**РиСПСиИТ**

**Лекция 5**

**Тема «Разработка пользовательских интерфейсов»**

Связующим звеном при работе с компьютером является пользовательский интерфейс. Для этого раньше использовалась командная строка. Сейчас, как правило, интерфейс является графическим. Он содержит изображения объектов: форм, управляющих элементов, текстовых сообщений и др.

Важным в создании пользовательских интерфейсов является использование поведенческих шаблонов пользователей. Они описывают поведение людей в схожих ситуациях, их ожидания.

**Использование поведенческих шаблонов в конструировании интерфейсов**

Интерфейс, поддерживающий шаблоны, поможет пользователям достичь поставленных целей намного более эффективно, чем интерфейс, не учитывающий их. Шаблоны эти относятся не только к интерфейсу. Однако, будучи дизайнером интерфейсов или дизайнером взаимодействия, вы должны считаться с шаблонами точно так же, как все остальные участники команды. Возможно, вы окажетесь в наилучшей позиции, чтобы выступить в защиту пользователей. Примеры поведенческих шаблонов:

• Шаблон 1. **Safe Exploration** (Безопасное исследование).

• Шаблон 2. **Instant Gratification** (Мгновенное вознаграждение).

• Шаблон 3*.* **Changes in Midstream** (Изменения на полпути).

• Шаблон 4. **Satisficing** (Разумная достаточность).

• Шаблон 5. **Deferred Choices** (Отложенный выбор).

• Шаблон 6. **Incremental Construction** (Пошаговое построение).

• Шаблон 7. **Habituation** (Привыкание).

* Шаблон 8. **Spatial Memory** (Пространственная память).
* Шаблон 9. **Prospective Memory** (Проспективная память).
* Шаблон 10. **Streamlined Repetition** (Организованное повторение).
* Шаблон 11. **Keyboard Only** (Только клавиатура).
* Шаблон 12. **Other People's Advice** (Советы других людей).

Шаблон 1. **Safe Exploration (Безопасное исследование)**

*«Позвольте мне исследовать программу, не теряясь в ней и не попадая в непри­ятности».*

Когда пользователь понимает, что у него есть возможность исследовать интер­фейс без ужасающих последствий, то, скорее всего, он в итоге узнает больше и по­лучит более позитивные впечатления о приложении, чем тот, кто на это не решил­ся. Хорошее программное обеспечение позволяет людям пробовать неизвестные функции и возвращать систему в исходное состояние, снова пробовать что-то но­вое и так далее, без всякого стресса.

«Ужасающие последствия» даже не обязаны быть такими уж страшными. Простой досады бывает достаточно, чтобы отпугнуть пользователя от доброволь­ного исследования. Закрывание всплывающих окон, повторный ввод ошибочно стертых данных, поспешное отключение звука на ноутбуке, когда веб-сайт неожи­данно начинает воспроизведение громкой музыки — все это может обескуражить. Создавая практически любой тип интерфейса, подготовьте для пользователя безопасные способы его изучения, чтобы пользователи могли поэкспериментиро­вать, не опасаясь никаких последствий.

Шаблон 2. Instant Gratification (мгновенное вознаграждение)

«я хочу *сделать это прямо сейчас, а не потом».*

Людям хочется видеть мгновенные результаты своих действий – это заложено в человеческой природе. Если кто-то начинает использовать приложение и получает «успешный опыт» в первые несколько секунд, это доставляем удовольствие. С большой вероятностью пользователь продолжит работу с этим приложением, даже если позже ему станет сложнее добиваться поставленных целей. Он будет чувствовать себя спокойнее в этом приложении и будет более уверен, чем если бы ему пришлось потратить много времени, чтобы разобраться, что к чему.

Мгновенное вознаграждение необходимо в различных областях дизайна. Можно предположить, что в первую очередь будет делать новый пользователь Приложения, и разработать пользовательский интерфейс так, чтобы это первое действие оказалось потрясающе простым. Если целью пользователя является, например, рисование объекта, то можно сразу же открыть новый холст и поместить рядом с ним палитру. Если пользователю необходимо выполнить какую-либо задачу, то подскажите ему типичную отправную точку.

Особенности пользовательского интерфейса Mathcad.

В книге «Разработка пользовательских интерфейсов» даны описания других поведенческих шаблонов.

**Организация экрана: компоновка элементов страницы**

Здесь также используются различные шаблоны. Пример:

Шаблон 33. **Center Stage (Центральная сцена)**

Поместите самую важную часть пользовательского интерфейса в самый большой подраздел страницы или окна; объедините дополнительные инструменты и содержимое на небольших панелях, окружающих сцену (рис. 1). Примеры: Delphi, Word, Excel и др.



Рис. 1Форма, содержащая центральную сцену

Использовать, когда...

Основная задача страницы — продемонстрировать пользователю логически связанное содержимое, позволить ему отредактировать документ или выполнить оп­ределенную задачу. «Центральную сцену» можно использовать в большинстве приложений — сюда входят текстовые таблицы и электронные расчетные таблицы, формы, веб-страницы и графические редакторы.

**Почему**

Взгляд пользователя необходимо сразу же перевести к началу наиболее важной информации (или задачи), не позволяя ему растерянно бродить по странице. Четко выраженный центральный элемент приковывает внимание. Так же как первая строка в новостной статье определяет ее сущность и назначение, элемент, находящийся в центральной сцене, определяет назначение пользовательского интерфейса.

Как

Создайте визуальную иерархию, в которой «центральная сцена» будет домини­ровать над всем остальным. (Обсуждение визуальной иерархии см. во введении к этой главе.) При разработке интерфейса на основе шаблона **Center Stage** (Цен­тральная сцена) учитывайте следующие факторы (хотя ни один из них не являет­ся абсолютной необходимостью).

**Размер**

Содержимое «центральной сцены» должно быть как минимум в два раза шире всего, что находится в левом и правом полях, и в два раза выше того, что за­нимает верхнее и нижнее поля. (Пользователю можно разрешить менять ее размер, но при первом открытии интерфейс должен выглядеть именно так.)

**Цвет**

Используйте цвет, контрастирующий с информацией, находящейся на полях. В пользовательских интерфейсах настольных приложений белый цвет хорошо контрастирует со стандартным серым цветом Windows, особенно в таблицах и деревьях. Также белый цвет зачастую хорошо работает на веб-страницах, так как в рекламных объявлениях и навигационных панелях обычно применяют другие цвета, отличные от фонового; кроме того, считается, что пользователи веб-сайтов «приучены» искать обычный текст на белом фоне.

