндекс (или Япаех).

Пользовательский интерфейс Яндекс в целом соответствует известному стандарту CUA фирмы IBM, определяющему общие принципы организации текстовых и оконных интерфейсов большинства современных программ. Поэтому его изучение и использование не представляет трудностей для людей, знакомых с современным программным обеспечением персональных ЭВМ.

Общий вид первой поисковой страницы сервера www.yandex.ru приведен на рис. 7.3.

Рис. 7.3. Первая страница поисковой машины Яндекс

От одного обращения к другому вид экрана может изменяться вследствие обновления рекламной и текущей информации, но это не должно смущать посетителей Интернета, привыкших к такой динамике. Главное - это общая структура экрана, а она в Яндекс достаточно проста и наглядна; страница не перегружена информацией, но содержит практически все необходимое для начала плодотворной работы.

В процессе поиска Яндекс вычисляет для каждого найденного документа величину релевантности (соответствия) содержания этого документа поисковому запросу. Список найденных документов перед выдачей пользователю сортируется по этой величине в порядке убывания. Релевантность документа зависит от ряда факторов, в том числе от частотных характеристик искомых слов, веса слова или выражения, близости искомых слов в тексте документа друг к другу и т.д.

Пользователь может повлиять на порядок сортировки, используя операторы веса и уточнения запроса.

В отличие от некоторых словарных поисковых систем, обрабатывающих (индексирующих) только вполне определенные части исходных документов, например, заголовки или первые 100-200 строк, Яндекс является средством полнотекстового поиска информации, то есть работает с каждым документом от начала до конца. Важным свойством Яндекс является то, что эта система работает с учетом морфологии (законов формообразования слов) русского языка. Это свойство означает, что если в запросе есть слово «идти», то в результате поиска будут найдены ссылки на документы, содержащие слова «идти», «идет», «шел», «шла» и т.д. Учет морфологии существенно повышает удобство и эффективность использования поисковой системы, освобождая пользователей от необходимости составления сложных запросов. Помимо этого сокращается объем индексных файлов, так как слова в них хранятся в одной — «нормализованной» форме, что, помимо экономии ресурсов сервера, сокращает время поиска информации.

ЭЛЕКТРОННЫЕ МАГАЗИНЫ

С распространением в Интернете электронной коммерции все более актуальной становится задача побуждения посетителей к повторному визиту. В связи с этим

для порталов, реализующих функции электронных магазинов, на первое место выходит необходимость реализации на них средств коллективной фильтрации, ранжирования и ускорения выдачи результатов поиска. Таким образом, эффективность электронной коммерции опять-таки зависит от умения приспособливаться к интересам пользователей. Своевременно осознав это, фирмы DeJaNews и HotBot разрабатывают механизмы коллективной фильтрации, через которые будут собираться отзывы пользователей о просмотренных ими материалах. Тем не менее, даже не имея в своем распоряжении таких инструментов, разработчик электронного магазина может достичь требуемого уровня его посещаемости, используя следующие рекомендации:

- предоставьте потенциальному покупателю несколько способов поиска нужных ему товаров;
- продумайте классификацию предлагаемых товаров, сопроводив каждую группу товаров краткой характеристикой;
- при использовании графических иллюстраций учитывайте скорость их загрузки;

в этом отношении очень полезным может оказаться применение миниатюрных изображений, которые будут служить ссылками на полномасштабные версии иллюстраций;

- обеспечьте посетителям легкий и удобный доступ к дополнительной информации по заинтересовавшему их товару;

- не жалейте времени и сил на эстетическое оформление создаваемого узла; в связи с этим уместно еще раз привести соображение, высказанное в первой главе книги: чтобы привлечь внимание человека, совсем не обязательно хватать его за рукав.

ТАБЛИЦЫ СТИЛЕЙ

Весь предшествующий материал, посвященный Web-дизайну, должен был убедить читателя в том, что весьма сложно выбрать такой стиль оформления Web-узла, который отвечал бы как вкусам разработчика, так и интересам посетителей. Но, пожалуй, еще труднее многократно воспроизвести найденное удачное решение, не забыв при этом тот или иной нюанс. И чем крупнее узел, тем острее встает проблема сохранения стиля на всех его страницах. А как быть в том случае, если первоначальный проект удалось улучшить уже после того, как узел, насчитывающий десятки страниц, был «сдан в эксплуатацию»?

Решение указанных проблем было найдено в 1996 году, когда Консорциумом 3W (WWW Consortium - W3C) была стандартизована технология иерархических таблиц стилей (Cascading Style Sheets - CSS). Суть технологии CSS заключается в том, что она позволяет добавлять в HTML-код описание «стиля страницы», содержащее такие атрибуты, как тип шрифта, цвет, отступы, способ выравнивания элементов теги т.д. Другими словами, таблицы стилей по своему предназначению аналогичны шаблонам текстовых документов, используемым, например, в редакторе MS Word.

Используя CSS, Web-дизайнер может создать один файл с таблицей стиля и затем применить его ко всем страницам узла. Соответственно, любое изменение в таком файле приведет к автоматической корректировке всех использующих его страниц.

Иерархия таблиц стилей имеет два аспекта.

Во-первых, речь идет о том, что для одного Web-документа может существовать одновременно несколько таблиц стилей, образующих иерархию; например, может существовать таблица стилей, общая для всех страниц узла, но при этом для некоторых страниц может быть определен индивидуальный стиль. В таких случаях реализуется стиль самого нижнего уровня.

Во-вторых, иерархию образуют таблицы стилей, созданные автором узла (страницы) и посетителем: если созданные ими стили конфликтуют, то приоритет отдается авторскому стилю (хотя, следуя золотому правилу «клиент всегда прав», браузеры позволяют изменить иерархию на противоположную).

ЕЩЕ РАЗ О СТАНДАРТИЗАЦИИ

Стремительное развитие Интернета породило проблему Интернет-стандартов, суть которой состоит в споре между поставщиками и пользователями о том, кто же, собственно, должен формировать эти самые стандарты. Распространение различных групп и тактик, быстрые технические перемены и необходимость ускорения текущих работ по стандартизации делают мониторинг стандартов делом практически невозможным. Эксперты отмечают, что даже большим корпорациям, которые традиционно имели возможность выделять персонал и ресурсы для тщательной проработки документов со спецификациями, с трудом удается увязывать готовую продукцию с тем или иным действующим стандартом.

Пытаясь совместить такие понятия, как согласованность продуктов разных компаний, с одной стороны, и беспощадную рыночную конкуренцию — с другой, некоторые промышленные группы создают стандарты на базе популярных коммерческих технологий. Например, большинство групп, занимающихся стандартами на программное обеспечение, пользуются процедурами быстрого отслеживания, которые позволяют им оперативно обращаться к производителям или группам производителей с просьбой формулировать спецификации и представлять их на рассмотрение в качестве отправных точек для эталонных реализации тех или иных технологий.

В настоящее время реальное влияние на процесс стандартизации технологий, используемых в Интернете, оказывают следующие организации и группы.

World Wide Web Consortium (W3C).

Консорциум, созданный в 1994 году, насчитывает около 200 членов, представляет интересы конечных пользователей, научных учреждений и компьютерных фирм и имеет солидную международную репутацию. Его представительства есть в лаборатории компьютерных наук Массачусетского технологического института, Национальном институте информатики и автоматизации во Франции и в университете Keio University Shonan Fujisawa Campus в Токио.

W3C занимается тем, что предлагает и поддерживает Web-технологии, а также публикует коды эталонных реализации. W3C работает с технологиями браузеров, такими как HTML и XML, и рядом стандартов, включая HTTP адреса URL и цифровые сертификаты. Одна из основных заслуг этой организации состоит в том, что ей удалось определить некие общие рамки, позволившие объединить идеи двух конкурирующих компаний — Microsoft и Netscape - относительно применения указанных технологий.

Internet Engineering Task Force (IETF) определяет себя как «самоорганизованная группа людей», которая представляет на рассмотрение и оценивает новые Интернет-технологии. В IETF нет официального членства, но

группа собирается регулярно, три раза в год. Она взаимодействует с Internet Society, Internet Engineering Steering Group и Internet Architecture Board. В компетенцию IETF входят проблемы архитектуры Интернета и разработка технологий, обеспечивающих работу в Сети, таких как Electronic Data Interchange, каталоги, календари и планировщики, а также приложения электронной почты.

International Organization for Standardization (ISO) — Международная организация по стандартизации -имеет статус федерации национальных организаций по стандартизации. ISO основана в 1947 году, в нее входит порядка 100 стран. Миссия ISO — всемерно поддерживать разработку стандартов по большому спектру продуктов и технологий в разных странах.

7.3. ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС СИСТЕМ РЕАЛЬНОГО ВРЕМЕНИ

Автоматизация процессов управления уходит своими корнями в глубокое прошлое. Собственно, применение ЭВМ в качестве «управляющего органа» началось практически сразу, как только появилась техническая возможность сопрягать их с объектами, подлежащими управлению. Однако при этом ЭВМ (да и те, кто на них работал) продолжали жить в своем, «нереальном» мире, наполненном битами, байтами, плавающими запятыми и другими, непонятными нормальным людям категориями. Даже время в них измерялось по-своему: не днями, не часами, и даже не минутами, а совсем другими единицами - мили- микро- нано- и другими секундами. Более того, во многих случаях интервалы между соседними событиями внутри ЭВМ выражались даже не этими псевдо-секундами, а числом тактовых импульсов, сформированных тактовым генератором. Тем не менее, ЭВМ, включенные в контур управления, вынуждены были синхронизировать свои действия с управляемыми объектами, работающими в реальном мире и, соответственно, в реальном времени. Чтобы отличать такие вычислительные системы от тех, которым реальный мир оставался безразличен, их стали называть «системами реального времени» (СРВ). Одна из наиболее удачных на наш взгляд трактовок этого понятия приведена в [6]:

Система реального времени - это аппаратно-программный комплекс, реагирующий в течение предсказуемого времени на непредсказуемый поток внешних событий». Данное определение требует некоторых пояснений. Во-первых, перечень типов событий, на которые должна реагировать система, как правило, определяется на этапе ее создания; неизвестны «только» последовательность этих событий и моменты их возникновения. Во-вторых, система должна успеть отреагировать на произошедшее событие в течение времени, критичного для этого события (точнее, для управляемого объекта), и это время должно быть предсказано (вычислено) при создании системы. Отсутствие реакции в течение заданного (или допустимого) интервала считается ошибкой. В-третьих, поскольку на управляемых объектах могут происходить два или более событий одновременно, должна быть задана приоритетность каждого из них с точки зрения целевого предназначения системы. Различают СРВ двух типов:

- жесткого реального времени;
- мягкого реального времени.

Для систем жесткого реального времени недопустима задержка реакции ни при каких условиях, поскольку это может привести либо к катастрофическим последствиям, либо к тяжелым экономическим потерям (что, собственно, для потерявшего равносильно катастрофе). К таким системам относятся, в частности,

бортовые системы управления, системы военного назначения, системы аварийной защиты (например, на атомных электростанциях) и некоторые другие.

Для систем мягкого реального времени задержка менее критична, хотя и может привести к снижению качества управления. Например, задержка в оформлении авиабилетов за 10 минут до вылета вряд ли приведет к человеческим жертвам, но определенным образом повлияет на работу аэропорта.

Особый класс СРВ составляют так называемые системы диспетчерского управления (или человеко-машинные системы - ЧМС), в которых одним из обязательных звеньев управления (а иногда и главным) является человек (диспетчер, оператор, или Лицо, Принимающее Решение - ЛПР). Качество работы такой системы в значительной степени определяется тем, насколько адекватно воспринимает оператор поступающую информацию и насколько своевременно он на нее реагирует. Очевидно, что при достаточном уровне подготовленности персонала основным фактором, влияющим на работу оператора, является качество организации его взаимодействия с системой, то есть ее интерфейс. Вместе с тем, даже в случае принятия правильного решения оператор может допустить так называемую функциональную ошибку (нажать не ту клавишу, выбрать не ту команду) и т.д. Опасность функциональных ошибок существенно возрастает в стрессовых ситуациях. Например, опыт американских военных летчиков показывает, что в условиях ведения боевых действий оказывается неэффективным использование меню.

В табл. 7.1. приведены численные значения вероятностей различных типов функциональных ошибок. Если решаемая оператором задача требует выполнения цепочки операций, вероятности возможных ошибок складываются; суммарное значение вероятности ошибки, превышающее 0,03, считается критическим [7].

Таблица 7.1.

Вероятности различных типов функциональных ошибок

Операция	Вероятность ошибки
Актуализация из памяти или запоминание значения параметра	0,0005
Мысленный выбор одной из двух альтернатив	0,0005
Мысленное сравнение ситуации с типовой, требующей определенного действия	0,0010
Чтение (1-3 слова)	0,0010
Ввод текста (1-3 слова)	0,0020
Восприятие символа (знака, транспаранта)	0,0040
Восприятие сообщения	0,0020
Восприятие показаний стрелочного индикатора	0,0070
Восприятие показаний цифрового индикатора	0,0020

Нажатие клавиши на клавиатуре	0,0050
Двойной щелчок мышью	0,0030
Выбор элемента на экране	0,0050

Другими словами, качество работы СРВ зависит от формы представления информации о текущей ситуации в системе и от доступных оператору средств воздействия на исполнительные компоненты системы.

Таким образом, при разработке пользовательского интерфейса СРВ основное внимание должно быть уделено следующим вопросам:

1. Детальному проектированию сценария диалога с целью выбора оптимальных маршрутов перемещения оператора по дереву диалога, а также предотвращения ситуаций, которые могут потребовать перезапуска системы.

2. Реализации средств динамического изменения структуры диалога в зависимости от текущей ситуации, складывающейся в системе.

3. Тщательному выбору визуальных атрибутов отображаемой информации, в том числе выбору средств привлечения внимания пользователя (оператора).

При всем при этом должно обеспечиваться свойство естественности интерфейса СРВ. Имеется в виду следующее. Во многих системах управления технологическими процессами за годы их существования была сформирована оптимальная структура средств индикации и контроля, а также соответствующая ей система условных обозначений, используемая на операторских пультах. При создании рабочих мест операторов учитывались результаты весьма глубоких эргономических исследований. Поэтому при проектировании интерфейса автоматизированных рабочих мест (АРМ) на базе ПЭВМ целесообразно сохранить основную схему визуализации процессов, протекающих в данной системе управления. Неслучайно практически все инструментальные средства, предназначенные для разработки интерфейса систем диспетчерского управления, содержат графические библиотеки, позволяющие воссоздавать на экране монитора мнемосхемы, идентичные использовавшимся на прежних пультах (рис. 7.4).

Рис. 7.4. Пример мнемосхемы технологического процесса, отображаемой на АРМ оператора

Такой подход позволяет использовать при работе оператора не только визуальные, но и другие (в первую очередь - звуковые) средства индикации и привлечения внимания.

Следуя основной концепции книги, в данном разделе мы более подробно рассмотрим аспекты проектирования визуальных компонентов пользовательского интерфейса СРВ. Чтобы еще раз подчеркнуть их влияние на качество программного продукта в целом, приведем такой факт. В шведском стандарте на пользовательский интерфейс систем управления электростанциями есть пункт, предусматривающий, что требующая срочного внимания оператора информация должна сопровождаться символом красного цвета, заметным не менее чем с двух метров от монитора. Если это требование не было выполнено, то при пропуске оператором аварийной ситуации ответственность за происшедшее автоматически возлагается на разработчика пользовательского интерфейса.

Дабы не оказаться в положении такого разработчика, при выборе визуальных атрибутов элементов интерфейса следует учитывать как требования имеющихся стандартов, так и физиологические особенности зрения оператора.

Прием визуальной информации содержит ряд элементарных процессов: обнаружение, различение, опознание и декодирование. На выполнение этих процессов основное влияние оказывают следующие характеристики зрения оператора:

- яркостные
- пространственные
- временные
- цветового восприятия.

Все они в значительной степени зависят от размеров и свойств излучения объектов, отображаемых на экране.

Яркостные характеристики

Они определяют размер зоны видения светящегося объекта, а также скорость и безошибочность обработки светящейся информации.

Зрительное восприятие светящегося объекта возможно в диапазоне яркостей $10^6 \dots 10^5$ кандел/м². Яркость светящегося объекта может быть рассчитана по формуле $B-K-0,251\ln(a)+0,79$, где K — степень ослепления (при $K=1 \dots 2$ оператор испытывает дискомфорт, а при $K=3 \dots 8$ — болевые ощущения); a — угловой размер светящегося объекта (измеряется в градусах). Яркость, превышающая $15 \cdot 10^6$, является слепящей.

Для обеспечения длительной зрительной работоспособности оператора яркость наблюдаемых на экране объектов не должна превышать 64 кд/м²; при этом перепад яркостей в поле зрения оператора должен быть не более 1:100. Наивысшая быстрота различения сложных объектов достигается при яркости $3 \cdot 10^5$ кд/м².

Необходимо также учитывать, что требуемая острота зрения при восприятии светлых объектов в 3-4 раза ниже чем, для темных; светлые объекты на темном фоне обнаруживаются легче, чем темные на светлом.

Пространственные характеристики

Данная группа характеристик влияет на обнаружение, различение и опознание объектов.

При решении практических задач необходимо учитывать следующие положения:

1. Основную информацию об объекте несет его контур; время различения и опознания контура объекта увеличивается с увеличением его сложности.
2. При различении сложных контуров безошибочность выше, чем при различении простых.
3. Решающее значение в восприятии формы объектов имеет соотношение «фигура/фон».
4. Минимальный размер объекта должен выбираться для заданных уровней контраста и яркости; уменьшение значений этих параметров требует увеличения угловых размеров объекта.
5. Для повышения вероятности различения с 0,5 до 0,98 требуется увеличение угловых размеров для простых фигур на 20...25%, а для знаков типа букв и цифр — в два раза.

