



Глава 7

«Составные части программного интерфейса. Окна»



Тема 22. Окна

1. Недолгая история окон на экране.
2. Типы окон.
3. Строки заголовка окна.
4. Панели инструментов.
5. Полосы прокрутки и их альтернатива.
6. Вкладки.

Окна

(Типы окон)

Поскольку разработка интерфейса заключается в основном в том, чтобы правильно помещать правильные элементы управления в правильные окна или экраны, окна требуют не меньше заботы, чем элементы управления.

Типы окон

Современная наука знает несколько типов окон, а именно:

- ☐ главные окна программы
- ☐ окна документа
- ☐ режимные диалоговые окна
- ☐ безрежимные диалоговые окна
- ☐ палитры
- ☐ окна браузера (поскольку используемая в интернете технология существенно отличается от технологии ПО, этот тип окон стоит несколько особняком).

При этом доля отдельных типов в общем пироге со временем изменяется:

- ☐ окна документов отмирают, заменяясь окнами программ,
- ☐ режимные диалоговые окна сменяются безрежимными,
- ☐ а безрежимные, в свою очередь, палитрами.

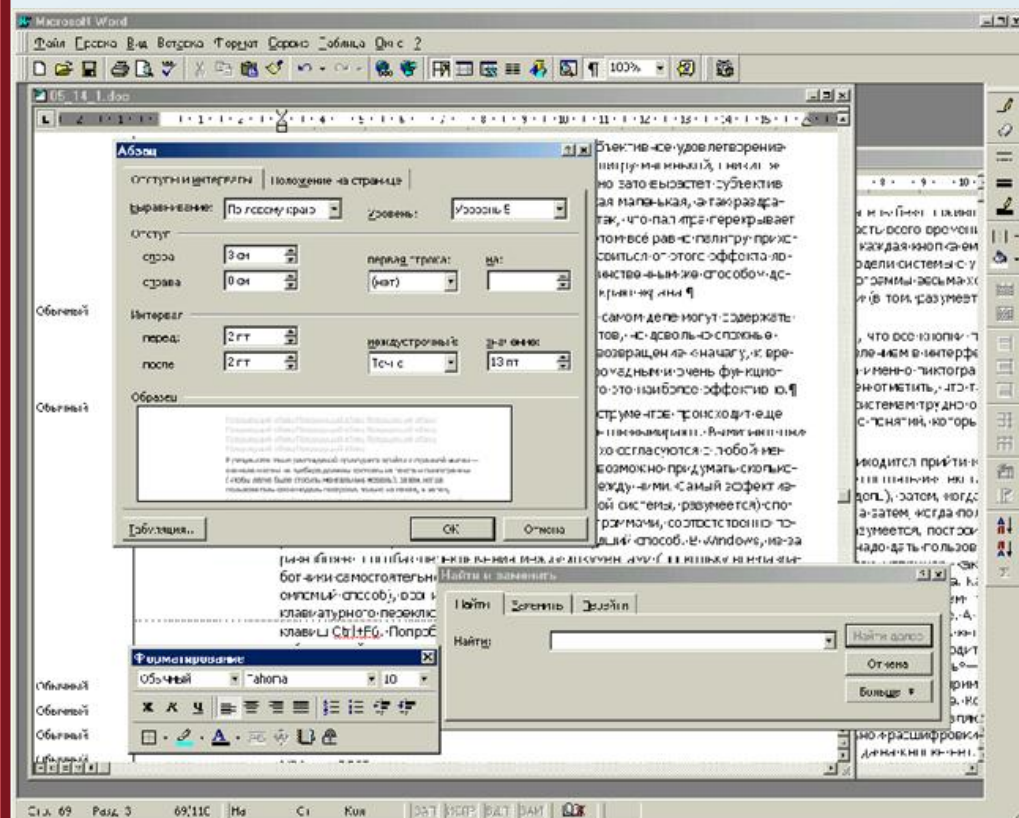
Интересно, что идея палитр тоже клонится к закату (палитры сменяются панелями инструментов), так что *в будущем, скорее всего, в ПО останутся только окна программ, панели инструментов и безрежимные диалоговые окна (которые разработчики поленятся переделывать).*

Окна

Недолгая история окон на экране

Сейчас многим в это трудно поверить, но сравнительно недавно никаких окон не было, даже диалоговых окон, которые уже стали восприниматься как данность. Вместо них какая-то часть экрана выделялась под меню, которое в те времена было функционально более богатым, чем меню теперешнее (так, нормой были поля ввода в меню).