**Заголовки**

Крупные заголовки являются точками фокусировки и могут привлекать вни­мание пользователя к верхней части «центральной сцены». Конечно же, этот принцип работает и в печатных средствах информации. Подробнее о заголов­ках и точках фокусировки рассказывается во введении к этой главе и в описа­нии шаблона **Titled Sections** (Именованные разделы).

**Контекст**

Что пользователь ожидает увидеть, когда он открывает страницу: графиче­ский редактор, длинную текстовую статью, карту или дерево файловой систе­мы? Воспользуйтесь тем, что знаете, чего он ждет — поместите этот объект на центральную сцену и сделайте его легко узнаваемым. Пользователь будет ис­кать именно этот объект; ожидания — это козырь среди всех прочих правил, описывающих визуальное восприятие. (Однако, это не означает, что можно на­меренно выводить пользователей из себя, пряча то, что они ищут. На некото­рых веб-сайтах основное содержимое находится на странице так низко, что оказывается под нижней границей небольших окон, и для того чтобы добрать­ся до него, приходится прокручивать окно. Это чистой воды садизм.)

Другим примером может быть шаблон **Именованные разделы**. В нем присутствуют равноправные разделы. Им можно дать нумерацию или другие обозначения и расположить рядом друг с другом. Например, районы или области на географических картах.

 Имеются также другие шаблоны, применяемые при компоновке экрана.

Шаблон 35. **Card Stack (Пачка карточек**)

Поместите разделы содержимого на отдельные панели (или карточки). Затем соберите их в пачку так, чтобы всегда была видна только одна карточка. Используйте вкладки или другие средства, чтобы предоставить пользователю доступ ко всем панелям. См. рис. 2.

Использовать, **когда...**

На странице слишком много материала. Множество элементов управления или блоков текста распределены по всему интерфейсу и их невозможно организовать в единую структуру (такую, как **Property Sheet** (Таблица свойств)); внимание пользователя рассеивается. Содержимое можно сгруппировать в именованные разделы; (шаблон **Titled Sections** (Именованные разделы)), но в этом случае они будут слишком большими, чтобы их все можно было одновременно разместить на странице.



Рис.2 Пачка карточек

Формы и элементы управления

Рано или поздно разрабатываемое вами программное обеспечение обязательно попросит пользователя ответить на какой-либо вопрос. Это может произойти уже в первые несколько минут взаимодействия между ними: куда следует установить данное приложение? Под каким именем вы будете входить в систему или по каким словам выполнить поиск?

Такие типы взаимодействия реализовать проще всего. Каждый знает, как работать с текстовыми полями, флажками и комбинированными полями. Эти эле­менты управления вводом часто становятся первыми элементами интерфейса, которые новички-дизайнеры используют при построении своих первых пользовательских интерфейсов или веб-сайтов.

Однако создать неуклюжую форму взаимодействия пользователя и приложе­ния совсем несложно. Например, приложение может задать такой вопрос: для какого района вы хотите получить прогноз погоды? Пользователь задумывается,

-нужно ли ему указать город, страну или почтовый индекс?

А аббревиатуры ли достаточно?

А что будет, если допустить орфографическую ошибку?

А что, если спросить город, неизвестный системе?

А нет ли здесь карты, чтобы выбрать место щелчком мыши?

И почему система не могла запомнить район, который он выбирал вчера?

В этой главе обсуждаются способы, позволяющие избавиться от этих проблем. Шаблоны, техники и элементы управления, о которых здесь пойдет речь, в основном применяются для дизайна форм, где под «формой» понимается простая последовательность пар вопросов и ответов. Однако они будут полезными и в других контекстах, например, когда речь идет о единичных элементах управления на веб-страницах или о панелях инструментов в приложениях. Дизайн ввода данных и дизайн-форм — это ключевые навыки для дизайн-взаимодействия, так как их можно применять в любых отраслях дизайна и на любых платформах.

**Основы дизайна форм**

Для начала я перечислю несколько принципов, о которых следует помнить при создании форм и средств пользовательского ввода данных.

**Удостоверьтесь, что пользователь понимает, какие данные у него запраши­вают и зачем.**

Здесь все зависит исключительно от содержимого, и любое обобщение может оказаться бессмысленным, но я попробую. Для написания меток нужно ис­пользовать слова, которые понимает ваша целевая аудитория, — простой язык для новичков и редких пользователей и тщательно продуманный профессио­нальный жаргон или специализированный словарь для экспертов в данной предметной области. В длинной или скучной форме потратьте немного про­странства на объяснение пользователю, зачем вам нужна вся эта информация, которая запрашивается в форме. Если вы помещаете на панель инструментов (или в другое место, также ограниченное в пространстве) элемент управле­ния, назначение которого неочевидно, добавьте описательную всплывающую подсказку или контекстно-зависимую справку другого типа, чтобы сообщить пользователю, что делает данный элемент.

**Если возможно, вообще не задавайте вопросы.**

Задавая пользователю вопрос, особенно если это происходит в процессе вы­полнения какой-либо другой задачи, вы оказываете на него давление. Вы за­ставляете его прервать поток размышлений и обратить внимание на то, чего он совершенно не ожидал. Даже в самых благоприятных ситуациях ввод дан­ных в текстовые поля — это не то, чем люди предпочли бы заниматься для соб­ственного развлечения.

Можно ли заранее заполнить элементы управления вводом известной или предсказуемой информацией, как рекомендует шаблон **Autocompletion** (Автозаполнение)?

Можно ли предложить разумное значе­ние по умолчанию, которое снимет задачу выбора с 80 % ваших пользовате­лей?

Можно ли не задавать вопросы?

В этом принципе есть одно важное исключение — безопасность. Иногда элемен­ты управления вводом используются для идентификации пользователя; напри­мер, приложение запрашивает пароль или номер кредитной карты. Очевидно, что совершенно не нужно пытаться обманывать такие механизмы безопасно­сти, небрежно заполняя поля ввода данных конфиденциальной информацией.