6. Для различения положения фигуры относительно вертикальной или горизонтальной оси пороговая величина обнаружения должна быть увеличена в 3 раза (порог обнаружения темного объекта на светлом фоне составляет 1 угловую секунду).

При наличии на экране движущихся объектов следует учитывать ряд дополнительных факторов. Например, при перемещении точечного объекта со скоростью 0,25 градус/с его непрерывное движение воспринимается как дискретное, при скорости 0,25...4 градус/с — как непрерывное, а при скорости более 4 градус/с изображение сливается в сплошную полосу.

Полезно также помнить о том, что существует три вида кажущегося движения:

- восприятие перемещения сигнала из одного положения в другое при последовательном предъявлении двух идентичных сигналов от различных объектов;
- кажущееся изменение размеров объекта при последовательном появлении двух объектов, имеющих идентичные контуры;
- кажущееся изменение размеров объекта при изменении яркости самого объекта или фона.

Временные характеристики

Зрительное восприятие светящегося объекта формируется у человека-оператора с некоторой задержкой по отношению к началу действия зрительного раздражителя и его прекращению, что обуславливает ряд особенностей функционирования зрительного анализатора. Эти особенности проявляются как при восприятии одиночных световых сигналов, так и их последовательности. Знание временных характеристик зрения позволяет обоснованно выбирать время экспозиции сигналов для обеспечения их минимальной различимости и временных интервалов предъявления сигналов в последовательности. Основные временные характеристик и зрительного восприятия приведены в табл. 7.2.

Таблица 7.2.

Временные характеристики зрения

Характеристика	Количественное значение	Условия наблюдения
Субъективно воспринимаемая яркость при мельканиях, %	200 100 50	Частота мелькания (Гц): 8...10 16...20 24...28
Критическая частота мельканий для их раздельного восприятия, Гц	15 25 50	Яркость объекта (кд/м ²): 0,1 1 100
Быстрота обнаружения, мс	<3 <30 <7 <60	Для объектов простой конфигурации то же, в плохих условиях наблюдения. Для знакомых человеку изображений (буквы, цифры). То же, в условиях помех

Характеристики цветового восприятия

Цвета различаются тоном, светлотой и насыщенностью. Число различимых оттенков цвета по всему спектру при яркости не менее 10 кд/м² и максимальной

насыщенности равно приблизительно 150. Различение степеней насыщенности колеблется от 4 (для желтого) до 25 (для красного). При изолированном предъявлении человек точно идентифицирует не более 10-12 цветовых тонов, а в комбинации с другими цветами - не более восьми.

Изменение яркости объекта влияет на восприятие его цвета. С уменьшением яркости от 180 до 0,5 кд/м² происходит уменьшение светлоты и постепенное обесцвечивание желтого и синего цветов, а спектр становится трехцветным: красно-зелено-фиолетовым.

Восприятие цвета зависит также от угловых размеров объекта: с уменьшением размера изменяется видимая яркость и искажается цветность. Наибольшему изменению подвержены желтый и синий цвета.

Для систем реального времени основным критерием выбора цветов отображаемых на экране символов и сообщений является *острота различения*. Она максимальна для символов белого цвета и минимальна для символов, имеющих крайние цвета спектра. Хотя белый цвет наиболее прост в применении и его часто используют, наилучшим в этом отношении является желто-зеленый цвет, который по насыщенности мало отличается от белого, но имеет максимальную видность; красный, фиолетовый и синий цвета не рекомендуется использовать для отображения символов или объектов сложной конфигурации.

При согласовании цветов символов и фона следует учитывать, что восприятие символов максимально для контрастных цветов (т.е. относящимся к противоположным границам спектра). При контрастности менее 60% читаемость символов резко ухудшается. Установлены следующие допустимые комбинации цвета символа с цветом фона (в порядке убывания четкости восприятия):

- синий на белом
- черный на желтом
- зеленый на белом
- черный на белом (только четвертое место!)
- белый на синем
- зеленый на красном
- красный на желтом
- красный на белом
- оранжевый на черном
- черный на пурпурном
- оранжевый на белом
- красный на зеленом.

При одновременном поступлении двух или нескольких сигналов (сообщений) на их восприятие оператором влияют следующие факторы: избирательность внимания, абсолютная и относительная интенсивность сигналов, взаимное расположение на экране, степень синхронности сигналов, объем поступающей информации и скорость ее поступления.

Наряду с рассмотренными выше характеристиками важное значение для эффективной работы оператора имеет способ передачи смыслового содержания отображаемой на экране информации. Этот способ может базироваться на использовании одного из четырех типов знаковых систем (или их комбинации):

- буквенной
- пиктографической
- цифровой
- геометрической.

При выборе знаковой системы следует учитывать:

1. Легкость опознания и декодирования знаков.
2. Требуемую длительность безошибочной работы оператора, в том числе в условиях стресса.
3. Уровень помехоустойчивости системы.
4. Скорость запоминания и длительность сохранения алфавита знаковой системы в оперативной и долговременной памяти оператора.

В качестве интегральной характеристики знаковой системы может использоваться *коэффициент оперативности кода* [8], K , представляющий собой отношение времени опознания символа (знака) к времени его декодирования. Значения этого показателя для перечисленных выше систем приведены в табл. 7.3.

Таблица 7.3.

Значения коэффициента оперативности кода

Знаковая система	Значение Коп
Буквенная (для одного слова)	0,9
Пиктографическая (для пиктограммы)	0,8
Цифровая (для одного числа, не более 4 разрядов)	0,6
Геометрическая (для одной фигуры)	0,6

Из приведенных данных можно, в частности, сделать вывод, что в стрессовых ситуациях числа до трехразрядных включительно целесообразно представлять на экране в текстовой форме (то есть словами). Вместе с тем, основные свойства объекта или описание требуемых действий эффективнее отобразить в виде пиктограммы. Так, фразу «переслать сообщение на вышестоящий уровень» лучше заменить соответствующей пиктограммой.

Однако, как уже отмечалось в главе 4, пиктограмма пиктограмме рознь. Попробуйте, например, самостоятельно определить (или вспомнить) смысл приведенных на рис. 7.5 пиктограмм, взятых из одного весьма популярного приложения.

Рис. 7.5. Пиктограммы — «загадки»

Экспериментально доказано [8], что наиболее значимые характеристики объекта должны кодироваться (отображаться) его контуром, а внутренними деталями - вспомогательные, второстепенные. При этом система опознавательных признаков формы знака, выбранная для определенных характеристик объекта, должна применяться для всего алфавита знаковой системы.

Количественные оценки влияния геометрического контура пиктограммы на эффективность ее распознавания даны в табл. 7.4.

При разработке знаковой системы следует учитывать, что симметричные символы легче усваиваются человеком-оператором и более прочно сохраняются в кратковременной и долгосрочной памяти.

В качестве различительных признаков знаков в пределах одного алфавита не рекомендуется использовать:

- Число элементов в знаке;
- Геометрические размеры знака (по крайней мере, более двух вариантов);
- Отличие знаков по принципу «позитив-негатив» и «прямое-зеркальное отражение».

Таблица 7.4.

Влияние геометрической сложности знака на его декодирование

Показатель	Значение показателя		
	Простые знаки	Знаки средней сложности	Сложные знаки
Минимальное время экспозиции, с	0,03	0,03	0,05
Среднее время декодирования при экспозиции 0,03с, с	3,06	2,55	2,76
Вероятность правильного декодирования	0,80	0,97	0,98

И в завершение еще один фактор, упоминавшийся в предыдущих главах - количество интерактивных элементов, одновременно отображаемых на экране. Естественно, на эффективность работы с ними влияют и зрительные характеристики оператора, и качество используемой знаковой системы. Тем не менее при выборе нужного элемента сказывается еще одна характеристика оператора - сенсомоторная. В качестве примера в таблице 7.5 приведены достаточно усредненные значения безошибочности (P_b) и времени (T_b) выбора требуемого элемента в зависимости от числа имеющихся.

Таблица 7.5.

Обобщенные показатели сенсомоторной характеристики оператора

Количество интерактивных элементов на экране (в активном окне)	P_b	T_b, c
3	0,999	1,5
7	0,997	3,0
10	0,995	4,0
15	0,97	5,0
20	0,94	7,0
60	0,92	10,0

Важным фактором, влияющим на эффективность работы оператора, является поддержание всех его «подсистем» на требуемом уровне готовности в течение достаточно длительного времени. В связи с этим дополнительно к рассмотренным должны быть решены две взаимосвязанные проблемы: предотвращение как «сенсорного голода», так и чрезмерной сенсорной перегрузки оператора.

Для борьбы с перегрузкой достаточно учитывать те общие рекомендации по размещению информации на экране, которые были приведены в главе 2, а также характеристики зрения оператора, рассмотренные выше.

«Сенсорный голод» может быть обусловлен чрезмерным искусственным снижением динамичности отображения текущей ситуации на экране, а также свето- и звукоизоляцией рабочих мест. В связи с этим для предотвращения «сенсорного голода» должна быть введена определенная избыточность представленной на

экране информации по сравнению с минимально необходимой. Например, на экране могут появляться сообщения, требующие той или иной реакции оператора, но реально не влияющие на процесс управления. Другой способ борьбы с «сенсорным голодом» основан на использовании нескольких форм представления информации (т.е. на применении мультимедийных технологий); речь идет в первую очередь о дополнении визуальной информации звуковым сопровождением. И хотя круг рассматриваемых в книге вопросов умышленно ограничен визуальной компонентой пользовательского интерфейса, в данном случае из этого круга необходимо ненадолго выйти. Объясняется это тем, что *полимодальная* организация предоставляемой оператору информации позволяет решить сразу несколько проблем. Во-первых, как было сказано выше, утолить его «сенсорный голод». Во-вторых, при дублировании информации по нескольким сенсорным каналам сокращается время реакции оператора. В-третьих, такой подход позволяет увеличить объем одновременно принимаемой информации. И, наконец, привлечение дополнительных сенсорных каналов позволяет разгрузить тот, по которому информация поступает наиболее интенсивно.

Вместе с тем, применение мультимедийных технологий значительно усложняет интерфейс, что требует при его проектировании решения дополнительных задач. Основная из них - согласование информации, поступающей по разным каналам, по времени и по содержанию. При несоблюдении этого условия эффект от полимодальности окажется прямо противоположным ожидаемому. При одновременном поступлении нескольких сигналов, требующих выполнения различных действий, время реакции оператора увеличивается; практически неизбежна также информационная перегрузка оператора и, как следствие, — его повышенная утомляемость.

В системах реального времени участие оператора в процессе управления проявляется главным образом при возникновении нештатных, аварийных или критических ситуаций. Вместе с тем, именно такие ситуации вызывают у человека дискомфортное или даже стрессовое состояние. В связи с этим особое значение для СРВ имеет проблема реализации средств поддержки пользователя. Очевидно, ведущую роль здесь должна играть контекстно-зависимая помощь и помощь, определяемая заданием. При этом целесообразно предусмотреть два способа предоставления помощи: по запросу оператора и автоматически, например, по истечении некоторого допустимого времени ожидания реакции оператора на возникшую ситуацию.

С тех же позиций следует рассматривать и возможность документирования действий оператора. Практика показывает, что для полноценного анализа действий оператора требуется не только регистрировать перечень выполнявшихся команд и значения регулируемых параметров, но и формировать снимки экрана в соответствующие моменты времени. При реализации такой возможности необходимо учитывать технические характеристики средств вывода (принтеров). Если для экспресс-анализа используется черно-белая печать, то это накладывает дополнительные ограничения на выбор цветовой палитры экрана и отображаемой на нем информации.

В связи с повышенными требованиями, предъявляемыми к надежности и быстродействию систем реального времени, при их создании значительно возрастает роль этапа макетирования пользовательского интерфейса и его согласования с потенциальными пользователями.

Завершая обсуждение проблем, связанных с проектированием и реализацией пользовательского интерфейса в конкретных предметных областях, еще раз

напомним, что выбранные области относятся, на первый взгляд, к противоположным границам спектра. В связи с этим приведем рекомендации по разработке Web-узлов редактора электронного журнала Corporate Internet Strategies ПятаЛошина [10]:

1. Использовать графику только там, где она будет нести реальную смысловую нагрузку;
 2. Использовать текст вместо графики где только возможно (для ускорения загрузки страниц);
 3. Использовать фреймы, таблицы и другие элементы структурной организации содержимого;
 4. Использовать одни и те же компоненты на разных страницах для сокращения времени их загрузки;
 5. Размещать наиболее важную информацию в самых доступных местах сайта;
 6. Содержимое сайта должно быть действительно полезным для посетителя.
- Отдельная группа рекомендаций связана с использованием Интернета в качестве источника информации о биржевых котировках. В этом случае для работающих в сети брокеров счет времени идет на секунды, а Интернет выступает в роли самой настоящей системы жесткого реального времени. Как говорится, комментарии излишни...

8. СРЕДСТВА РЕАЛИЗАЦИИ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

8.1. КЛАССИФИКАЦИЯ СРЕДСТВ РАЗРАБОТКИ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

Технология построения пользовательского интерфейса и инструментальные средства, используемые для ее реализации, образуют единое целое. Очередной шаг в развитии любой из этих составляющих дает толчок к дальнейшему развитию другой. Материальной же основой существования любого пользовательского интерфейса является перечень устройств ввода/вывода, доступных конечному пользователю. В те далекие времена, когда единственным средством ввода информации в ЭВМ служило устройство чтения с перфокарт, а средством вывода - его брат-близнец для вывода на перфокарты, ни один объектно-ориентированный язык программирования не помог бы представить пользователю результаты иначе, как в виде дырочек на перфокарте. Другими словами, тогда проблемы реализации пользовательского интерфейса для программистов просто не существовало («нет интерфейса - нет проблем»).

С появлением алфавитно-цифровых (символьных) устройств ввода/вывода, таких как алфавитно-цифровые дисплеи (АЦД) и алфавитно-цифровые печатающие устройства (АЦПУ), в языки программирования были включены соответствующие операторы (либо библиотечные функции), предназначенные для реализации взаимодействия пользователя с этими устройствами. На данном этапе технология программирования компонентов, относящихся к пользовательскому интерфейсу, практически не отличалась от программирования остальных функций приложения, да и сами эти компоненты были как бы «размазаны» по всей программе. Впоследствии такая технология получила наименование «внутреннее управление интерфейсом». Опирается она в основном на процедурные языки программирования, как высокого уровня (Фортран, Паскаль, Си), так и машинно-ориентированные (типа ассемблера). Очевидно, что при таком подходе разработка

интерфейса как самостоятельной компоненты программной системы практически невозможна. Хотя упомянутые выше процедурно-ориентированные языки не зря называются универсальными и позволяют реализовать любой тип интерфейса (в том числе и GUI), трудозатраты на практическую реализацию такой затеи могут оказаться не по силам подавляющему большинству разработчиков. Кроме того, необходимость «ручного» описания огромного числа атрибутов интерактивных компонентов приложения делает невозможной стандартизацию этих компонентов. В определенной степени упростить решение указанной задачи позволило появление проблемно-ориентированных языков программирования, таких как языки моделирования (SIMULA, GPSS, SOL) и языки управления базами данных (Clipper, dBASE, PAL). Особую группу процедурных языков образуют так называемые языки диалогового взаимодействия (или командные языки), созданные специально для облегчения работы пользователей в интерактивном режиме. Основу синтаксиса этих языков составляют макрокоманды (или макро-операторы.), реализующие определенную последовательность действий по вводу/выводу данных. Например, один из языков диалогового взаимодействия - ДИ-ФОЛ - содержит такие операторы:

DISPLAY - вывести информацию на экран;

UPR - создать незащищенное поле ввода;

NUM - создать цифровое поле;

BRO — создать неотображаемое поле;

BR1 — создать поле нормальной яркости;

BR2 — создать поле, отображаемое с повышенной яркостью.

Следует, видимо, вспомнить, что весьма популярный ныне язык BASIC, как и не менее популярный в свое время язык APL, также создавался изначально как диалоговый язык (BASIC - Beginners All-purpose Symbolic Instruction Code). Судя по всему, именно этим объясняется решение фирмы Microsoft использовать BASIC в качестве встроенного языка приложений. Но об этом чуть позже.

Из всех перечисленных групп языков наибольшее влияние на развитие технологии проектирования и разработки пользовательского интерфейса оказали языки управления базами данных. Еще до начала триумфального шествия «по странам и континентам» графической оболочки Windows 3.* и появления термина Data-Centered Design языки СУБД обеспечивали отдельное описание данных и средств работы с ними. В значительной степени это объясняется самой сущностью использования БД: для различных заданий или для различных пользователей требуется обеспечить разное представление одних и тех же данных. Например, в состав ранних версий СУБД Paradox уже входили такие компоненты:

Ask — генерация форм запросов;

Report — разработка спецификаций отчетов;

Create - создание структуры новой таблицы;

Forms — разработка спецификаций экранных форм;

Image - установка пользовательских характеристик представления таблицы на экране (в виде формы или графика).

Более того, в составе СУБД Paradox имеется так называемый генератор приложения (Personal Programmer), обеспечивающий создание приложений для работы с БД и способный выполнять свои функции даже при отсутствии на ПЭВМ ядра СУБД. При этом как перечисленные выше компоненты, так и генератор приложений ориентированы в первую очередь на непрограммирующих пользователей. С помощью системы меню и функциональных клавиш генератора приложений

пользователи могли создавать собственную конфигурацию интерактивных элементов приложения, в том числе выпадающие и иерархические меню, окна, а также средства помощи (окна со справочной информацией). Аналогичные возможности имелись практически во всех развитых СУБД.