Типы окон на примере MS Word 97.



Самое большое окно есть окно программы.

Внутри него два окна документов, в более свежих версиях Word их уже нет.

Слева сверху располагается режимное диалоговое окно (Абзац), под ним справа – безрежимное (Найти и заменить).

Слева внизу располагается палитра. Сверху и справа две панели инструментов (бывшие палитры).

Окна

Недолгая история окон на экране

Потом, с появлением графического режима, стало возможным реализовать в интерфейсе метафору рабочего стола (если бумажные документы могут лежать на столе друг на друге, то почему этого не могут делать электронные документы?). Появились окна программ, окна документов и диалоговые окна, первоначально сплошь режимные.

Понятие «режимное диалоговое окно» кажется довольно загадочным (еще более загадочным кажется его англоязычный вариант «модальное диалоговое окно»). На самом деле всё просто. Если открывшееся окно блокирует доступ к остальной части системы, происходит, фактически, запуск нового *режима работы* (поскольку функциональность отдельного диалогового окна никогда не совпадает с функциональностью системы в целом). После того, как окно закрыто, происходит возвращение предыдущего (основного) режима. В этом и есть всё значение термина «режимный».

Окна

Недолгая история окон на экране

Прошло несколько лет, и наличие режима в диалоговых окнах стало немодным.

- ❑ Во-первых, всех раздражает, что, вызвав диалоговое окно и обнаружив, что вызвано оно преждевременно, приходится закрывать окно и открывать его в следующий раз заново.
- ❑ Во-вторых, что важнее, в системах, ориентированных на документы, режим сбивает внимание пользователя и вообще лишает его ощущения управляемости (в отличии систем, ориентированных на формы ввода, в которых режим работает лучше, чем его отсутствие).
- ❑ В-третьих, сама по себе идея сближения интерфейса с реальным миром (в частности, метафора рабочего стола) протестовала против идеи режимов в любом их проявлении, поскольку в реальном мире вообще не бывает режимов, аналогичным интерфейсным. А поскольку «дизайн пользователей» был ориентирован на функционирование в реальном мире, **решили не переделывать пользователей, а переделать интерфейс.**




Окна

(Безрежимные диалоговые окна)

Недолгая история окон на экране


Избегайте режимов работы



Так появились **безрежимные диалоговые окна**, т.е. окна, которые можно было неограниченное время держать на экране, переключаясь по мере надобности между ними и собственно документом.

К сожалению, и здесь не без проблем. Дело в том, что **такие диалоговые окна нельзя делать тонущими, т.е. позволять пользователю перекрывать их окнами документа или программы.**


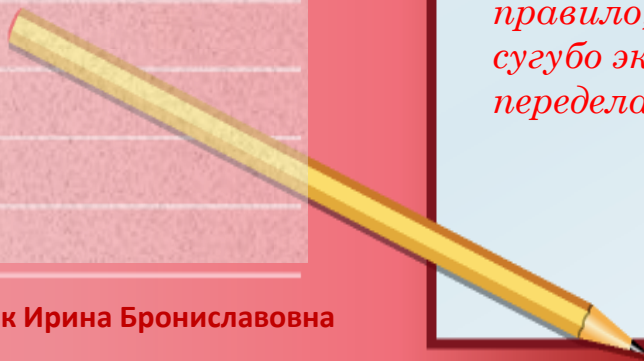
Причина проста – пользователи забывают, что они когда-то открывали соответствующее окно и пытаются открыть его заново. Зачем, спрашивается, такие окна? Поэтому решили сделать такие окна плавающими, т.е. перекрываемые только другими плавающими окнами этой же программы или другими программами.





Окна (Палитры)


Недолгая история окон на экране



Некоторые диалоговые окна невозможно сделать безрежимными: например, что делать с сообщениями об ошибках? Но, в целом, с переводом окна в безрежимное состояние нет особой проблемы.

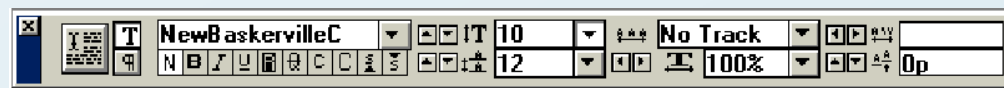
Но и тут обнаружилась проблема. Дело в том, что просто диалоговое окно, даже будучи безрежимным, малополезно, поскольку перекрывает слишком много важного и нужного. Решение этой проблемы было эволюционным, и поэтому относительно простым – были придуманы **палитры**, т.е. *окна, из которых выжали всё пустое место*. Сразу оказалось, что палитры, помимо малых размеров, имеют одно большое достоинство: *пользователи очень любят их расставлять на экране индивидуальным порядком*. Пользы это особой не приносит, зато существенно повышает субъективное ощущение контроля над системой.