**Знания «из внешнего мира» зачастую точнее, чем знания «в голове»1.**

Не стоит ожидать от людей, что они с легкостью будут запоминать различ­ные списки и перечисления. Если вы просите пользователя сделать выбор из

1 *Согласно концепции американского психолога Дональда Нормана, знания, которыми мы пользуемся автоматически (например, слепая печать), относятся к «знаниям в голове», а знания, которые мы получаем извне, чтобы осуществить какое-то малопонятное дейст­вие, относятся к знаниям «из внешнего мира» (человек, увидевший клавиатуру впервые, будет искать надпись на каждой клавише, чтобы понять, какую букву она воспроизведет при нажатии).*

предопределенного набора элементов, то предоставьте ему этот набор или список, чтобы пользователь мог без труда просмотреть его содержимое. Рас­пространенным исключением может быть поле «штат или область», часто необходимое при вводе адреса для доставки товара. Очевидно, что большинство людей в состоянии запомнить названия нужных им регионов. Раскрываю­щиеся списки, комбинированные поля, простые списки и прочие подобные элементы управления дают пользователям возможность просматривать пере­числения элементов. А когда элементам сопутствуют визуальные объекты, на­пример изображения, схемы и цвета, то для их отображения можно использовать шаблон **Illustrated Choices** (Иллюстрированный выбор).

Аналогично, если вы просите пользователя ввести данные, отформатирован­ные особым образом, например дату или номер кредитной карты, то нужно *со­общить* ему с помощью подсказки, какой именно формат ожидается в каждом конкретном случае. Даже если пользователь уже встречался с пользователь­ским интерфейсом вашей программы ранее, он может не помнить, что именно здесь требуется, и ненавязчивое напоминание будет очень кстати. Этой целью служат шаблоны:

**Good Defaults** (Хорошие варианты по умолчанию),

**Structure\* Format** (Структурированный формат) и

**Input Hints** (Подсказки при вводе).

Шаблон **Autoconnection** (Автозаполнение) идет на шаг впереди, показывал пользователю допустимый формат вводимых данных или напоминая ему, что он вводил в том же месте раньше. Пример, интеллектуальный помощник текстового редактора Delphi

**Остерегайтесь буквального воспроизведения программной модели, лежащей в основе приложения.**

Многие формы создаются специально для редактирования записей в базе данных или для редактирования объектов в объектно-ориентированном языке программирования. Имея на руках подобную структуру данных, очень легко разработать форму для ее заполнения. Каждый элемент структуры получает, во-первых, метку и, во-вторых, элемент управления (или набор сгруппированных элементов управления). Они сортируются в разумном порядке, перечисляются сверху вниз, и форма готова, не так ли?

Не совсем. Дизайн, буквально отражающий структуру данных, работает довольно хорошо, но в итоге интерфейс получается либо утилитарный и скуч­ный, либо слишком сложный. Что, если элементы структуры не соответствуют тому, какие данные пользователь ожидает ввести? А что, если структура содержит, скажем, тридцать элементов? В некоторых контекстах, например, в таблицах свойств в программных средах, допустимо отображать структуру так, как они фактически реализованы, — в этом и заключается смысл. Но и для всего остального лучше подходит более элегантное и ориентированное на пользователя представление.

Итак, проблема заключается в следующем: можно ли сделать форму менее обременительной, сыграв на зависимостях между элементами структуры, знакомых графических конструкциях (например, этикетках с адресом), несформу­лированных предположениях или сведениях о пользователе, полученных при предыдущем взаимодействии с ним? Можно ли упростить проблему так, чтобы ее можно было решить прямой манипуляцией, например, путем перетаскивания объектов по экрану? Проявляйте фантазию!

**Всегда тестируйте удобство использования.**

Вследствие определенных причин, когда речь идет о формах для ввода дан­ных, разработчики и пользователи часто делают принципиально разные пред­положения относительно терминологии, возможных ответов, навязчивости и вопросов, связанных с контекстом использования. Об этом уже говорилось рань­ше и повторим еще раз: всегда тестируйте удобство использования, даже если вы уверены, что у вас получился хороший дизайн. Это даст вам подкреплен­ные реальным опытом свидетельства того, что работает, а что нет в вашей кон­кретной ситуации.

**Выбор элементов управления влияет на то, какие вопросы будет ожидать пользователь, так что подходите к этому с умом.**

Переключатель предполагает выбор одного элемента из группы, а текстовое поле из одной строки подталкивает к короткому, но довольно свободному по форме ответу. Сознательно или нет, пользователи основываются на физиче­ской форме элемента управления — его типе, размере и т. д., — когда размыш­ляют о том, что у них спрашивают, и их ожидания формируются соответст­вующим образом. Если вы используете текстовое поле для запроса числа, то пользователи полагают, что допустимы любые значения; когда они вводят 12, а вы удивляете их сообщением «Допустимый диапазон чисел — от 1 до 10». Это выбивает почву из-под ног. Ползунок или счетчик был бы в данной ситуа­ции намного лучшим решением.

В следующем разделе представлен список возможных элементов управления для разнообразных типов пользовательского ввода. Вы сами или разработчи­ки, с которыми вы сотрудничаете, должны принимать решения относительно семантики каждого вопроса. Это бинарное значение? Дата или время? Выбор одного элемента из нескольких? Выбор нескольких элементов из многих дос­тупных? Допускающее вариации, но требующее проверки значение? Выясни­те это, а затем выберите элемент управления, основываясь на ограничениях конкретного дизайна.

Выбор элементов управления

Далее описаны элементы управления и шаблоны для различных типов информа­ции, которую вы, вероятно, будете запрашивать у пользователей, например для числовых значений или списка, допускающего выбор одного элемента. Естест­венно, ни в коем случае этот набор нельзя считать полным. В действительности вы можете придумать множество других элементов управления. Однако перечис­ленные здесь типы данных довольно распространены, а элементы управления прекрасно подходят для создания понятных и удобных интерфейсов.

При выборе среди возможных элементов управления для каждого типа ин­формации основывайтесь на следующих факторах.

**Доступное пространство**

Одни элементы управления занимают много экранного пространства, другие могут быть меньше по размеру, но сложнее в использовании, чем более крупные. В коротких формах на веб-страницах можно тратить экранное пространств: на переключатели или иллюстрированные списки. Но в сложных приложениях обычно возникает необходимость упаковать как можно больше содержимого в как можно меньшее пространство. Панели инструментов и таблицы свойств (см. шаблон **Property Sheet** Таблица свойств) накладывают осо­бенно сильные ограничения, так как чаще всего их высота допускает только одну строку текста, и по ширине они также обычно бывают небольшими.