Наряду с другими достижениями в области технологий программирования, появление объектно-ориентированных баз данных способствовало внедрению объектно-ориентированного подхода в практику создания пользовательских интерфейсов [5]. Технология объектно-ориентированного программирования позволила еще более явно отделить друг от друга компоненты приложения, реализующие его функциональное предназначение, и компоненты, относящиеся к пользовательскому интерфейсу.

Чрезвычайно большое влияние на все последующее развитие интерактивных систем оказала растровая графика. Ее применение в качестве основы инструментов визуального программирования привело к появлению принципиально нового типа пользовательского интерфейса — графического (основные концепции GUI были рассмотрены в главе 2).

Средства визуальной разработки, обеспечивающие реализацию объектно-ориентированного программирования, позволяют создавать макет пользовательского интерфейса, используя технологию WYSIWYG (What You See Is What You Get - «что вы видите, то и получите», то есть результат выглядит так же, как и прототип во время разработки). Средства визуальной разработки были созданы практически для всех популярных языков программирования, а также для вновь появившихся (например, для Java). Все эти инструменты обладают двумя основными достоинствами: во-первых, существенно повышают производительность труда программиста, и, во вторых, обеспечивают стандартизацию пользовательского интерфейса за счет использования однотипных базовых элементов. В результате, глядя на готовое приложение, практически невозможно определить, на каком языке и с помощью какого инструмента оно было создано. Например, на рис. 8.1 представлены два первичных окна, одно из которых было получено с помощью Visual C++, а второе - с помощью Visual Smalltalk.

Наиболее удачно реализованные инструменты визуального программирования позволяют не только формировать облик отдельных окон и диалоговых панелей, но и представлять в наглядной форме взаимосвязь между элементами пользовательского интерфейса (рис. 8.2); это обеспечивает решение многих проблем проектирования интерфейса, рассмотренных в главе 2.

Рис. 8.1. Первичные окна, созданные с помощью Visual C++ и Visual Smalltalk

Рис. 8.2. Макет пользовательского интерфейса, созданный в среде Visual Smalltalk

Аналогичными возможностями обладают сегодня и многие инструментальные средства, созданные на базе проблемно-ориентированных языков. Например, на рис. 8.3 показан внешний вид окна редактора GUI Г, входящего в состав пакета MATLAB, а рядом - макет окна, созданного с помощью этого редактора.

Рис. 8.3. Окно редактора GUI пакета MATLAB и созданный с его помощью макет интерфейса

Как и до появления средств визуального программирования, особое место среди других проблемно-ориентированных систем разработки занимают СУБД. Применение в них технологии WYSIWYG позволило им практически сравняться по мощности и эффективности с универсальными инструментами разработки GUI-приложений. И даже более того, наличие в СУБД средств визуального представле-

ния инфологической модели данных позволяет во многих случаях создавать более корректную модель пользовательского интерфейса по сравнению с универсальными инструментами. На рис. 8.4. приведен пример инфологической модели данных и экранная форма, созданные в СУБД Access.

В силу того, что интерфейс систем реального времени имеет целый ряд существенных особенностей (основные из которых были рассмотрены в предыдущей главе), для его построения используются, как правило, специализированные инструментальные средства. Они сформировались в результате слияния SCADA-систем (Supervisory Control And Data Acquisition system - систем сбора данных и оперативного диспетчерского управления) и средств визуального программирования «общего назначения» на

Рис. 8.4. Визуальное моделирование интерфейса приложения в СУБД Access

базе одного из универсальных языков (чаще всего - Visual Basic). Такой симбиоз получил название HMI/SCADA-систем (или MMI/SCADA), где аббревиатуры HMI и MMI соответствуют термину «человеко-машинный интерфейс» (Human Machine Interface или Man Machine Interface). В настоящее время такие инструментальные средства существуют практически для всех платформ, на базе которых разрабатываются системы реального времени. Интерфейс создаваемых с их помощью приложений зависит в основном от специфики конкретной области применения и в значительно меньшей степени - от используемой операционной системы и ее графической оболочки. Например, интерфейс АРМ оператора, который был приведен на рис. 7.2, создан в графической среде Photon microGUI операционной системы QNX, а интерфейс АРМ, показанный на рис. 8.5 - в среде Windows.

Упомянутый выше язык Visual Basic (точнее, одна из его спецификаций — Visual Basic Application — VBA) оказал большое влияние на технологию создания приложений, настраиваемых пользователем. Продуманность и логическая завершенность решений, предложенных Microsoft, привела к тому, что VBA прочно занял свою собственную «нишу» среди инструментальных средств формирования пользовательского интерфейса приложений. Пожалуй, в этом отношении он является даже уникальным инструментом, и не случайно многие фирмы-производители ПО лицензировали VBA у Microsoft с целью использования в качестве встроенного языка приложений.

Рис. 8.5. Интерфейс АРМ, созданный с помощью HMI/SCADA-системы

Несмотря на суммарные потенциальные возможности систем визуального программирования, они в большинстве своем обладают одним существенным недостатком - в них (за редким исключением) изначально не предусмотрена поддержка проектирования разработки и сопровождения создаваемых приложений как единого технологического процесса. Именно это обстоятельство зачастую негативно влияет как на уровень программного продукта в целом, так и на качество его пользовательского интерфейса. Осознание этого факта привело к тому, что разработчики инструментов стали дополнять их относительно самостоятельными компонентами, поддерживающими отдельные этапы жизненного цикла программных продуктов. Например, практически все современные инструменты разработки имеют в своем составе компоненту, предназначенную для управления версиями программного продукта (в пакете Visual Studio фирмы Microsoft такая компонента называется SurfaceSafe; аналогичные компоненты имеются и для инструментов разработки на Java). Появились также и специализированные инструменты тестирования GUI-

приложений. Одним из наиболее мощных из них на сегодняшний день можно считать продукт Rational Performance Suite фирмы Rational Rose. Данное средство обеспечивает автоматическую генерацию тестов, имитирующих работу пользователя, а также регистрацию и анализ результатов тестирования приложения, прежде всего с точки зрения качества пользовательского интерфейса.

Тем не менее, в инструментах визуального программирования поддержку получают в основном этапы жизненного цикла, относящиеся к разработке и реализации приложения, и в значительно меньшей степени - относящиеся к этапам проектирования.

Указанного недостатка лишены так называемые CASE-системы (CASE - это Computer Aided Software Engineering - компьютерное проектирование программного обеспечения). Понятие CASE является весьма широким и охватывает как собственно технологию, так и средства ее реализации. Обязательным атрибутом CASE-системы является возможность автоматической (или по крайней мере автоматизированной) генерации кода программы на основе ее спецификации. Существенной особенностью CASE-систем является также поддержка практически всех основных этапов жизненного цикла создаваемого приложения, в том числе:

- Стратегическое планирование (описание целей, факторов, ресурсов; моделирование стратегии; формирование структуры плана и политики фирмы-разработчика);
- Описание предметной области (описание объектов предметной области и отношений между ними; интеграция различных моделей предметной области);
- Анализ возможностей реализации (анализ существующих проектов);
- Определение требований (моделирование потоков данных; создание и анализ прототипов; контроль полноты и согласованности требований);
- Системное проектирование (декомпозиция и сборка проекта, имитационное моделирование создаваемого приложения);
- Программирование (генерация кода и анализ его метрических характеристик);
- Тестирование (автоматическая генерация контрольных примеров, регистрация и анализ результатов тестирования);
- Документирование (создание и сопровождение библиотеки спецификаций);
- Сопровождение и управление проектом.

Как следует из перечисленных особенностей CASE-систем, их применение способствует проектированию и реализации пользовательского интерфейса, обладающего требуемыми свойствами. Более того, некоторые из таких систем имеют в своем составе компоненты, предназначенные специально для разработки пользовательского интерфейса создаваемого приложения. Например, продукт CASE/4/0 фирмы MicroTOOL GmbH содержит так называемый «дизайнер диалогов», обеспечивающий создание и моделирование пользовательского интерфейса. Вместе с тем, сами по себе CASE-системы достаточно сложны в освоении и использовании, поэтому эффективность их применения прямо пропорциональна сложности создаваемого продукта.

Рассмотренные выше этапы эволюции инструментов и технологий разработки приложений могут быть положены в основу схемы классификации существующих средств создания пользовательского интерфейса (рис. 8.6). При всей условности такой (да и любой другой) классификации она дает достаточно полное представление о применяемых в настоящее время подходах к реализации интерактивных приложений.

Рис. 8.6. Классификация средств разработки пользовательского интерфейса

Приведенная на рис. 8.6 схема требует небольшого пояснения. На книжных прилавках в достаточном количестве имеются издания по всем отраженным в ней инструментам, за исключением средств разработки Help-систем и инструментов создания Web-материалов. Поэтому в двух следующих разделах основное внимание уделено именно этим категориям программных продуктов.

Существенное возрастание количества и многообразия интерактивных приложений, а также расширение области их применения обусловили наличие двух тенденций:

во-первых, все существующие инструменты создания приложений стали оцениваться (классифицироваться) помимо других критериев еще и с точки зрения их пригодности для создания пользовательского интерфейса определенного уровня;

во-вторых, появились инструментальные средства, специально предназначенные для проектирования и реализации пользовательского интерфейса.

Согласно [9], инструментальные средства создания пользовательского интерфейса могут быть отнесены к одному из следующих классов:

- Системы управления пользовательским интерфейсом (User Interface Management System — UIMS);
- Инструментальные средства проектирования и разработки интерфейса (Interface Builder — IB);
- Инструментальные средства разработки интерфейса (Tools&Toolkit — T&T);
- Средства прототипирования интерфейса (Prototyping Tools — PT).

Система управления пользовательским интерфейсом (UIMS) - это интегрированный набор средств, помогающих программисту в создании и управлении различными интерфейсами пользователя. Основной концепцией UIMS является идея разделения интерфейса и прикладной программы (точнее, ее функционального наполнения).

Как правило, UIMS состоит из двух частей: одна обеспечивает разработку интерфейса, а вторая - управление пользовательским интерфейсом в процессе его работы с приложением. Многие UIMS имеют собственный язык определения интерфейса для представления требуемого диалога и генератор, который автоматически создает необходимый код из исходного описания на этом языке. В идеале UIMS должна, с одной стороны, позволять создавать различные интерфейсы для работы с одним и тем же приложением, а с другой - поддерживать один и тот же интерфейс для различных приложений. Список наиболее распространенных UIMS, доступных через Интернет, приведен в приложении 1. Из рассмотренных выше инструментальных средств к данному классу могут быть отнесены, некоторые CASE-средства и наиболее развитые из систем типа HMI/SCADA.

Класс **инструментальных средств проектирования и разработки интерфейса** (Interface Builder) образуют средства, которые обеспечивают создание интерфейса определенного (стандартизованного) типа для различных приложений, функционирующих в соответствующей операционной среде. Примерами таких средств могут служить Visual C++ и Delphi для MS Windows, Tk/TCL для XWindows или Photon Application Builder (Phab), обеспечивающий создание GUI-приложений в графической среде Photon microGUI операционной системы QNX. Некоторые представители данного класса поддерживают только этап проектирования пользовательского интерфейса и ориентированы на совместное использование с одним из инструментов визуального программирования.

Инструментальные средства разработки интерфейса (Tools&Toolkit) близки по своим характеристикам представителям предыдущего класса, но либо имеют более ограниченные функциональные возможности, либо представляют собой набор (библиотеку) элементов, на основе которых могут быть реализованы различные варианты GUI.

Средства прототипирования, как следует из их названия, предназначены для построения макета (прототипа) пользовательского интерфейса и для сравнительной оценки альтернативных вариантов.

Взаимосвязь двух аспектов классификации инструментов создания пользовательского интерфейса показана на рис. 8.7.

Рис. 8.7. Взаимосвязь двух аспектов классификации инструментов создания пользовательского интерфейса

Список характерных представителей перечисленных классов (доступных в Интернете) приведен в Приложении.

8.2. ИНСТРУМЕНТЫ РЕАЛИЗАЦИИ СРЕДСТВ ПОДДЕРЖКИ ПОЛЬЗОВАТЕЛЯ

Инструменты реализации компонентов приложения, обеспечивающих поддержку пользователя, занимают особое место среди инструментальных средств построения пользовательского интерфейса.

Наиболее распространенные виды такой поддержки были рассмотрены в главе 6. Однако лишь немногие из них могут быть реализованы теми же средствами, которые используются для создания самого приложения (либо для этого потребуются слишком большие затраты сил и времени).

Проблемно-ориентированная помощь и Справочник, рассмотренные в главе 6, появляются на экране благодаря компоненте WinHelp (или WinHelp32), входящей в состав ОС Windows. Так называемые Help-файлы, открываемые с ее помощью, могут быть созданы как «вручную», так и с помощью специализированных средств. В обоих случаях технология формирования Help-файла практически одна и та же и состоит в выполнении следующих основных шагов:

- Создание разделов (страниц) помощи в одном из текстовых редакторов (например, MS Word) с использованием специальных символов разметки;
- Преобразование полученного текстового документа в формат RTF;
- Создание проекта Help-файла (.hprj);
- Компиляция файлов .rtfu .hprj в результирующий Help-файл (.hip). Наиболее трудоемким этапом является формирование структуры Help-файла (разбиение на разделы, описание связи между разделами, включение рисунков и интерактивных элементов и т.д.). Именно его выполнение позволяют автоматизировать упомянутые выше специализированные средства. Одним из них является продукт фирмы Microsoft, который называется Microsoft Help Workshop. Данное приложение реализовано на основе MDI и позволяет одновременно получать информацию о различных аспектах создания Help-файла (например, просматривать содержимое файла проекта и результаты его компиляции, как показано на рис. 8.8).

Рис. 8.8. Общий вид приложения Microsoft Help Workshop 296

Перечень разделов главного меню приложения зависит от того, какое дочернее окно активно в данный момент. При первоначальном открытии приложения пользователю доступны разделы File, View, Test, Tools и Help. Для ознакомления с технологией создания Help-файлов в MS Help Workshop целесообразно

воспользоваться имеющейся в нем проблемно-ориентированной помощью, доступной из раздела Help (рис. 8.9).

Рис. 8.9. Вызов проблемно-ориентированной помощи

Пользуясь советами, отображаемыми в окнах разделов задания, можно без каких-либо затруднений создать несложную справочную систему в формате .hlp. MS Help Workshop обеспечивает также формирование структурированной информации для броузера разделов (отображаемой на вкладках *Содержание*, *Предметный указатель* и *Поиск*).

Как было отмечено в главе 6, в последнее время все большую популярность среди разработчиков приложений завоевывает новый формат Help-файлов (.chm). Для поддержки этого формата Microsoft создала соответствующий инструмент -HTML Help Workshop, который входит в состав Visual Studio 6, но может использоваться и как самостоятельное приложение.

HTML Help Workshop, как и его предшественник, реализован в виде MDI-приложения и на первый взгляд мало чем от него отличается (рис. 8.10).

Рис. 8.10. Внешний вид основного (родительского) окна HTML Help Workshop

Тем не менее, отличия есть, и они весьма существенные. Основное заключается в том, что исходный файл для создания справочной системы должен быть подготовлен на языке HTML. Благодаря этому HTML Help Workshop может использоваться не только как средство для создания справочных систем, но и в качестве полноценного редактора Web-страниц. В частности, с его помощью в Help-файл (который теперь корректнее называть HTML-файлом) могут быть помещены ActivX-элементы или Java-апплеты. Для облегчения работы пользователя при создании справочной системы в составе HTML Help Workshop имеется соответствующий Мастер, который позволяет также преобразовать в новый формат имеющиеся Help-файлы, созданные «в старом стиле».

Вообще же порядок работы с HTML Help Workshop различается в зависимости от того, какие цели вы ставите перед собой, и какие исходные данные у вас имеются. Выбрав в разделе File команду New, вы можете продолжить работу по одному следующих направлений (рис. 8.11):

Рис. 8.11. Выбор типа создаваемого объекта

- создать файл проекта (Project), аналогичный по структуре и назначению «старым» файлам .hlp (теперь такие файлы имеют расширение -hhp);
- создать текстовый файл (Text) в формате блокнота Notepad;
- создать HTML-файл;
- Подготовить информацию для броузера разделов: - для вкладки Содержание (Table of Contents) и Предметного указателя (Index).

При выборе одного из вариантов открывается соответствующее дочернее окно (как правило, с собственной дополнительной панелью инструментов). Например, на рис. 8.12 показан вид HTML Help Workshop при создании (или редактировании) HTML-файла.

Необходимо отметить, что в составе справочной системы HTML Help Workshop имеется специальный раздел, посвященный описанию синтаксиса HTML.

Рис. 8.12. Вид HTML Help Workshop при создании (редактировании) HTML-файла.

Для подготовки иллюстрации к электронным справочникам, создаваемым средствами HTML Help Workshop, он содержит встроенный графический редактор (который, впрочем, может использоваться и автономно) HTML Help lineage Editor. Его запуск производится с помощью одноименной команды из раздела Tools главного меню HTML Help Workshop.

Графический редактор также реализован в виде MDI-приложения и обеспечивает выполнение следующих основных функции (рис. 8.13):

- Создание снимков экрана;
- Просмотр, редактирование и конвертацию графических файлов;
- Просмотр изображений, помещаемых в HTML-файл, с учетом их размещения на странице.