К сожалению, *визуальный дизайн палитр, как правило, довольно сложен и длителен, так что сугубо экономические причины мешают переделать в палитры все диалоговые окна*.



Окна

Недолгая история окон на экране



Пример палитры из программы
Adobe PageMaker

Как легко догадаться, проблема была найдена и в палитрах.


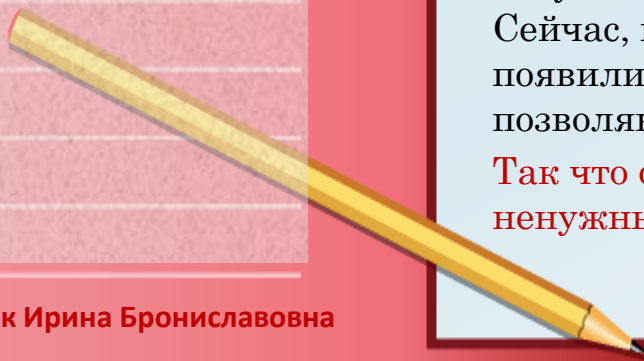
Существует неформальный, но на удивление верный закон, гласящий, что *субъективная важность информации, перекрываемой диалоговым окном (палитрой в частности), не зависит ни от размеров, ни от положения окна, а зависит только от периметра.*

В результате постоянно оказывается, что пользователи, стараясь открыть нужную информацию, перекладывают окна с места на место, что снижает производительность (несущественно) и субъективное удовлетворение (существенно). При этом если сделать палитру маленькой, снизится вероятность её вынужденного перетаскивания, но зато вырастет субъективное неудовольствие от её перетаскивания (*«такая маленькая, а так раздражает»*). Более того. Гораздо чаще оказывается так, что палитра перекрывает не всю нужную информацию, но её часть; при этом всё равно палитру приходится перемещать. Единственным способом избавиться от этого эффекта является *уменьшение периметра палитры, а добиться этого можно, только прикрепив палитры к краю экрана.*



Окна (Панели инструментов)

Недолгая история окон на экране



Так родились *панели инструментов*, которые на самом деле *могут содержать (и содержат)* не только пиктограммы инструментов, но довольно сложные элементы управления.

Параллельно с рождением сложных панелей окна документов вымирают по двум простым причинам.

- ☐ Во-первых, они плохо согласуются с ментальной моделью большинства пользователей.
- ☐ Во-вторых, невозможно придумать сколько-нибудь эффективного способа переключаться между ними.

Для того чтобы запустить две одинаковые программы, каждая с одним документом внутри, не хватало ресурсов компьютера, вот и приходилось запускать одну программу с двумя документами. Сейчас, напротив, памяти достаточно, к тому же появились технологии программирования, позволяющие ни о чем таком даже не думать.

Так что окна документов постепенно становятся ненужными.