**Умение пользователя обращаться с компьютером в целом**

Текстовые поля знакомы практически всем пользователям любых приложе­ний, которые вы можете разрабатывать, но не у каждого получится с легкостью разобраться в использовании ползунка с двумя бегунками. Если на то пошло, многие случайные пользователи компьютера также не имеют представления, как обращаться с окном списка, допускающим множественный выбор.

**Опыт пользователя в предметной области**

Текстовое поле представляет собой прекрасный выбор элемента управления, если все пользователи знают, что, предположим, в данном месте допустимы только значения от 1 до 10 и от 20 до 30. Новички могут сделать ошибку, но если они составляют очень малую часть пользовательской аудитории (и если контекст достаточно прост для изучения), то можно не обращать на это внимания — крохотное текстовое поле все равно может быть намного удобнее использовать, чем большой набор взаимосвязанных элементов управления.

**Ожидания, сформированные другими приложениями**

Выделение текста жирным шрифтом, подчеркнутым шрифтом или курсивом, как правило, обозначено на кнопках с соответствующими пиктограммами. Было бы странно, если бы в вашем приложении это задавалось переключателем.

**Доступная технология**

На момент написания этой главы язык HTML предоставляет очень небольшой набор элементов управления, которые используются в настольных приложениях: текстовые поля, переключатели и флажки, прокручиваемые и раскрывающиеся списки, допускающие ввод значений.

Коммерческие средства для разработки графических интерфейсов пользователя и инструменты с открытым кодом обеспечивают дизайнера более широким ассортиментом элементов управления. Набор предлагаемых ими элементов управления может варьироваться, но большинство таких инструментов допускают программное расширение, благодаря чему можно создавать специальные элементы управления для особых ситуаций.

В следующих разделах перечисляются разнообразные варианты элементов управления для четырех наиболее распространенных сценариев пользовательского ввода: списки элементов, текст, числа, дата и время.

**Списки элементов**

Огромный диапазон знакомых элементов управления позволяет пользователям выбирать элементы или варианты из списков. Ваш выбор элемента управления зависит от того, сколько элементов или вариантов разрешается выбирать пользователю (один или много), а также от числа элементов в списке (два, несколько или много).

Элементы управления для выбора одного из двух вариантов (бинарный выбор)

**Флажок**

* Преимущества: простой, занимает мало места.
* Недостатки: недвусмысленно определяет только один вариант, поэтому его противоположность остается неявной и несформулированной. В результате пользователь может не понять, что означает сброшенный флажок.

**Два переключатели**

• Преимущества: оба варианта сформулированы и видны на
экране.

• Недостатки: занимают vбольше места.

» Преимущества: оба варианта сформулированы; небольшой и предсказуемый объем занимаемого пространства; просто» расширение при необходимости добавления более чем дву; кяпиантов.

 

много места.

**Раскрывающийся список с двумя вариантами выбора**

В Delphi это комбинированный список.

 Преимущества: оба варианта сформулированы. Мало места, возможность увеличения количества вариантов.

Недостатки: одновременно на экране виден только один вариант; требует определенной сноровки.

**«Залипающая» кнопка-переключатель**

• Преимущества: те же, что и у флажка; традиционный эле­мент управления, занимает мало экранного пространства.

• Недостатки: те же, что и у флажка; помимо этого, в отличие от флажка, не воспринимается в качестве стандартного элемента управления для выбора текстовых значений.



**меню с двумя элементами, представляющими собой переключатели**

• Преимущества: занимает очень мало места в главном поль­зовательском интерфейсе, так как находится в полосе ме­ню или всплывающем меню.

• Недостатки: всплывающие меню бывает трудно обнаружить; требует боль­шой сноровки.

**Элементы управления для выбора одного из N элементов, когда число N невелико**



**N переключателей**

* Преимущества: все варианты выбора всегда видны на экране.
* Недостатки: занимает много места.



**Раскрывающийся список из N элементов**

* Преимущества: занимает мало места.
* Недостатки: одновременно виден только один вариант выбора, за исключением ситуации, когда список открыт; требует
определенной сноровки.

**Набор из N взаимоисключающих пиктографических кнопок-переключателей**

* Преимущества: занимает мало места; видны все варианты выбора.
* Недостатки: значки могут быть сложными для понимания и требовать всплывающих подсказок; пользователи могут не осознавать, что варианты выбора взаимоисключающие.

**Меню из N элементов, представляющих собой переключатели**

* Преимущества: занимает очень мало места в главном пользовательском интерфейсе, так как находится в полосе меню или всплывающем меню; видны все варианты выбора.
* Недостатки: всплывающие меню бывает трудно обнаружить; требует большой сноровки.

**Список или таблица, допускающие единственный выбор**

* Преимущества: видны несколько вариантов выбора; рамку можно сделать совсем небольшой, оставив на виду только три элемента.
* Недостатки: занимает больше места, чем раскрывающийся список или счетчик.

**Счетчик**

* Преимущества: занимает мало места.

Недостатки: одновременно виден только один вариант выбора; требует большой сноровки; незнаком неподготовленным пользователям компьютера. Обычно лучше использовать раскрывающийся список.

**Дата и время**

Из-за множества возможных форматов и проблем с обозначением даты в разных странах принять введенную пользователем дату и время может быть довольно непросто. Варианты ввода следующие:

**Текстовое поле с «великодушным форматом»**

* Преимущества: визуально простое; допускает большой диапазон форматов и типов даты; доступ с клавиатуры. 03/12/2008
* Недостатки: по внешнему виду элемента управления нельзя точно определить ожидаемый формат данных, из-за чего пользователь может временно войти в замешательство; требует тщательной проверки по факту ввода.

**Текстовое поле со структурированным форматом**

Три однострочных редактора с комментариями.

•Преимущества: ожидаемый формат легко определить,

исходя из внешнего вида элемента управления.

•Недостатки: может занимать больше места; не допускает отклонений от указанного формата, даже если это необходимо.

**Элемент управления в форме календаря или часов**

* Преимущества: очевидная метафора; ввод ограничивается только допустимыми значениями.
* Недостатки: занимает много места; может не поддерживать доступ с клавиатуры.

**Раскрывающийся селектор с элементом управления в форме календаря или часов**

* Преимущества: сочетает преимущества текстового поля и календаря; занимает мало места.