При создании снимков экрана HTML Help Image Editor позволяет произвольно выбирать размер и расположение «фотографируемого» участка. Благодаря наличию трех режимов работы — на основе клавиатурного доступа, с помощью мыши и с управлением по времени - он обеспечивает снимок даже тех элементов, которые исчезают с экрана при нажатии клавиши на клавиатуре или кнопки мыши. Интересной особенностью редактора является то, что на время выполнения снимка экрана сам он автоматически «прячется», не оставляя от себя даже кнопки входа на Панели задач.

Еще одно достоинство HTML Help Workshop - это облегченная процедура создания всплывающих окон контекстно-зависимой помощи (которые, напомним,

Рис. 8.13. Общий вид встроенного графического редактора HTML Help Image Editor появляются на экране при выполнении команды Что это?). Вся процедура состоит в установке требуемых значений параметров окна в панели свойств (рис. 8.14).

Рис. 8.14. Панель свойств для установки атрибутов окна контекстно-зависимой помощи
Для создания Help-файлов может быть также использована программа под названием Help&Manual, которая по своим функциональным возможностям близка MS HTML

Help Workshop. Тем не менее, технология работы с этой программой имеет определенные особенности. Первая из них заключается в том, что вся информация о создаваемой справочной системе хранится в одном файле проекта (.hin2). По этой причине Help&Manual представляет собой однооконное приложение, главное окно которого разделено на несколько подокон (рис. 8.15); окно встроенного текстового редактора (Help Text) и окно свойств создаваемого раздела (Topic Options) реализованы как «страницы» Рабочей книги.

Рис. 8.15. Главное окно Help&Manual

После создания разделов справочной системы она может быть скомпилирована в одном из следующих форматов:

- Справка в формате WinHelp (для Windows 3.* или Windows 95);
- Справка в формате HTMLHelp (т.е. в «новом стиле»);
- В виде RTF-файла;
- В виде HTML-документа.

После выбора одного из форматов в соответствующем вторичном окне могут быть произведены дополнительные настройки параметров работы компилятора (рис. 8.16).

Подобно HTML Help Workshop, в Help&Manual имеются средства формирования снимков экрана, однако их возможности весьма ограничены. Вместе с тем, Help&Manual предоставляет пользователям богатый и достаточно удобный арсенал

инструментов для внедрения в создаваемую справочную систему различных графических объектов, в том числе видеоклипов.

Важным достоинством Help&Manual является то, что он поддерживает работу с русским языком, причем как при формировании структуры справочника (выбор заголовков разделов и т.п.), так и при генерации индексного файла.

Еще один инструмент, предназначенный для реализации средств поддержки пользователя - это пакет RoboHELP Office. На наш взгляд, по своим функциональным возможностям и технологичности использования он превосходит все аналогичные продукты, названные выше.

Рис. 8.16. Окно выбора формата и установки дополнительных параметров компиляции файла проекта

RoboHELP поддерживает разработку HELP-систем не только для Windows, но и для других платформ. Подробное описание этого пакета заняло бы не один десяток страниц. Поэтому мы ограничимся описанием одной из наиболее интересных его компонент - What's This? Help Composer, предназначенной для создания окон контекстно-зависимой помощи, вызываемых по команде *What's This?* (*Что это?*). Данная компонента может использоваться как в составе RoboHELP, так и самостоятельно. Особенность этой программы состоит в том, что она позволяет разрабатывать контекстно-зависимую помощь для любых исполняемых файлов (.exe), связанных с ними файлов .dll, а также для файлов проектов на Visual Basic (.VBP) и OCX-компонентов (.ocx).

Поясним технологию применения What's This? Help Composer на небольшом примере.

На рис. 8.17. показано окно утилиты PrcView, предназначенной для сбора и отображения информации о запущенных процессах, и одно из ее вторичных окон. В исходном варианте контекстная подсказка для элементов данного окна не предусмотрена.

Чтобы создать контекстную подсказку, необходимо указать имя исполняемого файла и маршрут доступа к нему. После этого What's This? Help Composer сформирует проект файла помощи и отобразит дерево диалоговых панелей утилиты в подокне Dialog Boxes;

выбранная в нем панель отображается в соседнем подокне. Элемент, для которого будет создаваться всплывающая подсказка, указывается щелчком мыши, а текст подсказки вводится в расположенном выше текстовом поле Help Text (рис. 8.18).

Рис. 8.17. Первичное и вторичное окна утилиты PrcView

Рис. 8.18. Окно What's This? Help Composer после создания проекта файла справки

После этого остается только выполнить компиляцию файла проекта и оценить результат работы (рис. 8.19).

Рис. 8.19. Контекстная подсказка, созданная с помощью What's This? Help Composer

8.3. СРЕДСТВА РАЗРАБОТКИ WEB-ДОКУМЕНТОВ

Практически для всех областей применения программных продуктов характерна одна и та же закономерность: технологии создания продуктов, принципы их внешнего оформления и концепции взаимодействия с пользователем развиваются одновременно. Не является исключением в этом отношении и Интернет. Да это и понятно, ведь Интернет - это «пользовательский интерфейс в квадрате», поскольку практически все его посетители являются потенциальными создателями Web-материалов, на форму подачи которых оказывают огромное влияние уровень профессиональной подготовки, возраст, культурные и религиозные взгляды их авторов. Сюда же следует добавить запас свободного и «интернетовского» времени, которым располагает автор. Примерно те же факторы влияют и на выбор используемых инструментов подготовки Web-материалов к публикации.

Существует достаточно распространенное мнение, что Web-узлы прошли в своем развитии три стадии и, таким образом, наиболее современные из них относятся к третьему поколению. Такая градация напрямую связана с эволюцией тех инструментов, с помощью которых создавались представители каждого поколения.

Для Web-узлов первого поколения была характерна однотипная структура - обычный сайт состоял из одной линейной страницы, представлявшей собой последовательность текста и «картинок». При этом графические элементы создавались с помощью обычных графических редакторов, ориентированных на жесткие требования издательского дела; вследствие этого изображения получались очень высокого качества, но были весьма требовательны к имеющимся вычислительным ресурсам. Странички писали на «чистом» HTML с использованием простейших редакторов, так как текстовые процессоры с их внутренними форматами документов не годились для Интернета, главными требованиями которого были и остаются компактность и переносимость на другие платформы. Вот когда был пик популярности редактора Notepad (Блокнот), входящего в набор стандартных приложений MS Windows. Некоторые «спецы» и сейчас нет-нет, да и заявят, что профессионалы должны работать только в нем. Такое пуританство, конечно, может вызвать уважение (как храм, построенный без единого гвоздя, да еще и одним только топором), но с точки зрения производительности труда и доступности для широких масс не выдерживает никакой критики.

Страницы узлов второго поколения уже содержали интерактивные элементы, обеспечивавшие более активное участие пользователя в формировании облика просматриваемого Web-документа. Простейшими из таких элементов были ссылки с контактными почтовыми адресами, обеспечивавшими вызов программы электронной почты. Чуть позже появились страницы, которые генерировались сценариями, выполнявшимися по запросам пользователей. Такие изменения стали возможны благодаря расширениям HTML и его совместного использования с другими технологиями (PERL, CGI и т.п.). Тем не менее основными инструментами создания страниц по-прежнему оставались текстовые редакторы. В результате сайты второго поколения были так же трудно управляемы, как и их

предшественники, и так же ненадежны. Заслугой этого поколения узлов явилось то, что они продемонстрировали практически безграничные возможности языков разметки — HML и XML (extensible Markup Language) - в формировании облика Web-страниц. Нужны были инструменты, которые позволили бы свести к минимуму ручной труд и тем самым превратить Web-дизайн в разновидность народного творчества.

Итак, возможность (Интернет) породила спрос (желание опубликоваться), а спрос вызвал предложение (средства разработки web-материалов). И они не заставили себя долго ждать, появившись в таком количестве и разнообразии, что без их классификации просто не обойтись. Но прежде необходимо определить те признаки, которые могут быть положены в основу классификации существующих средств разработки. Основные из них, на наш взгляд, следующие.

1. Требуемый уровень конечного продукта, то есть что должно стать результатом применения соответствующего инструмента (элементы страниц, отдельные страницы, сайты, серверы).

2. Степень автоматизации процесса разработки и использования технологии WYSIWYG.

3. Диапазон поддерживаемых Интернет-технологий (имеются в виду технологии представления информации и обмена ею между пользователем и Web-ресурсом).

Первый вариант классификации подразумевает разделение средств разработки по признаку целевой направленности. Основными группами в этой классификации являются:

- Узко специализированные программы (утилиты), ориентированные на реализацию отдельных элементов web-документов. Предназначены для генерации специфичных частей страниц (списков, таблиц, фреймов, форм и т.п.) или механизмов (счетчиков, гостевых книг). Многие утилиты обеспечивают реализацию интерфейсных элементов, которые затем легко встраиваются в HTML-страницы. Часто так реализуют кнопки, сенсорные карты, анимационную графику. Достаточно большая группа утилит предназначена для конвертирования файлов различных типов в HTML-текст.

- Программы (как правило, простые текстовые редакторы или «программистские» редакторы, не дополняющие вводимый текст элементами форматирования), предназначенные для создания отдельных страниц или небольших узлов на языке HTML.

- Развитые HTML-редакторы, обеспечивающие поддержку всего процесса разработки - от проектирования до размещения готового сайта на сервере. Во многих случаях такие средства разработки поддерживают возможность сопровождения сайта, предоставляют некоторые инструменты для администрирования сервера.

По уровню наглядности средства разработки можно разделить на три категории:

- не обеспечивающие никаких средств визуализации результатов разработки и требующие просмотра разрабатываемых страниц во внешнем браузере (Notepad);

- обеспечивающие автоматическое форматирование и «подсветку» синтаксиса HTML-текста, а также предоставляющие возможность просмотра результата разработки (в ходе ее выполнения) собственными средствами (HotDog, Arachnofilia, HomeSite);

- визуальные редакторы, обеспечивающие работу в режиме WYSIWYG, которые не требуют просмотра во внешнем браузере, а иногда даже не предусматривают работу с «чистым» HTML-текстом (Netscape Composer, MS Frontpage, Macromedia Dreamweaver).

Широта охвата Интернет-технологий — характеристика достаточно условная. Самые сложные и продвинутое технологии теоретически могут быть реализованы и в Notepad. Благо, в основе даже такой экзотики, как виртуальная реальность, лежит все тот же текст (правда, на специальном языке VRML - Virtual Reality Modeling Language). Все определяется трудоемкостью реализации этих технологий. Например, трудно представить, что, работая в простом редакторе, кто-то вручную станет считать пиксели картинки для создания сенсорной карты. Скорее, автор просто откажется от использования подобной возможности на своей странице. Если же ему удастся найти инструмент, поддерживающий создание сенсорных карт, и при этом обеспечивающий сочетание простоты реализации с высоким качеством формируемого изображения, то вопрос о включении в страницу сенсорной карты отпадет сам собой. Автору и посетителям его сайта останется лишь восхищаться наглядностью хороших продуманных изображений и удобством работы (естественно, при условии соблюдения рекомендаций, приведенных в этой книге).

Примером такого инструмента может служить программа CuteMap фирмы GlobalSCAPE. Общий вид основного окна приложения показан на рис. 8.20.

Рис. 8.20. Основное окно CuteMap — инструмента для создания сенсорных карт

CuteMap имеет комбинированный пользовательский интерфейс, представляющий собой сочетание MDI (для одновременной работы с несколькими картами) и Рабочей книги, позволяющей легко переключаться между изображениями сенсорной карты, переходя с одной страницы на другую.

CuteMap предоставляет разработчику следующие основные возможности

- Визуальное редактирование сенсорной карты с одновременным отображением вносимых изменений в специальном окне редактора;
- Сохранение созданной сенсорной карты (или ее копирование) в существующем HTML-документе;
- Применение технологии drag-and-drop, позволяющей создавать сенсорные карты с помощью разнообразных панелей инструментов, без использования ручного редактирования HTML-кода
- Выбор браузера для оперативного просмотра результатов работы.
- Раздельная установка цветов для выбранных и невыбранных зон при одновременной работе с несколькими сенсорными картами.

Итак, практически о любой из технологий можно сказать, поддерживается она или нет тем или иным средством разработки. Часто именно это является одним из основных критериев выбора инструмента. С другой стороны, не следует забывать о качестве и полноте реализации тех или иных возможностей конкретным средством разработки. К сожалению, не редки ситуации, когда заявленная поддержка большого количества эффективных технологий выливается в крайне низкую надежность и громоздкость инструмента, наличие скрытых ограничений, невысокое качество конечного продукта. Интернет развивается слишком быстро, что пока не позволяет довести до совершенства ни один из известных нам инструментов. Очевиден вывод, что лишь одним средством, даже универсальным (т.е. ориентированным на создание сложных Web-документов), обойтись не удастся. Каждый из Web-дизайнеров постепенно формирует собственный набор инструментов, оттачивает мастерство владения ими и на практике совершенствует опыт создания качественных страниц. Приведенна ниже (рис. 8.21)

схема может служить своеобразным компасом в постоянно расширяющемся мире инструментов создания Web-материалов.

К сожалению, формат книжной страницы не позволил отобразить здесь же местоположение конкретных представителей этого мира, в связи с чем придется ограничиться словесной характеристикой хотя бы двух из них.

Рис. 8.21. Классификация инструментов разработки Web-приложений

Начнем с достаточно простых и дешевых (зачастую бесплатных) HTML-редакторов. На сайтах и в файловых архивах бесплатного и условно бесплатного программного обеспечения можно найти довольно обширные коллекции подобных редакторов. Одним из наиболее популярных продуктов данной категории является Arachnophilia 3.6. Программа поистине замечательная, можно сказать шедевр, особенно учитывая, что ее авторство принадлежит одному человеку (Paul Lutus), а платой за пользование этим редактором он считает хорошие поступки, которые мы должны совершать ежедневно. Но это лирика, а редактор действительно заслуживает добрых слов. С 1996 года по 1998 он впитывал новые возможности, совершенствовался, но сохранил достаточно скромные размеры и высокую производительность. На рис. 8.22 мы намеренно представили все доступные пользователю панели инструментов, чтобы показать, сколько возможностей языка HTML поддерживает этот редактор.

Рис. 8.22. Окно HTML-редактора Arachnophilia 3.6

К слову сказать, пользовательский интерфейс этой программы может служить хорошим примером согласованности и продуманности, а справочная система содержит не только информацию о программе, но и некоторые сведения об Интернете для начинающих пользователей. Программу полезно использовать при изучении HTML. Синтаксис выделяется цветом, хорошо реализовано автоматическое форматирование исходного текста описания страницы. Arachnophilia объединила в себе множество функций, обычно реализуемых вспомогательными утилитами (генерация таблиц, форм, конвертирование RTF-файлов и многое другое). Программа способна вызывать до шести разных браузеров для просмотра результатов работы, причем настройка выполняется очень просто. Интересно, что автор предлагает использовать свой редактор в качестве вспомогательного для MS Frontpage и утверждает, что этим будут устранены некоторые слабые стороны упомянутого колосса!. Мы ограничимся лишь небольшим резюме: для разработки отдельных страниц и небольших сайтов, сроки сдачи которых не лимитированы и хочется глубже разобраться в языке HTML (а средств на приобретение мощных инструментов нет), Arachnophilia-лучший выбор.

Следующий инструмент - это HotDog, редактор очень популярный в среде web-дизайнеров «первой волны», которые начинали с Notepad и привыкли работать с

HTML-кодом. HotDog не мешает работать в привычном им стиле, но существенно повышает производительность труда за счет синхронного показа «порождаемой» страницы и множества сервисных возможностей. Работа в этом редакторе сочетает как ручной ввод HTML-тегов с клавиатуры, так и обращение к многочисленным генераторам типовых фрагментов HTML-кода. Во втором случае нажатие пиктограмм, обозначающих теги и группы тегов на инструментальных панелях, приводит к появлению в текущей позиции исходного текста страницы соответствующего фрагмента HTML-кода. По мере ввода и редактирования текста страницы, в нижней части экрана можно наблюдать, как изменяется образ этой страницы, формируемый одним из браузеров, установленных на машине пользователя (если их несколько, то можно выбрать, каким именно).

Создатели HotDog (фирма Sousage) широко использовали средства мультимедиа. Многие процессы при работе сопровождаются довольно нетривиальными звуками, которые, конечно, отнимают некоторое количество системных ресурсов, но заметно поднимают настроение пользователя, что в конечном итоге приводит к повышению производительности его труда. Впрочем, если «собачьи голоса» начнут раздражать, то их легко отключить...

Интерфейс редактора нагляден и достаточно точно соответствует функциональному назначению этой программы. Очень развиты средства поддержки пользователя, много интересных решений в рамках традиционных возможностей графического Windows-ориентированного интерфейса. К достоинствам HotDog следует отнести множество сервисных функций - «изюминок», которых так не хватает при работе над страницами в более серьезных системах разработки Web-материалов. К таким возможностям относятся:

- «многоместный» буфер переноса, в который можно поместить часто используемые текстовые фрагменты и извлекать их оттуда по мере необходимости;
- полный список тегов языка HTML со всеми их атрибутами, организованный в виде упорядоченного по алфавиту дерева (представлено в правой части рис. 8.23), из которого теги можно просто перетаскивать на разрабатываемую страницу традиционным способом drag-and-drop;
- нумерация строк исходного текста и линейки на результирующем образе страницы, которые можно независимо включать и отключать.