Недостатки: сложное взаимодействие; требует сноровки для выбора значений в раскрывающейся части

**Шаблнон 73. Autocomplection (Автозаполнение)**

Автозаполнение часто встречается в текстовых редакторах и пользовательских интерфейсах на основе командной строки. Когда пользователь вводит команды или фразы, приложение предлагает варианты для завершения фразы.[ Очень хорошо подходят для этого редакторы программного кода и оболочки операционных систем, так как используемый язык небольшой и предсказуемый! (в противоположность человеческому языку, например английскому); то есть спрогнозировать, что пользователь пытается ввести, намного проще.

Почему

Избавляйте пользователей от тяжелой необходимости печатать строки целиком —делайте это за них. Вы сэкономите им бесчисленное количество секунд при работе и сбережете их запястья. Дополнительным преимуществом является предотвращение ошибок: чем длиннее и необычнее нужная строка, тем больше шансы, что пользователь сделает опечатку. Автоматическое завершение записей позволяет избежать этого.

Как

С каждым новым символом, который вводит пользователь, приложение незамедлительно формирует список возможных вариантов завершения уже введенной строки. Если пользователь вводит одно из ограниченного числа допустимых значений, то используйте этот набор допустимых значений. Если же возможных значений очень много, то для завершения ввода можно использовать одну из следующие заготовок:

* предыдущие записи, ранее введенные этим пользователем, сохраненные в истории или настройках;
* распространенные фразы, которые часто использовались в прошлом другими пользователями, поставляемые в качестве встроенного «словаря» приложения:
* объекты, подходящие по контексту, например корпоративный список контактов для внутренней электронной почты.

**Шаблон 74- Dropdown Chooser (Раскрывающийся селектор)**



**Рис. 3.** Панель выбора цвета в Word

Используйте концепцип меню, но в более широком смысле — создайте раскрывающуюся или всплывающую панель, содержащую

более сложный пользовательский интерфейс для выбора значения.

**Основы интерфейса и дизайна в редакторах**

Пользователи, создающие что-либо, работают нелинейно: они создают небольшой фрагмент, оценивают, как он выглядит или работает, меняют его, дополняют, что-то удаляют и делают шаг назад, чтобы бросить очередной оценивающий взгляд. Когда инструменты реагируют на действия пользователя и предназначены специально для выполнения определенной задачи, они «растворяются» в сознании пользователя, и он полностью уходит в творческий процесс.

Для того чтобы добиться состояния потока, пользователь должен уметь эффективно и искусно пользоваться инструментами. К счастью для пользователей, большинство редакторов, относящихся к одной идиоме, прошли долгий путь эволюции и стали очень похожими, поэтому навыки и привычки, приобретенные в одном редакторе, могут переноситься и в другие. Однако не нужно делать свой редактор идентичным какому-нибудь другому — в дизайне редакторов достаточно много пространства для инноваций. Тем не менее лучше не пытаться за один раз «изобретать слишком много велосипедов.

Есть еще вопрос эффективности. Ничто так не нарушает рабочий поток, как излишне длинные последовательности диалоговых окон, трудоемкие и изнурительные движения мышью или необходимость ждать, когда же программное обеспечение выдаст ответную информацию. Создавая редакторы, всегда стремитесь, чтобы реакция приложения была незамедлительной: обычные шаги должны выполняться минимальным количеством нажатий клавиш или движений мыши, а каждое действие должно давать немедленный результат.

Многие решения в дизайне взаимодействия, принимаемые вами при разработке любого компоновщика или редактора, относятся к редактированию в стиле WYSIWYG, прямой манипуляции, режимам или выделению. Правильное понимание этих концепций критически важно для того, чтобы обеспечить пользователю удобную и эффективную работу.

В большинстве редакторов наиболее важными являются четыре элемента:

возможность редактирования по принципу WYSIWYG,

возможность напрямую манипулировать интерфейсом и элементами данных,

разнообразие режимов работы;

широкий диапазон возможностей выделения текста или объектов.

**Редактирование WYSIWYG**

Сокращение WYSIWYG расшифровывается как «what you see is what you get» — что видишь, то и получаешь (хотя вы, наверное, это и так знали). В таких интерфейсах пользователю проще редактировать содержимое, так как во время разработки ему не приходится напрягаться, чтобы понять, как в конечном итоге будет выглядеть результат.

Редактирование WYSIWYG лучше всего подходит для изображений, векторной графики, видео и анимации, некоторых компоновщиков графического пользовательского интерфейса и «жестко» отформатированного текста — когда конечный продукт выглядит одинаково, независимо от контекста, в котором он просматривается.

Но подумайте об HTML. Когда вы создаете веб-страницу, у вас может не быть» полного контроля над готовым продуктом. В зависимости от того, как написан код, страница может управляться множеством факторов на стороне конечного пользователя: таблицы стилей, размер текста, размер окна, разнообразие браузеров и устройств отображения, параметры размещения элементов и даже настраиваемое содержимое. Что означает WYSIWYG, если каждый зритель видит уникальный конечный результат? Это открытый вопрос.

Помимо этого, если вы разрабатываете текстовый редактор, производящий отформатированный текст, то учтите, что практически на каждом компьютере под управлением Windows есть копия приложения Microsoft Word, поэтому в голове пользователя существуют вполне конкретные и очень сильные ожидания относительно того, как должен работать текстовый редактор. С другой стороны, практически у каждого пользователя есть электронная почта, и большинство редакторов электронной почты создают простой текст. Предполагалось изначально, так как электронная почта по своей сути — это средство передачи простого текста. Существует миллион веб-страниц, позволяющих редактировать сообщения на форумах или в блогах и в основном с использованием простого текста, хотя некоторые из них разрешают выполнять несложное форматирование при помощи тегов. В про­граммах мгновенного обмена сообщениями также применяется простой текст.

**Прямая манипуляция**

*Прямая манипуляция* означает, что интерфейс позволяет пользователю хватать, нажимать, перетаскивать, отпускать, менять форму, накладывать друг поверх друга, рисовать, открывать, закрывать объекты интерфейса или каким-либо другим способом оперировать ими. Ключевым моментом здесь является ощущение «материальности» — несмотря на то, что в действительности объекты являются виртуальными, а управление ими осуществляется при помощи мыши или других устройств. У пользователя создается иллюзия прямого физического контакта с объектами.