Этот перечень далеко не полон; редактор поддерживает множество интернет-технологий, — от анимационной графики и сенсорных карт, до апплетов и CGI-скриптов. Конечно, степень поддержки довольно скромная, но вполне соответствует той «экологической нише», для которой этот редактор создан. В заключение следует отметить большое внимание, которое уделила фирма Sousage, вопросам поддержки пользователя. Начиная с процедуры инсталляции HotDog, ощущаешь заботу и уважение к пользователям самых разных уровней подготовки, а настройка интерфейса программы доставит истинное удовольствие многим ценителям гибкости, наглядности и простоты (во всяком случае, версия 5.0 на наш взгляд в этом отношении практически безупречна). Итогом краткого рассмотрения HotDog 5.0 может служить рекомендация этого средства разработки как для новичков, так и для опытных авторов, особенно в качестве альтернативы Notepad и другим неспециализированным редакторам.

9. ПРОДОЛЖЕНИЕ СЛЕДУЕТ, ИЛИ ТЕНДЕНЦИИ И ПЕРСПЕКТИВЫ РАЗВИТИЯ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

На данный момент уже достаточно четко обозначились четыре основные тенденции в развитии технологий создания пользовательского интерфейса:

1. Интеграция интерфейса «настольных» приложений с Web-интерфейсом.
2. Унификация интерфейса приложений, созданных на различных аппаратно-программных платформах.
3. Повышение уровня адаптивности («интеллектуальности») интерфейса.
4. Более широкое внедрение мультимедийных технологий в интерфейс приложений, вне зависимости от их функционального предназначения.

Изменение стандартов пользовательского интерфейса по первому направлению происходит одновременно с развитием Интернет-технологий. Причем процесс идет настолько бурно, что прогнозы даже на не очень отдаленную перспективу - дело весьма затруднительное. Практически каждый новый программный продукт, предоставляющий пользователям тот или иной сервис в Интернете, добавляет очередной штрих к интерфейсу «настольных» приложений. Судя по всему, уже в ближайшее время невозможно будет отделить компоненты интерфейса, обеспечивающие взаимодействие пользователя с локальными ресурсами его ПК, от компонентов, предоставляющих ему доступ к сетевым ресурсам. Ярким примером такого слияния является новая версия диалоговых панелей для работы с файлами (Save, Open, Save as), используемых в MS Office 2000. Они теперь поддерживают работу пользователя с папками специального типа - Web Folders, а в список фильтров Files of type добавлены типы файлов .htm, .html и .uri; это позволяет выполнять соответствующие операции с Web-документами как с локальными данными.

«Экспансия» Web-интерфейса во все другие области применения информационных технологий позволяет говорить о том, что три остальные направления развития пользовательского интерфейса лишь способствуют реализации этой «генеральной линии». Тем не менее каждое из них заслуживает отдельного комментария.

Возможность унификации интерфейса приложений, работающих на различных платформах, основана на совместном использовании клиент-серверных и компонентных технологий. Суть имеющихся на сегодняшний день решений заключается в том, что наэкране пользовательского (клиентского) ПК отображаются лишь визуальные элементы интерфейсной части приложения, а обработка действий пользователя возлагается на серверную компоненту. Надо сказать, что по такой технологии изначально была организована работа «конкурента» MS Windows - графической среды XWindow, созданной для семейства UNIX-систем. И не случайно наиболее удачные решения по реализации «межплатформного» интерфейса были получены в тех случаях, когда ставилась задача интеграции MS Windows с UNIX-системами.

Одним из таких решений является пакет Winted, выпущенный корпорацией TriTeal в конце 1998 года. Пакет предназначен для интеграции настольных систем, работающих под управлением MS Windows (95/NT Workstation), и UNIX-систем. Он предоставляет пользователям так называемый CDE-интерфейс (Common Desktop Environment - единый Рабочий стол), содержащий виртуальные экраны, а также единые для обеих систем службы работы с файлами и печати. Кроме того, обеспечивается передача данных между приложениями через буфер обмена.

Важным достоинством Winted является то, что он поддерживает DCD-технологию. Благодаря этому щелчок мышью на имени или пиктограмме файла данных приводит к запуску связанного с ним приложения, независимо от того, для какой из двух платформ оно было разработано.

Другой, не менее эффективный вариант интеграции приложений основан на применении технологии ICA (Independent Computing Architecture), разработанной корпорацией Citrix Systems. Компания QSSL встроила программную поддержку ICA в операционную систему реального времени QNX. Это делает возможным распространение существующих Windows-приложений практически на все виды встроенных систем или тонких клиентов. В результате приложения типа Word или Excel могут выполняться, например, даже на портативных вычислительных устройствах с емкостью ОЗУ 4Мб. Такой эффект достигается за счет того, что сетевой протокол ICA передает на сервер действия пользователя, связанные с нажатием клавиш клавиатуры, кнопок мыши, ее перемещением, а также с обновлением экрана (рис. 9.1).

В настоящее время существует два программных продукта, обеспечивающих интеграцию графического интерфейса для QNX (Photon microGUI) с другими графическими средами - Phindows (Photon in Windows) и Phinx (Photon in X). Первый из них, в полном соответствии со своим названием, предназначен для интеграции QNX с Windows-приложениями, а второй решает ту же задачу применительно к XWindow.

При обсуждении возможных способов повышения «интеллектуальности» пользовательского интерфейса мы вновь вынуждены отделить Web-интерфейс от интерфейса приложений, используемых в других предметных областях. Объясняется это основным предназначением Интернета - предоставление посетителю интересующей его информации. Работа нынешних поисковых систем основана на ранжировании текста по ключевым словам. Поэтому применительно к ним говорить об интеллектуальной адаптации к потребностям пользователя не приходится. Наиболее перспективными направлениями изменения ситуации являются два:

- реализация естественно-языкового интерфейса;
- использование динамической (изменяемой) модели пользователя.

Рис. 9.1. Реализация интегрированного интерфейса по технологии ICA

Естественно-языковой интерфейс предполагает наличие процедур лексического и семантического анализа текста. На этих принципах построены средства интеллектуального поиска данных (data mining), способные выявлять скрытые закономерности. Работы в этой области ведутся очень активно, в том числе и в России [11], но говорить о практических результатах пока рано.

Другой аспект естественно-языкового интерфейса - это независимость результатов поиска от языка запроса. Другими словами, посетитель Интернета должен иметь право формулировать запрос на родном языке и получать интересующие его сведения независимо от того, на каком языке они представлены в Интернете. Идеальный вариант - когда результат поиска также представляется на родном языке посетителя. С целью объединения усилий различных категорий специалистов для решения указанных проблем была сформирована специальная отрасль информационных технологий - **языковая инженерия**. Одним из проектов, относящихся к этой отрасли, является система MULINEX, над которой работают специалисты из Германии, Франции и Италии. MULINEX должна обеспечивать избирательный доступ к информации, просмотр и навигацию в многоязычной среде по запросу,

сформулированному на любом из поддерживаемых языков. Важным направлением работ в области языковой инженерии являются исследования по применению системы кодирования текстовой информации UNICODE, обеспечивающего работу с многоязычными текстами.

Более подробную информацию по проблемам языковой инженерии можно получить по следующим адресам:

- <http://www.linglink.lu/le/en/mdex.litml> - европейский сервер по языковой инженерии;
- <http://mulinex.dfki.de/> — проект MULINEX;
- <http://www.unicode.org/> — проект UNICODE.

Адаптация интерфейса в соответствии с моделью (характеристиками) пользователя предполагает наличие средств построения этой модели. Существующий уровень аппаратного и программного обеспечения не позволяет реализовать эти средства таким образом, чтобы они выполняли свои функции, не замедляя работу пользователя с «настольным» приложением. Другое дело - Интернет. Темп взаимодействия пользователей с сетевыми ресурсами значительно ниже скорости их работы с «настольными» приложениями. Поэтому дополнительные затраты времени на адаптацию здесь менее заметны. Вместе с тем, и для Интернета уже назрела необходимость перехода от ресурсоемких автономных приложений, предназначенных для настройки содержимого узла (групповых фильтров и продуктов типа Learn Sesame) к более изящным решениям. Например, на основе нейронных сетей. Судя по имеющимся тенденциям, такие решения будут получены сначала именно для Интернета, и лишь после этого реализованы в «настольных» приложениях.

Внедрение мультимедийных технологий связано, как известно, с расширением диапазона средств взаимодействия пользователя с компьютером. Наряду с основным каналом приема информации - зрительным - задействуются и те, которые человек обычно использует в реальной жизни (слуховой, тактильный, обонятельный, кинестетический). Кроме того, расширяются и формы представления визуальной информации - двумерное, «плоское» изображение заменяется трехмерным, значительно шире и разнообразнее применяются анимация и видеоклипы. Очевидно, что перспективы развития мультимедийных технологий связаны в первую очередь с совершенствованием имеющихся и появлением новых устройств ввода и вывода данных (разумеется, в сочетании с разработкой соответствующего программного обеспечения).

Следуя основной концепции книги, мы приведем лишь несколько примеров имеющихся достижений в области создания мультимедийных устройств ввода, сосредоточив основное внимание на перспективах визуализации информации.

Первый из таких примеров - это устройство, обеспечивающее бесконтактный ввод команд в компьютер, одной лишь «силой мысли» пользователя. Устройство, получившее соответственное название — Mind Drive, — разработано фирмой The Other 90% Technologies в 1997 году и по внешнему виду напоминает увеличенный раз в три наперсток. Его работа основана на анализе информации, поступающей от вмонтированных в «наперсток» датчиков. Такой информацией являются пульс, температура кожи, ее проводимость и электрическая активность, а также скорость изменения этих показателей. Для работы с Mind Drive требуются определенные навыки, на формирование которых уходит один-два часа. Пока устройство способно выдавать только один аналоговый сигнал, поэтому для управления объектами в двух измерениях необходимо переключать управление с помощью

клавиатуры. В планах фирмы The Other 90% Technologies - разработка многокоординатных устройств и специального «словаря» для распознавания типов сигналов.

Более близкие перспективы связаны с использованием сенсорных и голосовых технологий.

Считается, что сенсорный ввод очень хорошо согласуется с генетически заложенным в человеке стереотипом поведения: чтобы усилить субъективное ощущение надежности источника (или приемника) информации, человеку хочется его потрогать. Вспомните, например, как мы любим водить пальцем по карте, отыскивая нужную точку, или как многие, у кого берут интервью, пытаются отобрать микрофон у интервьюера (хотя ни то, ни другое на самом деле не требуется). Еще одно достоинство сенсорных технологий - повышенная защищенность от ошибочных действий оператора. Поскольку при использовании сенсорного ввода клавиатура на экране монитора формируется программно, то на экране можно оставлять только те клавиши, которые необходимы в данный момент. Программно-сенсорные экраны позволяют также полностью эмулировать работу со стандартной мышью. Драйвер позволяет описать реакцию на нажатие, отжатие, двойное прикосновение к экрану.

В настоящее время на рынке сенсорных технологий ведущую роль играют четыре:

- на основе поверхностных акустических волн;
- на основе изменения распределенной емкости;
- на основе инфракрасных волн;
- резистивная технология.

Несмотря на различия физических принципов, положенных в их основу, все эти технологии достаточно близки по предоставляемым возможностям, за исключением обеспечиваемого разрешения. Эта характеристика изменяется в широких пределах: от 64 точек на кв. дюйм для инфракрасных волн до 100 000 точек на кв. дюйм для резистивной технологии.

В начале 1998 года появилась еще одна разновидность сенсорного экрана - Scribex фирмы ЕЮ. Такие экраны предоставляют возможность рукописного ввода информации.

Существующие в настоящее время аппаратно-программные средства, реализующие голосовые технологии, обеспечивают точность распознавания речи не более 95% [12]. Это означает, что при голосовом вводе одной страницы печатного текста примерно 20 слов воспринимаются неправильно. В некоторых случаях это может привести к искажению смысла текста. Вместе с тем, такая точность приемлема при вводе отдельных команд. Примерами программ, предназначенных для обработки слитной речи, являются Voice Xpress Plus фирмы Lernount&Hauspie Speech Products N.V. и NaturallySpeaking фирмы Dragon Systems. Оба эти продукта рассчитаны на постоянного пользователя, а точность их работы повышается по мере адаптации к голосу; процесс начальной адаптации занимает около 40 минут. Различия между ними заключается в том, что Voice Xpress Plus встраивается непосредственно в текстовый процессор MS Word, а NaturallySpeaking имеет собственную текстовую программу, которую необходимо перед применением копировать в используемый текстовый процессор. Voice Xpress Plus позволяет не только вводить текст, но и форматировать его с помощью команд меню Word.

И все-таки, «лучше один раз увидеть, чем сто раз услышать». Поэтому вернемся к перспективным технологиям визуализации информации, отображаемой на экране.

Речь идет в первую очередь о технологии создания *виртуальных миров* на основе языка моделирования виртуальной реальности VRML (Virtual Reality Modeling Language). С помощью VRML в Интернете можно создавать управляемые трехмерные пространства с гинерсвязями, называемые «мирами» [13]. VRML не просто язык, позволяющий ввести трехмерность в Интернет, — это одна из наиболее перспективных технологий, которая открывает новые возможности в организации общения между пользователем и компьютерными системами. Благодаря применению VRML Интернет становится более «осязаемым», и путь к Web-узлу может быть задан практически также, как мы указываем маршрут в реальном мире, например, «Вам следует подняться на второй этаж и войти во вторую дверь налево». Если же посетитель знает маршрут, то ему достаточно щелкнуть мышью на изображении нужной двери (или на каком-то другом объекте), чтобы оказаться в требуемой точке Интернета. Другими словами, в Интернете виртуальный мир может играть роль трехмерной сенсорной карты. Отличие состоит в том, что по воле разработчика посетитель оказывается как бы «внутри» этой карты. Основным препятствием для широкого распространения VRML-технологии является ее ресурсоемкость. VRML-файлы имеют большой объем, а поскольку выполняются они в режиме интерпретации, то реализуемые на их основе виртуальные миры оказываются весьма «неповоротливыми». Указанную проблему можно в значительной степени преодолеть, используя VRML совместно с HTML и Java.

Именно такой подход применила фирма Microsoft при разработке продукта Chrome, предназначенного для создания и исполнения мультимедийных Web-приложений (его можно также использовать для построения настольных мультимедийных систем). Предполагается, что уже в ближайшее время Chrome будет реализован в виде дополнительного модуля для Windows 2000 и Windows NT. Это позволит использовать его в качестве стандартного средства построения трехмерного графического пользовательского интерфейса. Другим примером применения Chrome является генерация трехмерного куба, грани которого отображают различные формы представления визуальной информации (текст, графику, видео). Состав отображаемых на экране элементов содержимого зависит от того, как пользователь взаимодействует с этим виртуальным кубом.

Еще одной причиной, сдерживающей широкое распространение VRML, является достаточно сложный синтаксис языка (по сравнению с XML и HTML). Это обуславливает необходимость создания развитых визуальных VRML-редакторов, которых пока не очень много и которые не позволяют реализовать все возможности языка. Два наиболее используемых VRML-редактора, предназначенные для работы в среде Windows - Pioneer фирмы Caligari и 3-D Website Builder фирмы Virtus. Описание их основных функциональных возможностей можно найти в [13].

Реализация рассмотренных направлений делает все более актуальным вопрос стандартизации пользовательского интерфейса на глобальном уровне, вне зависимости от аппаратно-программной платформы и предметной области, для которых разрабатывается то или иное приложение.

10. ПРИЛОЖЕНИЕ

КАТАЛОГ ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ ПРОЕКТИРОВАНИЯ И РАЗРАБОТКИ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

Название продукта	Разработчик; URL	Платформа (среда)	Примечание
Системы управления пользовательским интерфейсом (DIMS)			
Alpha UIMS	LoneWolf Systems http://www.lonewolf.com/	X, Windows NT	\$7995
Amulet	Brad Myers Human-Computer Interaction Institute, Carnegie Mellon University	X, MS Windows	Свободно распространяемый продукт (FREE)
Andrew User Interface System	Fred Hansen, Andrew Consortium, Carnegie Mellon University http://www.cs.cmu.edu/afs/cs.cmu.edu/project/atk-ftp/web/andrew-home.html	X	Свободно распространяемый продукт
Autocode	Integrated Systems	UNIX, VMS	\$20,000 ориентирован на системы реального времени, в том числе используемые в авиации
Chiron	Richard Taylor, CS, Univ Calif., http://www.ics.uci.edu/Arcadia/Chiron/chiron.html	X: Motif или Openlook	Свободно распространяемый продукт
Choreographer	Company apparently defunct	PC/OS2 PM	\$7,500
CUM	International Lisp Associates, Cambridge, MA	Common Lisp	
EaselII	Easel	DOS, OS/2	\$10,000
FormsVBT	Marc H. Brown, DEC Systems Research Center http://www.research.digital.com/SRC/modula-3/html/home.html	Modula-3 для X и Windows	Свободно распространяемый продукт
Garnet	Brad Myers, CMU, School Computer Science, http://cs.cmu.edu/project/garnet/www/garnet-home.html	Common Lisp, X или Mac	Свободно распространяемый продукт
GINA	GMD (German National Research Center for Computer Science) http://zeus.gmd.de/i3/mmk/diva/gina/home.html	LISP / Motif	Свободно распространяемый продукт
InterMAPhics	Gallium Software Inc. http://www.gallium.com/ProductInfo/InterMAPhics	Sun, DEC, большинство UNIX-платформ	\$45,000 Ориентирован на системы реального времени, в том числе используемые в авиации
BA Dialog Manager	BA Informationssysteme GmbH	UNIX/Motif, UNIX/ASCII, VMS/Motif, VMS/ASCD, OS/2, Windows (3.1, NT, '95)	
JAM	JYACC http://www.jyamc.com	Практически для всех платформ	\$6000 Ориентирован на приложения, работающие с базами данных

Open Dialogue	HP/Apollo Computer	X	язык описаний (declarative language)
OpenUI	Open Software Associates Inc http://www.osa.com/	Windows (3.1, 95, NT), MAC, Motif, UNIX, VMS	\$5,000 Ориентирован на INTERNET и распределенные приложения
Oracle Tools	Oracle Corporation	Практически для всех платформ	\$6,000 Ориентирован на приложения, работающие с базами данных
RIPL	Computer Technology Assoc.	VAXStation, VMS	Свободно распространяемый продукт (для некоммерческого использования)
Serpent	Carnegie Mellon Univ./SEI ftp://ftp.sei.cmu.edu/pub/serpenV	X	Свободно распространяемый продукт
SET	CasetCorp.	X	\$5,000 Инструмент моделирования сценариев диалога
SL-GMS	SL Corp.	X, VMS	\$12,500 Ориентирован на системы реального времени
Название продукта	Разработчик; URL	Платформа (среда)	Примечание
Sim-	Randy Pausch, UVA, Dept. Comp. Sci http://www.cs.virginia.edu/~suit/	Платформнонезависимый	Свободно распространяемый продукт; используется для обучения
Thistle	Language Technology Group, University of Edinburgh http://www.ltg.ed.ac.uk/software/thistle/index.html	Java	Свободно распространяемый продукт (при условии некоммерческого использования)
VAPS	Virtual Prototypes http://www.VirtualPrototypes.CA/	SUN and HP	\$10,000-\$41,500
WINTERP	Niels P. Mayer http://www.cybertribe.com/mayer/winterp/	UNIX/X/ Motif	Свободно распространяемый продукт
Инструментальные средства проектирования и разработки интерфейса (liter-face Builder)			
Action!	ExperTelligence	Lisp для Mac	\$595
ActivAda for Windows	Thomson Software Products http://www.thomsoft.com	Windows	\$995
AdaSAGE	Idaho National Engineering Laboratory (INEL), Lockheed Martin Idaho Technologies	DOS, Windows 3.1/95/NT, UNIX, Sun, AT&T, IBM RS6000	
Altia Design	Altia	UNIX, Windows	
AppMaker	Bowers Development http://members.aol.com/bowersdev/index.html	Mac	\$295
Builder Xcessory	Integrated Computer Solutions, Inc. http://www.ics.com	UNIX/X/Motif	\$3,200
CanAda	DAINA Engineering http://wuarchive.wustl.edu/languages/ada/swtools/canada/	MS Windows/ Ada	Свободно распространяемый продукт

Case PM	Casework	PC/OS2 PM	\$1,995 Пакет CASE-средств
Data Views	DataViews Corporation http://www.dvcorp.com/	UNIX, VMS	\$17,700
Название продукта	Разработчик; URL	Платформа (среда)	Примечание
DevGuide	Sun	Open Windows Devel, OpenLook	\$250
Display Construction Set	AT&T	UNIX, X, OpenLook	
Druid	Gurminder Singh, Institute of Systems Science, National University of Singapore http://www.iss.nus.sg/public/ISSOTHER/druidnew.html	X and Motif	\$1250
ExoCODE	EXOC	Motif, OpenLook, SunView	\$1,500
EZX	Sunrise Software Sys	Motif	\$3,500
Forms	Mark H. Overmars, Department of Computer Science, Utrecht University ftp://ftp.cs.mu.nl/pub/SGVFORMS/	SGIGL	Свободно распространяемый продукт
GENIE	Advantech http://www.prosoft.ru	Windows 3.*/*95/98	HMVSCADA-система
GENESE32	ICONICS http://www.prosoft.ru	Windows 95/98/NT	HMVSCADA-система
GIB	TAO Research Corp.	MS Windows	\$475
GRAM MI	SETT, Inc.	Ada,X	
HP Interface Architech	Hewlett Packard	UNIX/X	
MacA&D, WinA&D	Excel Software, http://www.excelsoftware.com/	Mac, Windows	\$1995CASE-средств
MotifGUDE	OlafBecker, CANADA http://www3.bc.sympatico.ca/Guide/	UNIX	Свободно распространяемый продукт
Next Interface Builder	Next, Inc.	UNIX/NeXT	
Название продукта	Разработчик; URL	Платформа (среда)	Примечание
ObjectBuilder	Openware Technologies http://www.openware.com/	Sun OS, Solaris, HP, BM, SCO.SGI.DEC, (планируется выпуск версий для Windows 95/NT)	\$5,400
Omnis7	BIyth Software, Incorporated http://www.biyth.com	Mac, PC	\$500
Open Interface	Neuron Data	Motif, OpenLook, PM, Windows, Mac	\$10,000
PowerCharger for MFC	ViewSoft Inc. http://www.viewsoft.com	Visual C++/MFC	\$199

Progress Version 7	Progress Software Corporation	Windows & Motif & DOS	\$300
RTWin	SWD Системы реального времени http://www.swd.ru	QNX -	SCADA-система
Sammi	Kinesix http://www.kinesix.com/	UNK(HPUX, IBM AIX, SCO, SunOS, Solaris, OSF/1, Ultrix, IRIX, REAL/EK, UnixWare, QNX, Lynx, Venix,	Ориентирован на системы реального времени
TAE Plus	Century Computing http://www.cen.com/tae/	X	Свободно распространяемый продукт для пользователей NASA
Teleuse	Aonix http://www.aomx.com/Products/UIMS/uims.html	Motif	\$7,500
Tigre Interface Designer	Tigre Object Systems	Smalltalk; MS Windows, Mac, UNIX	\$1,500
Tk/Tcl	Scriptics Corporation http://www.scriptics.com	X/11, PC, Mac	\$1000
UIM/X	Visual Edge Software, LTD	Motif	
UIM/X	Bluestone, Inc. http://www.bluestone.com	для всех платформ, за исключением HP	\$5,000
Vennont Views with	Vermont Creative Software http://www.vtsoft.com	DOS, UNIX	
Visaj	Imperial Software Technology http://www.ist.co.uk/visaj	Java	
Visual/Recital	Recital Corporation	Motif/X11	\$4000
Vsystem	Vista Control Systems http://www.vista-control.com	X/Motif, DEC VMS, Vaxeim, OSF/1	
Windows-MAKER	Blue Sky Software http://www.blue-sky.com/	Windows 3.0	\$795
Xbuild	Siemens Nixdorf, 4 Cambridge Center	UNIX/X/Motif	\$1,895
X-Designer	Imperial Software Technology http://www.ist.co.uk/xd	X Windows, Java	\$3,500
XFaceMaker	Nova Software Labs http://www.nsl.fr	UNIX X/Motif	
XVT	XVT Software Inc http://www.xvt.com/xvt	MS Windows, Windows NT, OS/2, Macintosh, OSF/Motif,	\$1950-\$6300
zApp	Rogue Wave Software, Inc. http://www.roguewave.com/products/zapp/	DOS, Windows(16b), Windows 95/NT, OS/2, Warp, HP-UX, BM AIX, SCO UNIX, SunOS, Solaris,	
Zinc	Zinc Software Inc. http://www.zinc.com	DOS, MS Win, OS/2, Mac, Motif	\$500
Инструментальные средства разработки интерфейса (Tools&Toolkit)			
Action!	Macromedia http://www.macromedia.com/Tools/Action/index.html		\$100 Ориентирован на разработку мультимедийных приложений

Actor	The Whitewater Group	PC/Windows	\$475
Ada95 GUI Library	Asterisk Business Solutions http://www.asterisksolutions.co	PC, Mac, Motif, b-ix, Solaris	
Aspect	Open Inc.	Motif, OpenLook, Windows, OS2PM	\$800-5000 Virtual Toolkit
COBOL sp2	Flexus	Windows	\$1,195 Средство разработки пользовательского интерфейса для COBOL-программ
DesignInReal Time (DIRT),	University of Kent-Canterbury	X	Свободно распространяемый продукт
Fresco	X Consortium Inc. http://www.x.org/consortium/projects.html ; ftp://ftp.x.org/pub/R6untarred/	C++/X/UNIX	Свободно распространяемый продукт
Galaxy	Ambiencia Information Systems, Inc.	Mac, Windows, Motif, OpenLook	Virtual Toolkit
Groupkit	Saul Greenberg, University of Calgary, Dept of Computer Science, http://www.cpsc.ucalgary.ca/projects/grouplab/projects/	UNIX, Tcl/Tk, Tcl-DP	Свободно распространяемый продукт
GX Series Developer's Pak	Genus	PC	\$589
LOG Views	LI-OG Inc. http://www.ilog.com	UNIX, OS/2, Windows 3.1, NT	\$5,000
INT Widgets	INTCorp. http://www.int.com	X/Motif	\$1750-\$3000
Interviews	Stanford University ftp://interviews.stanford.edu	C++/X/UNIX	Свободно распространяемый продукт
ivtools	VectaportInc. http://www.vectaport.com/ivtoo	C++/X/UNIX	Свободно распространяемый продукт
KEE	Intelli Corp	USP (PC, UNIX)	\$5,000
Knowledge Pro	Knowledge Garden, Inc. http://www.kgarden.com	PC	\$449 Инструмент разработки экспертных систем
Lab VIEW	National Instruments	DOS/Windows/Mac	Инструмент визуального моделирования
MetaCard	MetaCard Corporation http://www.metacard.com	UNIX, Windows NT, Windows 95.	\$995 Средство разработки мульти- и гипермедийных приложений
Macintosh	Apple	Macintosh	
MATLAB Guide	Math Works, Inc. http://www.mathworks.com http://www.softline.ru	Windows 95, NT Linux, Mac	Демо-версия распространяется свободно
Motif	Open Software Foundation http://www.osf.org/motif/index .	UNIX, X/11	\$500
MrEd	Matthew Flatt, Department of Computer Science http://www.cs.rice.edu/~mflatt/mred.html	Motif; XView; MSWindows	Свободно распространяемый продукт
New Wave	Hewlett Packard	PC	\$195
Next Step	Next, Inc.	UNIX/NeXT	

01	Openware Technologies	Sun OS, Solaris, HP, BM, SCO, SGI, DEC (в ближайшем будущем -Windows 95/NT)	\$5,400
OUT	Sun	UNIX, X, OpenLook	
POWERMED IA	OmniSoft	MS-Windows Window-NT	\$79 Средство разработки мультимедийных приложений
Presentation Manager	Microsoft	OS/2	
Proteus 5.0	Genus	PC	\$249
Qt	Troll Tech AS http://www.troH.no/qtinfo.html	Windows 95, NT, Linux, Solaris, HP-UX, AIX, Digital UNIX, IRIX, FreeBSD, BSDI	Свободно распространяемый продукт
Rapid Design	Emultek Inc http://WWW.emultek.com	Windows 95/NT	\$6,000 Язык визуального программирования
StarView	Star Division Corp.	MS-Windows 3.1, OS/2 2.1, Mac, Motif	\$495
Theseus	Computer Graphics Center ZGDV http://zgdv.igd.fhg.de/software/	X, Motif, C++	Свободно распространяемый продукт
Tilcon Real- Time Developer	Tilcon Software Ltd http://www.tilcon.com/	UNIX:QNX, Linux; Windows 95/98/NT, Windows CE	Средство разработки систем реального времени
UIM/X	Black & White Software http://www.blackwhite.com	UNIX	\$5000
V	Bruce E. Wampler, Department of Computer Science http://www.objectcentral.com	X Windows, Microsoft Windows3.1, C++	Свободно распространяемый продукт
ViewSoft Internet.	ViewSoft Inc. http://www.viewsoft.com	Internet	\$2,995 Средство разработки Интернет-приложений для тонких клиентов
VISION	Unify Corporation http://www.unify.com/	HP-UX Sun Solaris Dec UNIX, IBM AIX, MS-Windows, Window-NT, Macintosh	\$4,995 virtual toolkit для разработки баз данных и клиент-серверных приложений
Windows Develop. Kit	Microsoft	Windows	
wxWindows	Julian Smart	Windows, UNIX (Motif & GTK)	Свободно распространяемый продукт, virtual toolkit
X-In-Ada	Top Graph'X	X, PEXand Motif in Ada	
XRT	KL Group Inc. http://www.klg.com/	Motif	\$995 Widget Libraries, Also components for Windows and Java.
XView	Sun	UNIX/X OpenLook	Свободно распространяемый продукт

YACL	M . A. Sridhar, University of South Carolina http://www.cs.sc.edu/~sridhar/yacl.html	Windows, OS/2, X/Motif	Свободно распространяемый продукт Virtual Toolkit, C++ Class Library.
Средства прототипирования интерфейса (Prototyping Tools))			
Authorware	Macromedia http://www.macromedia.com/Tools/Authorware/index.html	PC	\$995
Demo-П	Lifeboat Publishing	PC	\$249 Средство построения диаграмм
Director	Macromedia http://www.macromedia.com/	Mac, PC/Windows	\$598
ICON Author	AimTech Corp	Windows 3.0	\$995
Protofinish	Genesis Data Systems	PC	\$300 Средство построения диаграмм
Protoscreens	Bailey & Bailey	PC	\$225 Средство построения диаграмм

11. ГЛОССАРИИ

А

accelerator key — клавиша-акселератор; см. *shortcut key*.

access key — клавиша доступа (мнемоническая клавиша) — клавиша, обеспечивающая быстрый выбор связанного с ней элемента интерфейса; как правило, это текстовая клавиша, обозначение которой соответствует подчеркнутому символу в названии пункта (раздела) меню или кнопки.

accessibility — доступность — свойство программного обеспечения, которое делает его пригодным и доступным для самого широкого круга пользователей, включая пользователей — инвалидов.

action — действие — некоторая предопределенная функция, выполняемая приложением; пользователь запрашивает требуемое действие различными способами:

вводом команды, нажатием функциональной клавиши, выбором пункта меню и т.д.

action handle — способ управления объектами приложения, при котором доступ к операциям над выбранным объектом обеспечивается с помощью всплывающего меню или на основе прямого манипулирования.

active — активен — состояние объекта, на котором установлен фокус ввода.

active end — текущая граница — граничная точка области выбора; как правило, такой точкой является объект, ближайший к горячей точке указателя мыши, когда пользователь выпускает кнопку мыши или поднимает конец пера от экрана (сравн. *anchor point*).

active window — активное окно — окно, в котором в данный момент работает пользователь. Активное окно обычно является окном самого верхнего уровня и отличается от других окон цветом полосы заголовка.

adornment — элемент управления, пристыкованный к краю диалоговой панели или окна (например, панель инструментов или регулятор).

anchor point — якорь — позиция, указывающая начальную точку (первый объект) области выбора. Якорь обычно устанавливается на объект, ближайший к указателю, когда пользователь нажимает кнопку мыши или прикасается пером к экрану (сравн. *active end*).

anti-aliasing — сглаживание — способ создания (или коррекции) растровых изображений, позволяющий избавиться от «ступенек» — эффекта, который имеет место при рисовании в растровом формате наклонных линий.

application option — режим приложения — вариант представления или взаимодействия, реализованный в приложении по усмотрению разработчика.

Apply — Применить — команда, обеспечивающая передачу приложению внесенных изменений или установок, сделанных во вторичном окне (как правило, без закрытия этого окна).

aspect ratio — пропорция — показатель, отражающий соотношение ширины и высоты какого-либо объекта или изображения.

auto-exit — авто-выход — автоматический переход из текстовой области, в которой находится фокус ввода, на другой элемент управления, после того как пользователь вводит последний символ.

auto-joining — авто-слияние — автоматическое перемещение текста с целью заполнить промежуток, образовавшийся после того, как пользователь удалит фрагмент текста.

automatic scrolling — авто-скроллинг — автоматическое перемещение отображаемой в активном окне области без прямого взаимодействия пользователя с полосой прокрутки.

auto-repeat — автоматическое повторное выполнение события или операции; как правило, имеет место в тех случаях, когда пользователь удерживает в нажатом состоянии клавишу или непрерывно воздействует на элемент управления (например, на кнопку полосы прокрутки).

available — доступный — состояние объекта или элемента интерфейса, при котором пользователь может с ним взаимодействовать.

В

background — фон — графическая область, на основе которой формируется основное содержимое Web-страницы.

bullet — маркер — графический символ, используемый при создании Web-страницы для визуального выделения тех или иных ее элементов.

С

Cancel — Отменить — команда, обеспечивающая прерывание выполняемой операции или процесса и возврат в исходное состояние, (сравн. *stop*).

cascading menu — иерархическое (каскадное) меню — способ организации меню, при котором один или несколько пунктов содержат меню более низкого уровня (дочерние меню — *child menu*, или субменю — *submenu*).

Cascading Style Sheets — каскадные таблицы стилей — технология введения и тиражирования дополнительных атрибутов при создании Web-страниц.

check box — флажок — стандартный элемент GUI, который визуальным образом отображает установку (или выбор) некоторого состояния, свойства или значения параметра; флажок может быть *установлен* или *снят*. (сравн. *option button*).

check mark — маркер, «птичка» — графический символ, используемый в качестве визуального признака того, что соответствующий флажок установлен (или выбран данный элемент в списке расширенного выбора).

child menu — дочернее меню — см. *cascading menu*.

child window — дочернее окно — первичное окно, содержащее документ (данные) и расположенное в пределах родительского окна; используется при реализации приложения на основе многодокументного интерфейса — *MDI*.

chord — аккорд — одновременное нажатие более чем одной кнопки мыши.

click — щелчок — кратковременное нажатие кнопки мыши, когда указатель установлен над объектом или интерактивным элементом GUI; Сравн. *press*.

client — клиент — программа, пользующаяся «услугами» (функциональными возможностями) другой программы (сервера — *se.n'er*)

Clipboard — буфер обмена — область памяти для хранения объектов, данных (или ссылок на них), для которых пользователь выполнил команду Копировать (Copy) или Вырезать (Cut).