Помимо этого, важна точность — пользователю необходи­мо точное управление местоположением указателя мыши и перетаскиваемых объек­тов, как если бы он перемещал их с места на место руками в реальном пространстве. Прямая манипуляция должна следовать некоторым общепринятым правилам. Как уже упоминалось ранее, люди предполагают, что они могут переносить ка­кие-то общие навыки из одного приложения в другое: например, умение копи­ровать и вставлять, перетаскивать, изменять размер при помощи манипуляторов и выделять несколько объектов путем рисования вокруг них трансформируемой рамки выделения.

**Режимы**

*Режим* интерфейса — это состояние, в котором действия пользователя приводят к результатам, отличным от обычной реакции приложения на те же действия. На­пример, щелчок и перетаскивание указателя мыши в программе рисования обычно означает вычерчивание рамки выделения вокруг нескольких объектов. Однако в режиме рисования линии щелчок и перетаскивание означает создание линии, идущей из первой точки во вторую.

Есть мнение, что наличие режимов в интерфейсе — это неправильно, так как из-за них пользователи слишком часто попадают в неприятности. Но это не всегда так. Пользователи, часто работающие с графическими редакторами, привыкают к ним. В действительности трудно представить, как разработать сложный графиче­ский редактор, не добавляя в него режимы и имея в качестве рабочих инструмен­тов только мышь и клавиатуру — эти устройства ввода будут функционально пе­регружены, чтобы обеспечить пользователю возможности рисования.

Однако режимы могут оказаться плохим решением, если:

* пользователю неочевидно, что интерфейс находится в определенном режиме;
* режимы заставляют пользователя делать лишнюю работу, так как их неудобно включать и выключать.

Первую проблему можно с легкостью решить, реализовав подходящую реак­цию пользовательского интерфейса. Например, указатель мыши может всегда отображать текущий режим приложения. Внимание пользователя фокусируется там, где находится указатель, так что такую подсказку пропустить сложно. Если для включения и выключения режимов используются кнопки, то «включенное» состояние кнопки должно заметно отличаться от «выключенного», например, она должна выглядеть нажатой, а не выступающей над поверхностью интерфейса.

Два связанных с режимами шаблона пытаются решить вторую проблему: **One-off Mode** (Одноразовый режим) и **Spring-Loaded Mode** (Подпру­жиненный режим). Оба упрощают задачу выключения режима; второй шаблон также упрощает включение режима. Сравните эти шаблоны с ситуацией, когда пользователю приходится перемещать указатель мыши с палитры через весь эк­ран в самый угол интерфейса, находить крохотную кнопку на палитре, щелкать на ней, чтобы включить режим. Затем следует снова переводить указатель на холст, рисовать что-то, возвращаться к палитре, чтобы включить другой режим, опять переходить к холсту... и так до бесконечности. Слишком много движений мышью.

**Выделение**

На первый взгляд, кажется, что возможность выделения реализовать несложно. Уже давно существуют соглашения о различных типах выделения, например выделении текста, элементов списка или графических объектов; люди уже привыкли ли к ним. В следующей таблице перечислены наиболее распространенные типы условных соглашений для трех групп интерфейсов-идиом:

* текстовые редакторы, включая редакторы простого и форматированного текста.
* векторная графика, большинство программ для рисования и компоновщики графического интерфейса пользователя;
* списки, таблицы и деревья.

В следующей таблице объединены наиболее часто встречающиеся действия для выделе­ния объектов (обратите внимание, что не все они приводят к выделению объектов). Пользователи с большой вероятностью будут ожидать, что ваше приложение следует этим правилам.

**Таблица** Наиболее распространенные действия по выделению объектов

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Текстовые редакторы** | **Редакторы векторной графики** | **Списки, таблицы, деревья** |
| Одинарный щелчок | Переместить сюда указатель вставки текста (не приводит к выделению объектов) | Выделить только этот элемент | Выделить только этот эле­мент |
| Двойной щелчок | Выделить это слово | Открыть или отредактировать этот элемент (не приводит к выделению объектов) | Открыть или отредактировать этот элемент (не пр»«-(не приводит к выделению объектов) |
| Тройной щелчок | Выделить эту строку или выделить этот абзац | Не применимо  | Не применимо |
| Щелчок, перетаскивание, отпускание  | Начать выделение здесь, закончить его там; выделить весь текст между этими двумя точками | Выделить все внутри получившейся прямо­угольной рамки выде­ления | Перетащить этот элемент отсюда туда (не приводит к выделению объектов) |
| Щелчокс удержаниемклавиши Shift | Начать выделение в точке, где находится указатель вставки текста, закончить его в точке щелчка и выделить весь текст между этими двумя точками | Добавить этот элемент в набор выделенных элементов  | Добавить в набор выделенных элементов этот элемент, а также все промежуточ­ные элементы («выделение непрерывной области») |
| Щелчокс удержаниемклавиши Ctrl | В зависимости от приложения | В зависимости от приложения | Добавить этот элемент в на­бор выделенных элементов («дискретное выделение») |

**Создание и использование меню и командных клавиш**

При организации в программе интерфейса пользователя необходимо решить основные задачи:

* предоставить пользователю возможность удобного для него выбора
нужного действия;
* дать возможность получить быструю подсказку по выбранному интерфейсному элементу;
* использовать удобную пользователю форму ввода необходимых для
решения задачи данных;
* обеспечить проверку правильности введенных данных с выдачей
понятного пользователю диагностического сообщения по возможности
сразу после совершения им ошибки;
* использовать стандартные интерфейсные элементы для облегчения
приобретения пользователем навыков работы с программой.

Основным интерфейсным элементом, используемым для решения первой из сформулированных выше задач, является *меню.* В приложениях Windows главное меню программы представляет собой совокупность го­ризонтально расположенных *подменю,* каждое из которых при выборе раскры­вается в вертикальный список *команд меню* и подменю нижних уровней,

Для предоставления пользователю возможности более быстрого вы­бора нужной ему команды меню или подменю существуют следующие возможности:

* использование *клавиши быстрого выбора,* нажатие на которую после
активизации главного меню или подменю позволяет выбрать нужный
элемент; для этого один из символов строки с названием элемента меню
выделяется подчеркиванием;
* использование *командных клавиш (акселераторов)* для наиболее ча­сто используемых команд меню; нажатие на командную клавишу позволяет сразу (без активизации меню) выполнить нужное действие; информация о командной клавише выводится рядом с названием команды меню;
* использование графических кнопок *панели управления (панели инст­рументов),* дублирующих основные команды меню.

По желанию программиста командные клавиши могут использовать­ся в программе и самостоятельно, т. е. без установки связи с какой-либо командой меню.

 При использовании кнопок панели управления, чтобы не увеличивать количество обработчиков событий, необходимо установить связь между командой меню и дублирующей ее кнопкой.

Для команд меню и подменю при их активизации (подсветке) пользователем должна выводиться поясняющая их смысл информация (так называемая длинная подсказка). Обычно эта подсказка размещается в *окне состояния,* соответствующего окна программы.

 Чтобы предупредить пользователя о невозможности в данный момент выполнения того или иного действия, соответствующие команды меню и дублирующие их графические кнопки должны быть заблокированы и отображаться блеклым цветом.

Иногда команды меню используются для сообщения пользователю об установке того или иного режима работы программы. В этом случае слева от названия команды может отображаться (или не отображаться) символ отметки — галочка (V). Примером таких команд могут служить команды управления отображением панели управления и строки состо­яния подменю «Вид».

Хотя чаще всего структура меню определяется на этапе проектирова­ния программ, при необходимости меню может быть модифицировано в ходе выполнения программы путем добавления или удаления элементов. Возможна и полная замена меню (например, при переходе по желанию пользователя от русскоязычного интерфейса к англоязычному, и наобо­рот).

Программист может организовать модификацию и так называемого системного меню, размещающегося в левом углу строки заголовка окна и содержащего команды для изменения размера, перемещения и закры­тия окна.

Кроме главного меню приложения в программе могут использовать­ся *контекстные (всплывающие) меню* (вертикальные), связанные с ка­ким-либо визуальным объектом и появляющиеся после того, как пользо­ватель нажмет на правую кнопку мыши в области этого объекта. Контек­стное меню содержит команды, определяющие применимый к данному объекту программы набор действий.

**Создание и использование диалоговых панелей**

Для организации диалога с пользователем в приложении Windows необходимо создать особый вид окон —- *диалоговые панели.* От обычных окон диалоговые панели отличаются как своим внешним видом, так и назначением. Особенности внешнего вида диалоговых панелей:

* наличие специальной рамки, не позволяющей пользователю изменять размеры окна;
* отсутствие кнопок свертывания и развертывания окна в строке за­головка и соответствующих им команд в системном меню окна;
* наличие в строке заголовка кнопки получения контекстной под­сказки о назначении размещенных в рабочей области окна элементов
управления.

В рабочей области панели диалога размещаются *элементы управления* (controls) — окна стандартных или определенных разработчиком клас­сов, предназначенные для предоставления пользователю удобной фор­мы ввода тех или иных необходимых для решения задачи исходных дан­ных. Элементы управления относятся к так называемым *дочерним* (child) *окнам* и, в отличие от обычных окон, не имеют заголовка, не могут менять своего размера и перемещаться по рабочей области их *родитель­ского* (parent) *окна* (диалоговой панели).

При подготовке панели диалога необходимо предусмотреть следую­щие возможности для облегчения работы пользователя:

* возможность перемещения между элементами управления с помощью клавиши табуляции;
* возможность быстрой активизации нужного элемента с помощью
комбинации клавиш Alt+символ быстрого выбора;
* возможность подтверждения ввода данных и закрытия панели диалога с помощью клавиши Enter;
* возможность отказа от ввода данных и закрытия диалоговой панели
с помощью клавиши Esc.

При работе с панелями диалога в приложении необходимо решить основные задачи:

1. задать начальные значения (или значения по умолчанию) элементам управления, с помощью которых пользователь вводит (выбирает) исходные данные, и обеспечить установку фокуса ввода на элемент, с которого должен быть начат ввод;
2. использовать блокировку тех элементов управления, доступ к которым пользователя в данный момент невозможен (заблокированные элементы отображаются блеклым цветом);
3. предоставить пользователю возможность получения краткой (всплывающей) подсказки, раскрывающей смысл выбранного в данный момент элемента управления панели диалога (контекстной помощи);
4. после подтверждения пользователем ввода данных проверить их правильность;
5. если пользователь ошибся при вводе данных, то вывести соответствующее сообщение (или иным образом указать на ошибку) и установить фокус ввода на элемент управления, содержащий неверные данные;
6. если введенные пользователем данные не содержат ошибок, то закрыть панель диалога и организовать передачу данных в соответству­ющую часть программы для их последующей обработки.

Панель диалога может в программе выполняться *модально* или *немодально.* Модальные панели не позволяют пользователю продолжить рабо­ту с программой до их закрытия (с подтверждением ввода данных или с отказом от него). Иначе говоря, модальные диалоги применяются в том случае, когда пользователь должен ввести необходимые исходные дан­ные для выполнения запрошенной им функции.

Модальные диалоговые панели должны содержать командную кноп­ку с действием по умолчанию и командную кнопку для отмены ввода. Командная кнопка с действием по умолчанию считается нажатой при нажатии пользователем клавиши Enter (если фокус ввода не находится, на другой командной кнопке). Обычно на такой кнопке размещается надпись «Ok». Командная кнопка для отмены ввода считается нажатой, если пользователь нажимает клавишу Esc. Обычно на такой кнопке раз­мещают надпись «Отмена».

При закрытии модального диалога в программу возвращается результат, позволяющий определить, ввел пользователь данные или отказался от ввода.

Немодальные диалоговые панели не требуют своего закрытия для продолжения работы пользователя с программой. Эта форма панелей диалога применяется, как правило, для информирования пользователя о ходе выполнения длительного процесса (печати документа, копировании файлов с компакт-диска или дискеты на жесткий диск и т. п.). Немодальные диалоги содержат обычно командную кнопку для прерывания длитель­ного процесса и не содержат кнопок с действием по умолчанию и для отмены ввода.

При закрытии немодального диалога в программу не возвращается никакого результата.

Иногда панели диалога выступают в роли главных окон приложений (например, в стандартном приложении Windows «Калькулятор»), В этом случае панель диалога может иметь меню, панель управления и строку.