Close — Закрыть — команда, приводящая к закрытию (удалению) окна, к которому она применяется.

code page — кодовая страница — таблица, описывающая способ кодирования символов.

collection — коллекция, набор — множество объектов, имеющих некоторое общее свойство.

column heading — заголовок (столбца) — стандартный элемент GUI; позволяет идентифицировать информацию, отображаемую в виде таблицы, регулировать ширину столбцов таблицы; может также использоваться в качестве кнопки.

combo box — комбинированный список — стандартный элемент GUI; представляет собой объединение текстового поля и списка.

command — команда — мнемоническое обозначение запроса на определенное действие приложения или системы, сопровождаемое (если это необходимо) указанием соответствующих параметров; способ ввода команды определяется используемой структурой диалога.

command button — кнопка (управляющая кнопка) — стандартный элемент GUI, используемый в Windows-приложениях, который предназначен для инициализации определенной команды или для установки параметра.

common dialog action — унифицированное действие диалога — действие, имеющее общий смысл во всех приложениях, реализованных на данной платформе (например, прокрутка, копирование объекта и т.д.).

composite — композиция — набор или группа объектов, агрегация которых дает объект нового типа (например, ячейки в электронной таблице, объединенные в столбец).

constraint — объединение — отношение между объектами в наборе, при котором изменения на одном объекте влияют на другой объект.

container — контейнер — объект, который содержит другие объекты.

context menu — контекстное меню; см. *pop-up menu*.

context-sensitive Help — контекстно-зависимая помощь — справочная информация (подсказка) о выбранном объекте или текущей ситуации; представляет собой ответ на вопрос «Что это?» или «Как использовать это?»

contextual — контекстный — соответствующий конкретным условиям, в которых нечто существует или происходит.

contiguous selection — непрерывный выбор — область выбора, которая содержит логически или пространственно близкие (смежные) объекты.

control — элемент управления — объект, который обеспечивает взаимодействие пользователя с приложением (ввод данных, инициализацию команды или операции, установку значений параметров) или отображает информацию, необходимую для выполнения задания.

coordinated scrolling — согласованная прокрутка — одновременная прокрутка двух (или большего числа) областей окна, при которой содержимое этих областей (подокон) взаимно согласовано.

critical message — критическое сообщение — информация, выдаваемая приложением (или системой) для описания действия, которое должно быть выполнено пользователем для продолжения работы приложения; такой информацией, например, может быть предложение заменить испорченный гибкий диск.

cursor — курсор — общий термин для обозначения визуального индикатора, указывающего позицию, к которой относятся выполняемые пользователем действия. См. также *input focus*, *insertion point*, *pointer*.

D

data-centered design — работа, управляемая данными — способ организации пользовательского интерфейса, при котором первичными являются подлежащие обработке данные, а не программные средства, необходимые для выполнения задания.

data link — связь по данным — вид связи, при которой обеспечивается передача (пересылка) данных между двумя объектами или позициями.

default — использование по умолчанию — предустановленные значения параметров или predetermined способ выполнения операции, которые программная система или приложение используют в своей работе, пока пользователь не изменит эти установки явным образом.

default button — predetermined кнопка — кнопка, для которой клавишей-акселератором по умолчанию является клавиша <Enter>: нажатие этой клавиши приводит к выполнению связанной с кнопкой команды. Predetermined кнопка обычно используется только во вторичных окнах.

Delete — Удалить — команда, используемая для удаления объекта или значения; удаленный объект не сохраняется в буфере обмена, но может быть восстановлен с помощью команды *undo* (если она поддерживается приложением).

desktop — Рабочий стол (или настольная система) — графическая рабочая область, которая заполняет экран монитора и создает фон для всех выполняемых операций. Кроме того, Рабочий стол является контейнером и может быть использован в качестве «хранилища» объектов, с которыми взаимодействует пользователь в процессе работы.

dialog — диалог — 1) общий термин для обозначения процесса взаимодействия пользователя с приложением; 2) взаимодействие пользователя с predetermined набором интерактивных элементов, требуемое для выполнения конкретного задания.

dialog base unit — единица диалоговой панели — аппаратно-независимая единица измерения, используемая для форматирования пространства диалоговой панели. Единица диалоговой панели равна: по горизонтали — одной четверти средней ширины символов установленного системного шрифта, по вертикали — одной восьмой средней высоты символов установленного системного шрифта.

dialog box — диалоговая панель — вторичное окно, которое обеспечивает ввод пользователем дополнительной информации, необходимой для работы приложения.

disabled — недоступен — состояние объекта или элемента интерфейса, при котором пользователь не может с ним взаимодействовать.

disjoint selection — отдельный (непересекающийся) выбор — область выбора, состоящая из объектов, которые либо не связаны между собой логически, либо разнесены пространственно.

docking — стыковка — перемещение элемента интерфейса к краю другого элемента, в результате которого он оказывается *пристыкованным* (прижатым и выровненным по краю); например, панель инструментов может быть пристыкована к краю окна или подокна.

document — документ — абстрактная единица данных (как правило, файл), относительно которой организуется выполнение задания пользователем или взаимодействие между пользователями, использующими общие данные.

document window — окно документа — окно, которое обеспечивает визуальное представление содержимого документа.

double-click — двойной щелчок (кнопкой мыши) — два смежных быстрых нажатия кнопки мыши; обычно используется в качестве ускоренного способа инициализации команды (операции, процесса).

DPI — Dots Per Inch — число точек на дюйм — единица измерения разрешения цифрового или печатного изображения.

drag — перетаскивание — перемещение указателя при нажатой кнопке мыши.

drag and drop — «перетаски и оставь» — техника взаимодействия с объектами приложения или другими элементами GUI, при которой все операции выполняются на основе выбора и перемещения объектов с помощью мыши. Интерпретация (результат) выполняемой операции зависит от свойств позиции-источника и/или позиции-приемника.

drop-down combo box — выпадающий комбинированный список — стандартный элемент GUI, используемый в Windows-приложениях; представляет собой композицию текстового поля и выпадающего списка.

drop-down list box — выпадающий список — стандартный элемент GUI; как правило, содержит перечень возможных значений свойства объекта и отображает текущую установку, но может также использоваться для представления набора объектов.

drop-down menu — выпадающее меню — *меню*, которое отображается при выборе одного из разделов основного меню приложения (расположенного в полосе меню первичного окна приложения).

Е

edit field — поле редактирования; см. *text box*.

Edit menu — раздел основного меню, который содержит команды редактирования объектов, отображенных в пределах окна (например, *Вырезать*, *Копировать*, *Вставить*); реализуется в виде выпадающего меню.

ellipsis — эллипсис, или многоточие «...» — символ, используемый в качестве дополнения к названию пункта меню или кнопки, чтобы указать пользователю на то, что для выполнения соответствующей команды требуется ввод дополнительной информации. Когда пользователь выбирает такую команду, обычно отображается диалоговая панель, предназначенная для ввода этой дополнительной информации.

embedded object — вложенный (внедренный) объект; см. *OLE*. **enter** — ввод — унифицированное действие диалога, обеспечивающее передачу приложению установок (значений параметров, свойств и т.д.),

выполненных пользователем во вторичном окне; ввод также указывает приложению на необходимость выполнения выбранного действия над выбранным объектом.

event — событие — действие или событие, на которые может реагировать приложение. Примеры событий — нажатие клавиши, щелчок или перемещение мыши.

explicit selection — явный выбор — выбор, который пользователь выполняет с помощью одного из устройств ввода; сравн. *implicit selection-extended selection* — расширенный (дополненный) выбор — техника выбора, которая обеспечивает включение в существующую область выбора новых объектов.

extended selection list box — список расширенного выбора — список, который поддерживает множественный выбор, но изначально ориентирован на выбор единственного объекта или единственной области.

F

File menu — раздел основного меню, который содержит команды, обеспечивающие работу с файлами (например, *Открыть, Сохранить, Печать*); реализуется в виде выпадающего меню.

flat appearance — способ визуального представления элемента управления, когда он вложен в другой элемент управления или находится в перемещаемой области. **folder** — папка — тип контейнера для объектов (как правило, файлов). **font** — шрифт — набор атрибутов для текстовых символов. **form** — форма — интерактивный элемент Web-страницы, предназначенный для ввода посетителем тех или иных взаимосвязанных данных.

frame — фрейм — средство разбивки окна броузера на несколько прямоугольных областей (подокон), которые могут просматриваться независимо друг от друга.

function key — функциональная клавиша — клавиша, нажатие которой приводит к выполнению predetermined последовательности действий; обычно это клавиши F1...F12.

G

glyph — общий термин, используемый для обозначения любого графического образа или рисунка, который может отображаться на кнопке или в окне сообщения. Сравн. *icon*.

graphical user interface (GUI) — графический пользовательский интерфейс — пользовательский интерфейс, основанный на визуализации объектов, с которыми взаимодействует пользователь в процессе работы, а также самого процесса взаимодействия.

grayed — «обесцвеченный» — см. *unavailable*.

group box — группирующий блок — стандартный элемент GUI, используемый в Windows-приложениях; обеспечивает визуальное объединение взаимосвязанных элементов управления.

H

handle — специальный элемент интерфейса, добавляемый к объекту (обычно графическому) с целью облегчения взаимодействия с ним пользователя; как правило, обеспечивает выполнение операций перемещения, калибровки и т.п.

Help menu — раздел основного меню, который содержит команды, обеспечивающие доступ к справочной информации или другим средствам поддержки пользователя; реализуется в виде выпадающего меню.

heterogeneous selection — гетерогенный (разнородный) выбор — область выбора, которая содержит объекты с разными свойствами или относящиеся к разным типам.

hierarchical menu — иерархическое меню; см. *cascading menu*.

highlighting — подсветка — выделение цветом (в том числе инверсным) или повышенной яркостью выбранного объекта, области, пункта меню и т.п.

hold — удержание — достаточно продолжительное (не менее 1с) удержание клавиатурной клавиши или кнопки мыши в нажатом состоянии.

homogeneous selection — однородный (гомогенный) выбор — область выбора, которая содержит однотипные или обладающие одинаковыми свойствами объекты.

hot spot — горячая точка (активная зона) — 1) часть указателя (или устройства выбора), которая определяет точную позицию или объект, на которые указывает пользователь; 2). интерактивная область сенсорной карты, используемая для навигации по системе Web-страниц.

hot zone — горячая зона — интерактивная область конкретного объекта или позиции, в пределах которой должна находиться горячая точка указателя, чтобы пользователь мог взаимодействовать с данным объектом.

Human Machine Interface (HMI) — человеко-машинный интерфейс — термин, используемый для обозначения пользовательского интерфейса систем сбора данных и оперативного диспетчерского управления (SCADA).

hypertext — гипертекст — данные, содержащие перекрестные ссылки на ключевые элементы; такая организация данных позволяет пользователю выбирать собственный маршрут просмотра информации.

I

icon — пиктограмма — условное графическое изображение объекта или понятия, как правило, передающее его основные свойства. Сравн. *glyph*.

Image Map — сенсорная карта — интерактивный графический элемент Web-страницы, обеспечивающий навигацию по системе Web-страниц, а также визуальное представление ее структуры.

implicit selection — косвенный выбор — область выбора, которая формируется на основе логической связи между объектами или в результате выполнения некоторой операции, отличной от операции явного выбора.

inactive — неактивный — состояние объекта, когда он не находится в фокусе ввода.

inactive window — неактивное окно — окно, в котором в данный момент не работает пользователь. Неактивное окно обычно отличается от активного цветом полосы заголовка окна.

indeterminate — неопределенный; см. *mixed-value appearance*.

input focus — фокус ввода — позиция (область) на экране, в которой пользователь в данный момент времени выполняет ввод или редактирование данных.

input focus appearance — представление фокуса ввода — визуальный признак того, что данный объект или элемент управления находится в фокусе ввода

insertion point — точка вставки — позиция, в которую будет помещен вставляемый текст или графика.

inside-out activation — внешняя активизация — техника, которая позволяет пользователю непосредственно взаимодействовать с содержимым вложенного объекта OLE, не выполняя явно команду активизации.

interactive — интерактивность — эффект, при котором какие-либо действия пользователя вызывают соответствующую реакцию со стороны приложения, Web-страниц или другого электронного объекта.

Interface Builder (IB) — построитель интерфейса — специализированные инструментальные средства разработки пользовательского интерфейса.

interoperability — взаимодействие — возможность различных программных систем работать совместно за счет использования общих протоколов; основа существования Интернета.

J

jump — специальная форма связи между объектами, которая обеспечивает переход от одного объекта к другому (другое название — гиперссылка — *hyperlink*).

justified — выровненный — элемент (или группа элементов) интерфейса, определенным образом форматированный (выровненный) относительно других элементов.

L

label — метка — текст или графика, связанные с элементом управления, поясняющие его назначение.

link — связь — форма отношения между двумя объектами; при выполнении операции связывания создается так называемая *ссылка на объект*, посредством которой и реализуется связь.

link path — маршрут связи — форма описания ссылки на позицию источника связи; различают абсолютный и относительный маршруты связи.

list box — список — стандартный элемент GUI; отображает перечень доступных объектов или свойств объекта.

list view — модифицируемый список — стандартный элемент GUI, используемый в Windows-приложениях, который позволяет отображать перечень объектов в нескольких форматах (например, в виде крупных или мелких значков).

locale — локальный — набор языково-зависимых установок пользователя для форматирования информации (например, форматы представления астрономического времени, даты или валюты).

localization — локализация — процесс адаптации программного обеспечения для различных стран, языков или культур.

M

Man Machine Interface (MMI) — человеко-машинный интерфейс — термин, используемый для обозначения пользовательского интерфейса систем сбора данных и оперативного диспетчерского управления (SCADA).

marquee — см. *region selection*.

maximize — Развернуть — команда, обеспечивающее расширение окна до его максимального размера.

MDI — многодокументный интерфейс; см. *multiple document interface*.

menu — меню — список вариантов действий, представленных в текстовой или графической форме, из которых пользователь может выбрать только один.

menu bar — полоса меню — горизонтальная область в верхней части первичного окна (расположенная ниже полосы заголовка), содержащая основное меню.

menu button — кнопка меню — разновидность управляющей кнопки, при нажатии которой открывается (или закрывается, если было открыто) выпадающее меню.

menu item — пункт меню.

menu title — название (заголовок) меню — текстовая или графическая метка, которая идентифицирует конкретное меню. Для выпадающих меню заголовком

служит текстовая метка, расположенная в полосе меню; для иерархических меню заголовком является имя соответствующего родительского пункта меню.

message — сообщение — информация, которая не запрашивалась пользователем, но была выдана приложением или системой в ответ на действие пользователя или какого-либо внутреннего процесса.

message box — окно Сообщение — вторичное окно, которое содержит сообщение пользователю о конкретном условии, влияющем на продолжение работы.

minimize — Свернуть — команда, предназначенная для сворачивания (минимизации размера) окна.

mixed-value appearance — представление неопределенного значения — способ визуального представления элемента управления, когда он относится к разнородному выбору.

mnemonic — мнемоническая клавиша; см. *access key*.

modal — модальный — используемый с определенными 'ограничениями, зависящими от установленного режима; например, модальное вторичное окно ограничивает возможность взаимодействия пользователя с другими окнами.

mode — режим — конкретная форма взаимодействия, зачастую исключающая другие формы (способы) взаимодействия.

modeless — независимый — способ или форма взаимодействия, независимый по отношению к установленному режиму; например, независимое вторичное окно не влияет на взаимодействие пользователя с другими окнами.

modifier key — клавиша-модификатор — клавиша, при нажатии которой изменяется действие других клавиш.

mouse — мышь — устройство ввода, которое имеет одну или более кнопок, обеспечивающее взаимодействие пользователя с информацией, представленной на экране; данный термин используется также для обозначения других устройств указания (например, трекболов).

multiple document interface (MDI) — многодокументный интерфейс — техника управления окнами, при использовании которой содержимое документов отображается в так называемых дочерних окнах, расположенных в пределах основного (родительского) окна.

multiple selection list box — список множественного выбора — список, который поддерживает возможность одновременного раздельного выбора нескольких объектов.

N

native file format — собственный («исконный») формат файла — формат, который может быть распознан только той программой, которая создает файлы в этом формате.

navigation — навигация — 1) перемещение пользователя (точнее, фокуса ввода) между элементами интерфейса, например, переход из одного открытого окна в другое, либо из одного текстового поля в другое, расположенное на той же диалоговой панели; 2) перемещение пользователя между ресурсами Интернета, в том числе с одной Web-страницы на другую.

navigation key — клавиша навигации — клавиша, обеспечивающая выполнение навигации; к таким клавишам относятся, в частности, клавиши управления курсором, клавиша табуляции и т.д.

netiquette — сетевой этикет — правила поведения при работе в Интернете.

nondefault drag and drop — альтернативная реализация техники drag and drop — техника выполнения операций пересылки, при которой их интерпретация определяется командой, выбранной пользователем; эти команды включаются во всплывающее меню, отображаемое в той позиции, куда пересылается (перетаскивается) объект.

notification message — информационное сообщение — информация, предоставляемая приложением в ответ на ввод команды пользователем и описывающая результат или возможные последствия выполнения команды, либо состояние системы.

O

object — объект — понятие или компонент, идентифицируемый пользователем и который может отличаться от других объектов своими свойствами, допустимыми операциями и связями.

object-action paradigm — парадигма объектного взаимодействия — основная модель графического интерфейса пользователя, при использовании которой сначала идентифицируется объект, на который будет направлено воздействие, и только затем — команда, которая должна выполняться.

OLE (MS OLE — Object Linking and Embedding) — связывание и внедрение объектов — понятие, которое описывает технологию взаимодействия объектов, созданных с помощью различных приложений.

OLE embedded object — внедренный объект OLE — объект данных, физически находящийся в другом документе, но для которого сохраняется возможность его редактирования и выполнения других операций средствами приложения, с помощью которого он был создан.

OLE linked object — связанный объект OLE — объект, который представляет или обеспечивает доступ к другому объекту, находящемуся в том же или в другом контейнере.

OLE visual editing — визуальное редактирование — возможность редактировать вложенный объект OLE непосредственно в той позиции, где он расположен, не открывая для этого отдельное окно.

open appearance — открытие объекта — визуальное представление содержимого объекта, посредством открытия для объекта собственного окна.

operation — операция — общий термин для обозначения действий, которые *могут* быть выполнены над объектом.

option button — переключатель — стандартный элемент GUI, который позволяет пользователю выбирать из фиксированного набора взаимоисключающих вариантов только один (см. также *radio button*); сравн. *check box*.

option-set appearance — визуальное представление элемента управления, соответствующее состоянию «выбран».

outside-in activation — техника взаимодействия, которая требует пользователя выполнить явно команду активизации вложенного объекта OLE для работы с его содержимым.

P

palette window — окно Палитра — вторичное окно, которое используется для отображения панели инструментов или других наборов взаимосвязанных интерактивных элементов (например, графических примитивов); в каждый момент времени может быть выбран только один из них.

pane — подокно — одна из областей в разделенном окне.

parent window — родительское окно — первичное окно, которое обеспечивает управление дочерними окнами при использовании многодокументного интерфейса (МОД).

persistence — инерционность — один из принципов объектного подхода при реализации интерфейса; заключается в том, что текущее состояние объекта автоматически сохраняется до тех пор, пока пользователь не изменит его.

pixel — пиксел — наименьший неделимый элемент цифрового изображения.

point — точка — единица измерения, используемая для определения положения указателя над конкретным объектом или позицией (равна приблизительно 1/72 дюйма).

pointer — указатель — графический объект, отображаемый на экране с целью визуального указания позиции устройства выбора; см. *cursor*.

pop-up menu — всплывающее меню — меню, которое отображается в позиции выбранного объекта (другое название — контекстное (context) или сокращенное (shortcut) меню); такое меню содержит команды, определяемые свойствами выбранного объекта или позиции, в которой находится указатель.

pop-up window — всплывающее окно — вторичное окно, не плюющее полосы заголовка, которое отображается около объекта; как правило, используется для вывода контекстной информации об этом объекте.

pressed appearance — визуальное представление элемента управления (обычно кнопки), соответствующее состоянию «нажато».

primary window — первичное окно — окно, обладающее определенными свойствами (например, для него создается кнопка входа на Панели задач) и в котором, как правило, пользователь выполняет основные действия по работе с данными; первичное окно остается открытым на протяжении всего сеанса работы с приложением, хотя и может быть свернуто либо перекрыто другими окнами. См. также *secondary window*.

progress indicator control — индикатор процесса — стандартный элемент GUI, который отображает в графической форме процент завершения конкретного процесса.

project — Проект — техника управления окном или заданием, которая позволяет хранить все объекты, относящиеся к выполняемому заданию, в одном контейнере.

properties — атрибуты — свойства или характеристики объекта, которые определяют его состояние или поведение.

property inspector — инспектор свойств — средство динамического просмотра свойств текущего выбора (или выбранного объекта).

property page — страница свойств — группа свойств, отображаемых на одной вкладке панели свойств. См. также *property sheet*.

property sheet — панель свойств — вторичное окно, которое отображает свойства объекта; как правило, данное окно открывается по команде Property (Свойства).

Prototyping Tools (PT) — средства прототипирования — инструментальные средства проектирования пользовательского интерфейса.

push button — кнопка — см. *command button*.

R

radio button — переключатель — см. *option button*.

range selection — область выбора — см. *contiguous selection*.

recognition — распознавание — правильная интерпретация приложением манипуляций пользователя как операции определенного типа, либо распознавание вводимых им условных обозначений (символов).

reference Help — Справка — форма оперативной помощи пользователю, которая предполагает предоставление ему концептуальной и пояснительной информации. Сравн. *task-oriented Help* и *context-sensitive Help*.

refresh — восстановить — команда, позволяющая вернуться к предыдущим установкам параметров и значений свойств; обычно инициируется с помощью одноименной кнопки, расположенной на панели свойств.

region selection — выбор области — способ выбора, который основан на перемещении (с помощью мыши) границы области с целью включения в нее выбираемых объектов (синоним — *marquee*).

relationships — взаимосвязь — отношения, в которых состоит данный объект с другими объектами, образующими предметную область или рабочую среду.

reset — Сбросить — команда, приводящая к отмене установленных значений параметров или свойств; как правило, используется в панелях свойств.

resume — Продолжить — команда, обеспечивающая возобновление работы приложения, прерванной тем или иным событием; как правило, инициируется с помощью одноименной кнопки, отображаемой в окне предупреждающего сообщения, обусловленного этим событием.

rich-text box — многострочная текстовая область — стандартный элемент GUI, который отличается от обычного текстового поля тем, что для него поддерживается индивидуальный выбор шрифта и других параметров форматирования, а также печать содержимого.

S

SCADA (Supervisory Control And Data Acquisition system) — система сбора данных и оперативного диспетчерского управления — специализированные аппаратно-программные средства, обеспечивающие управление технологическим процессом в масштабе реального времени, а также инструментарий для разработки программного обеспечения таких систем.

scope — область — часть существующего выбора, логически независимая от других элементов выбора; например, выбор, сделанный в одном окне, обычно считается независимым по отношению к выбору, сделанному в любом другом окне.

scrap — след — форма визуального представления пиктограммы объекта, пересылаемого в другой контейнер.

scroll — прокрутка — перемещение объекта или информации, чтобы сделать видимой скрытую часть.

scroll arrow button — кнопка прокрутки (стрелка) — компонент полосы прокрутки, который при нажатии на него обеспечивает дискретное перемещение информации; направление перемещения определяется направлением стрелки.

scroll bar — полоса прокрутки — стандартный элемент GUI, который обеспечивает прокрутку.

scroll box — движок — компонент полосы прокрутки, который указывает относительную позицию (а также относительный размер) видимой части информации.

scroll bar shaft — чувствительная область — компонент полосы прокрутки, по которому перемещается движок. Щелчок мышью в чувствительной зоне (выше или ниже движка) приводит к перемещению информации в соответствующем направлении на один экран.

secondary window — вторичное окно — окно, которое обеспечивает предоставление или ввод дополнительной информации, связанной с объектами, отображенными в первичном окне.

select — выбор — идентификация одного или более объектов, над которыми должно быть выполнено то или иное действие.

selection — область выбора, выбор — объект или множество объектов, которые были выбраны.

selection appearance — способ визуального представления выбранного объекта.

selection handle — дополнительный графический элемент интерфейса, который обеспечивает выполнение операций прямого манипулирования над выбранным объектом, таких как перемещение или масштабирование.

separator — разделитель — графический элемент, который обеспечивает визуальное разделение пунктов меню на подгруппы.

server — сервер — программа, предоставляющая услуги другой программе, например, для работы с внедренным объектом OLE.

shell — оболочка (графическая оболочка) — общий термин, который используется для обозначения графической среды, на базе которой реализуется GUI.

shortcut — сокращенный — общий термин, который характеризует упрощенную, ускоренную технику выполнения команд или ввода данных по сравнению с обычным (стандартным) методом.

shortcut icon — ярлык — специальная пиктограмма, которая визуально представляет ссылку на некоторый объект и обеспечивает доступ пользователя к этому объекту.

shortcut key — клавиша-акселератор, горячая клавиша — клавиатурная клавиша (или комбинация клавиш), которая инициирует конкретную команду.

shortcut menu — см. *pop-up menu*.

single selection list box — список единичного выбора — список, который поддерживает выбор только единственного пункта из предлагаемого перечня.

size grip — регулятор — специальный элемент, отображаемый на соединении горизонтальной и вертикальной полос прокрутки или на правой оконечности строки состояния, который обеспечивает пропорциональное изменение размера окна по обоим измерениям.

slider — ползунковый регулятор, ползунок — стандартный элемент GUI, предназначенный для установки значения непрерывных величин.

spin box — дискретная текстовая область — элемент GUI, который позволяет пользователю устанавливать одно из заданных значений дискретной величины, изменяющейся в некоторой ограниченной области; такие значения могут быть представлены в символьной форме (словами или строками).

split bar — полоса разделения — визуальная граница между подокнами.

split box — разделитель — специальный элемент управления, добавляемый к окну, которое может быть разделено на подокна; позволяет пользователю разделять окно или регулировать размер подокон.

status bar — строка состояния — область, которая обеспечивает вывод дополнительной информации о состоянии объектов или процессов, представленных в окне; обычно отображается в нижней части окна.

stop — Стоп — команда, останавливающая процесс или действие (как правило, без восстановления исходного состояния). Сравни. *cancel*

submenu — субменю — см. *cascading menu*.

Submit — кнопка — элемент формы в виде кнопки, при нажатии которой инициируется процесс передачи данных, введенных пользователем, Web-серверу.

T

tab control — этикетка вкладки — стандартный элемент GUI, на котором может отображаться текстовая или графическая метка вкладки и который обеспечивает навигацию между вкладками.

Table — таблица — формат представления данных на Web-странице, позволяющий отображать их в табулированном виде или в виде отдельных столбцов.

taskbar — Панель задач — панель инструментов, входящая в состав Рабочего стола; Панель задач содержит кнопку Пуск (Start), кнопки входа для каждого из открытых первичных окон и область сообщений (tray).

task-oriented Help — помощь, управляемая заданием — форма контекстной помощи пользователю, реализованная в виде инструкций по выполнению отдельных шагов (этапов) конкретного задания. Сравни. *context-sensitive Help* и *reference Help*.

template — шаблон — объект, который автоматизирует создание новых объектов.

text box — текстовое поле — стандартный элемент GUI (называемый также областью или полем редактирования — *edit field*), с помощью которого пользователь может ввести и отредактировать текст; может использоваться в комбинации с другими элементами GUI (например, со списком), а также для вывода на экран неизменяемой текстовой информации.

thumbnail — миниатюра — термин, '(пользуемый для обозначения уменьшенной (и «ухудшенной») копии графического изображения, которая используется в качестве ссылки на оригинал; применение миниатюр призвано сократить время на загрузку Web-страниц.

title bar — полоса заголовка — горизонтальная область в верхней части окна, которая идентифицирует окно; полоса заголовка выступает также в качестве интерактивной зоны для перемещения окна и вызова всплывающего меню окна.

toggle key — клавиша-переключатель — клавиша клавиатуры, которая включает или выключает определенный режим.

toolbar — панель инструментов — стандартный элемент GUI, который обеспечивает компактное представление набора взаимосвязанных элементов управления.

toolbar button — инструментальная кнопка — разновидность кнопки управления, включаемая в состав панели инструментов (или строки состояния).

tooltip — всплывающая подсказка — стандартный элемент GUI; представляет собой небольшое всплывающее окно, которое содержит пояснительный текст (например, назначение кнопки панели инструментов).

transfer appearance — форма визуальной обратной связи с пользователем, отображаемой в течение выполнения операции пересылки.

transaction — транзакция — единичное действие по изменению объекта (или отдельного его свойства).

tree control — модифицируемое дерево — стандартный элемент GUI, который обеспечивает визуальное представление набора иерархически связанных объектов с возможностью изменения степени детализации отображаемой информации.

type — значение определяется контекстом: 1) ввод символов с клавиатуры. 2) тип (объекта) — совокупность свойств и особенностей поведения объекта

U

unavailable — недоступно — состояние объекта или элемента управления, при котором пользователь не может использовать его по функциональному назначению.

unavailable appearance — способ визуального представления объекта или элемента управления, находящегося в состоянии «недоступно».

undo — отменить — команда, позволяющая отменить результат выполнения транзакции.

unfold button — специальная кнопка, используемая для дополнения (расширения) вторичного окна.

usability — удобство — показатель качества программного продукта, отражающий субъективную удовлетворенность пользователя уровнем пользовательского интерфейса.

User Centered Design (UCD) — разработка, ориентированная на пользователя — технология разработки программного обеспечения, при которой программное обеспечение проектируется и реализуется с учетом требований и характеристик потенциальных пользователей.

user interface (UI) — пользовательский интерфейс, интерфейс пользователя — совокупность правил, методов и программно-аппаратных средств, обеспечивающих взаимодействие пользователя с компьютером.

User Interface Management System (UIMS) —система управления пользовательским интерфейсом интегрированный набор средств, предназначенных для создания и управления различными интерфейсами пользователя. Основной концепцией UIMS является идея разделения интерфейса и прикладной программы (точнее, ее функционального наполнения). Как правило, UIMS состоит из двух частей: одна обеспечивает разработку интерфейса, а вторая — управление пользовательским интерфейсом в процессе его работы с приложением.

user option — режим пользователя — вариант представления или взаимодействия, выбираемый пользователем в процессе работы с приложением, и обеспеченный разработчиком.

W

warning message — предупреждающее сообщение — предупреждение пользователя о возникновении ситуации, которая требует от него выбора одного из возможных вариантов последующих действий приложения или системы; используется в тех случаях, когда предстоящая операция является потенциально опасной или имеет необратимые последствия. Сообщение может быть сформулировано в виде вопроса, например «Сохранить внесенные изменения?»

well control — коллекция — интерактивная панель, содержащая набор образцов (например, цветов, типов линий и т.п.), которая используется для выбора пользователем одного из них.

white space — белое пространство — фоновая область окна.

widget — деталь, заготовка — термин, используемый для обозначения базовых элементов (объектов), на основе которых строится графический интерфейс пользователя в системе XWindow; примерами таких базовых элементов являются полосы прокрутки, кнопки, меню.

window — окно — специальная область физического экрана, с помощью которой пользователь имеет возможность получить визуальное представление определенного аспекта решаемой задачи. Другими словами, окно является средством просмотра и редактирования информации, а также отображения содержимого и свойств объектов. Окна могут использоваться также для вывода на экран значений параметров, результатов выполнения команд, наборов инструментов и сообщений, информирующих пользователя о конкретной ситуации.

wizard — Мастер — форма помощи пользователю, которая обеспечивает автоматизацию выполнения задания посредством ведения диалога с пользователем.

workbook — Рабочая книга — техника управления окнами или заданиями, при использовании которой для представления информации применяется система взаимосвязанных страниц, отображаемых в пределах одного первичного окна.

workspace — Рабочая область — техника управления окнами или заданиями, при использовании которой все объекты, относящиеся к выполняемому заданию, хранятся в одном контейнере, содержимое которого может быть отображено в родительском окне.

WYSIWYG, What You See Is What You Get — «что вы видите, то и получите» — термин для обозначения технологии, обеспечивающей идентичность визуального представления информации (документа) как на этапе разработки, так и на этапе использования; например, электронный документ, созданный с помощью редактора Word, выглядит так же, как и его бумажная копия; аналогично, визуальные HTML-редакторы позволяют Web-дизайнеру представить создаваемую страницу в том же виде, в котором ее сможет просматривать пользователь с помощью браузера.

12. ЛИТЕРАТУРА

1. *Sidney L.Smith, Jane N.Mosier*. Guidelines for Designing User Interface Software // ESD-TR-83-122, MITRE Corporation, Bedford, MA (August 1986)
2. *Уаттс Р.* ЭВМ и непрофессиональные пользователи: Организация взаимодействия: Пер. с англ. - М.: Радио и связь, 1989.
3. *Коутс Р., Влейминк И.* Интерфейс «человек - компьютер»: Пер. с англ. - М.: Мир, 1990
4. *Донской М.* Последний фут. //PCWeek/Russian Edition, N 1, 1999, с. 5.
5. *Буч Г.* Объектно-ориентированное проектирование с примерами применения: Пер. с англ.- М.: Конкорд, 1992.
6. *Жданов А.* Операционные системы реального времени. // PCWeek/Russian Edition, N 8, 1999, с. 17.
7. *Фокин Ю.Г.* Оператор - технические средства: обеспечение надежности. - М.:Воениздат. 1985.
8. Информационно-управляющие человеко-машинные системы: Исследование, проектирование, испытания: Справочник/ Под общ. Ред. А.И.Губинского и В.Г. Евграфова. - М.: Машиностроение, 1993.
9. *Brad A. Myers*. User Interface Software Tools /ACM Transactions on Computer-Human Interaction. N1, 1995. т.2, с. 64-103.
10. *Бобровский С.* Упрощайте корпоративные узлы. // PCWeek/Russian Edition, N 40, 1998,с.15.
11. *Тузов В.А.* Языки представления знаний. Л.: Издательство ЛГУ. 1990.
12. *Бетоны Х.* Voice Xpress вас слушает. // PCWeek/Russian Edition, N 23, 1998, с. 21.
13. *Титтел Э., Сандерс К.* и др. Создание VRML-миров. -К. BHV, 1997.
14. *Денинг В., ЭсигГ., Маас С.* Диалоговые системы «Человек — ЭВМ». Адаптация к требованиям пользователя: Пер. с англ. — Мир, 1984.
15. Проектирование пользовательского интерфейса на персональных компьютерах. Стандарт Фирмы IBM: Пер. с англ. — Вильнюс: DBS LTD, 1992.
16. *Минаси М.* Графический интерфейс пользователя: секреты проектирования: Пер. с англ.—М.: Мир, 1996.