**Обработка ошибок пользователя при работе с панелями**

**диалога**

Любая профессионально написанная программа должна уметь обра­батывать неправильные по отношению к решаемой задаче данные, ко­торые ввел пользователь. Аварийное завершение работы программы с малопонятной диагностикой на английском языке или продолжение работы без всякой диагностики, но с выдачей в дальнейшем неверного результата одинаково недопустимо. При включении в приложение блока обработки ошибок следует руководствоваться следующими правилами:

 если ввод данных пользователем должен осуществляться в определенной последовательности, то использовать блокировку элементов уп­равления (например, командных кнопок) до ввода необходимых данных или организовать ввод данных с помощью мастера;

* если вводимые данные должны соответствовать определенному формату (вещественного числа, номера телефона, даты и времени), то блокировать попытку ввода пользователем недопустимых для данного фор­мата символов;
* ошибка при вводе данных должна быть обнаружена непосредственно после ее совершения пользователем;
* сообщение об ошибке должно отличаться от информационных сообщений по своему внешнему виду, быть максимально ясным и вместе с тем кратким;
* пользователь должен иметь возможность тут же исправить ошибку,
поэтому панель диалога не должна закрываться, а фокус ввода должен
быть установлен на элемент управления, в котором были введены не­
правильные данные.

Основными элементами управления, с помощью которых в панели диалога пользователь может ввести произвольные данные, являются тек­стовый редактор и таблица строк. Поэтому для проверки введенных дан­ных могут использоваться события, связанные с попыткой ввода с кла­виатуры очередного символа в этих элементах управления, потерей ими фокуса ввода (завершением ввода данных) и закрытием содержащей эти элементы панели диалога с подтверждением ввода данных (нажатием на кнопку «Ok» или подобную ей по назначению).

В сообщении о допущенной пользователем ошибке следует использо­вать стандартное окно.

**Организация просмотра документов**

Практически каждое приложение Windows должно уметь выполнять стандартный для операционной системы набор действий *с документами* (наборами данных, используемыми для решения конк­ретной задачи). Различают внутренний формат и внешнее представление документа. Под внутренним форматом документа понимают способ его представления при хранении во внешней памяти. Под внешним пред­ставлением документа понимают способ его отображения на устройстве вывода (экране или принтере).

В этом разделе будут рассмотрены способы организации работы с доку­ментами в приложениях Windows. Сначала рассмотрим вопросы органи­зации просмотра и редактирования документов.

Для просмотра документов могут использоваться как стандартные элементы управления (надписи, текстовые редакторы, расширенные списки, древовидные структуры и т.п.), так и средства более низкого уровня для непосредственного вывода графики и текста. Использование стандартных элементов управления было рассмотрено ранее. По­этому здесь мы остановимся только на возможностях управления вне­шним видом данных, отображаемых в стандартном элементе. Основное внимание будет уделено организации вывода текста и графики непосредственно в окне просмотра документа.

Важнейшим понятием при организации вывода текста и графики является понятие контекста устройства — структуры, содержащей информацию об основных характеристиках устройства вывода в целом (типе, мерах области вывода, режиме отображения и т.п.) и инструментах вывода, выбранных для выполнения очередной операции. К основным инструментам вывода относятся перо, кисть и шрифт.

Характеристики пера используются при рисовании контура геомет­рических фигур. Важнейшими характеристиками этого инструмента вывода являются цвет, толщина (в пикселях) и стиль вычерчиваемой пе­ром линии. Изменяя стиль пера, можно рисовать сплошные, пунктир­ные, невидимые и другие линии.

Основной характеристикой кисти является цвет, который использует­ся для заполнения внутренней области замкнутых геометрических фигур.

Шрифт используется при выводе текста, а его важнейшими характеристиками являются цвет контура выводимых символов, размер шрифта (в пунктах), название гарнитуры шрифта и стиль начертания символов (прямой, курсивный, полужирный и т.п.).

Вывод текста и графики возможен в двух режимах фона — прозра­чном и непрозрачном. При выводе в прозрачном режиме фона (TRANSPARENT) символы текста или линии геометрических фигур ри­суются непосредственно на «поверхности» области вывода. При выводе в непрозрачном режиме (OPAQUE) используется специально выбранный цвет фона.

По умолчанию в контексте устройства предполагаются выбранными перо черного цвета, рисующее сплошную линию толщиной в один пик­сель, кисть белого цвета, белый цвет фона и непрозрачный режим фона. Характеристики шрифта по умолчанию зависят от типа устройства вы­вода и используемой системы программирования.

Характеристика режима отображения, установленная для того или иного контекста устройства, влияет прежде всего на масштаб выводи­мого текста или изображения. По умолчанию используется режим ото­бражения ММ\_ТЕХТ, при котором один пиксель при выводе в про­грамме равен одному пикселю на устройстве вывода. Этот режим не­удобен при выводе с помощью одного фрагмента программы и в окно на экране, и на принтер, так как масштаб выводимых текста и фигур на разных устройствах будет различным. Кроме того, при выводе гео­метрических фигур возможны искажения их формы из-за разницы в вертикальном и горизонтальном разрешении устройства вывода. Для установки одинакового масштаба вывода целесообразно использовать режим отображения MM\_ISOTROPIC. При рисовании в этом режиме отображения также необходимо задать правило преобразования логи­ческих координат, используемых в программе, в физические коорди­наты устройства вывода.

Для редактирования документа пользователю могут быть также пре­доставлены элементы управления, рассмотренные ранее. Они могут размещаться на специальных панелях диалога или в панели уп­равления.

При организации просмотра документа помимо собственно отобра­жения текста и графики необходимо решить задачу «прокрутки» изобра­жения, не полностью помещающегося в окне просмотра. Хотя обычно эта задача решается достаточно просто с помощью стандартных средств системы программирования, в ней тоже есть более сложные элементы. Первым таким элементом является правильное задание размеров обла­сти отображения документа для автоматического появления и скрытия линеек «прокрутки», что позволяет пользователю наглядно представлять себе, просматривает он весь документ или его часть. Также необходимо обеспечить пользователю возможность управления линейками «прокрут­ки» с помощью не только мыши, но и клавиатуры, что не поддержива­ется стандартными средствами.

Еще одной задачей, тесно связанной с организацией просмотра до­кумента, является использование дополнительных возможностей «всплы­вающих» подсказок, которые могут быть связаны с элементами управле­ния, размещенными в области отображения документа, а также с лю­быми ее частями.