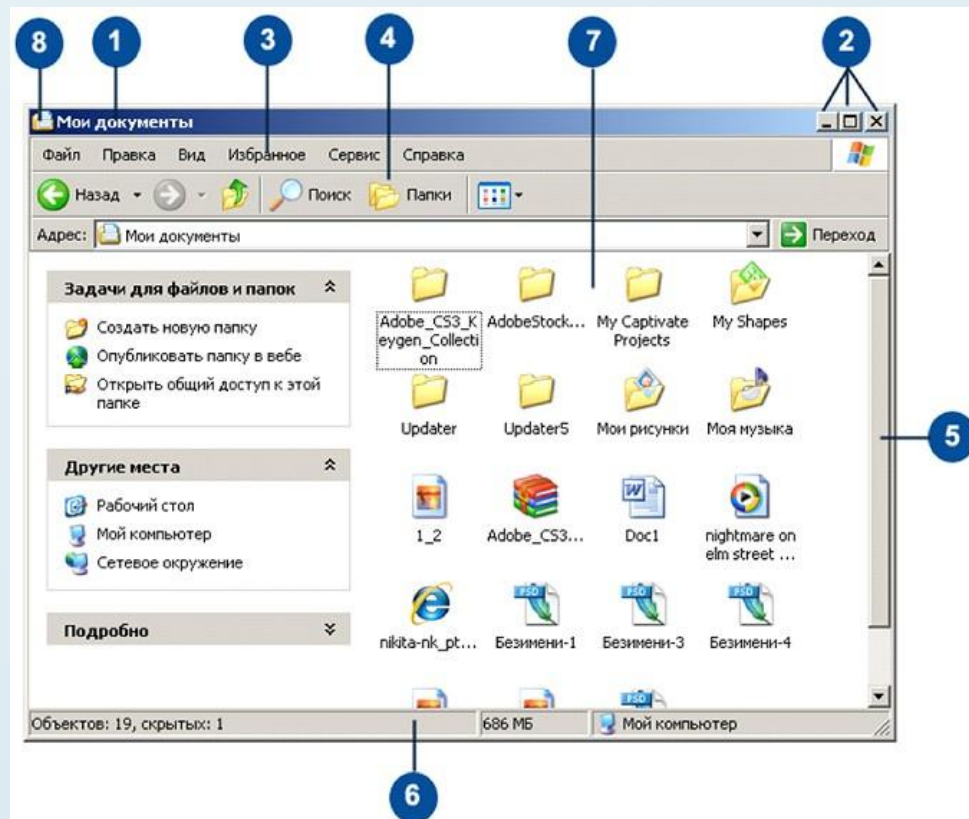


Окна

Элементы окна

Окна, помимо областей с элементами управления, имеют некоторые общие элементы, главными из которых являются:

- ☐ строки заголовка окна,
- ☐ строки статуса,
- ☐ панели инструментов
- ☐ полосы прокрутки.



- 1 - Заголовок окна;
- 2 - Кнопки управления окном;
- 3 - Строка меню;
- 4 - Панели инструментов;
- 5 - Полосы прокрутки;
- 6 - Строка состояния;
- 7 - Рабочая область;
- 8 - Системное меню;

Окна

Строка заголовка окна

У каждого окна есть строка заголовка. Поэтому пользователи строкой заголовка интересуются весьма мало, и обращают внимание на строку заголовка, только обучаясь пользоваться компьютером или в ситуациях, когда они совсем ничего не понимают в системе. Из этого, однако, не следует, что строкой состояния можно пренебрегать. Точнее, самой *строкой как раз пренебречь можно, но её содержимым – нельзя.*

Текст и, в меньшей степени, пиктограмма заголовка играют важную роль в ПО (они заведуют переключением задач) и очень важную в интернете (заведуют навигацией). Пользователь сохраняет способность опознать программу по её пиктограмме и обрывку текста, но теряет возможность опознавать документы. Тем не менее, *сокращать название программы нужно безусловно.*

Окна

Строка заголовка окна

Нажатие на пиктограмму в строке заголовка вызывает раскрывающееся меню, являющееся замечательным местом для вызова функций, которые нужны только наиболее опытной аудитории

Иная ситуация в интернете. Поскольку пиктограмма в строке заголовка приходит от браузера, нет особой возможности оптимизировать переключение задач. С другой стороны, качество этого заголовка оказывает существенное влияние на навигацию, поскольку при показе результатов поиска в поисковых системах заголовком элемента становится содержимое тега Title. Каковое содержимое и попадает в обычном режиме наверх экрана. При *этом в интернете нет проблемы с текстом заголовка – что хотим, то и пишем (стараясь не обращать внимания на то, что к этому прибавится название браузера)*.

Правило релевантности действует и здесь – в начале строки должна быть более релевантная информация, нежели в её конце. Поскольку связки «программа-документ» в интернете нет, эффективнее всего показывать адрес текущей страницы в навигационной системе сайта (если сайт иерархический). В данном случае релевантность требует, чтобы *сначала шло название текущего документа, затем раздела, в котором он находится, затем раздела более высокого уровня и так далее. Не надо также забывать, что размер строки ограничен, так что более 70-80 символов в ней быть не может.*

Также важно понимать, что тот факт, что пользователи редко читают заголовки окна, вовсе не означает, что заголовки пользователям не нужны. Напротив, хороший заголовок может здорово облегчить понимание работы диалога. Поэтому *наличие на экране заметного и адекватного заголовка окна часто оказывается очень полезным*. Жалко только, что в обычном Windows-интерфейсе места под него нет.

Окна

Панели инструментов

*Панель инструментов
нежелательно делать
единственным способом
вызова функции*

Все панели имеют следующие достоинства:

- ☐ они позволяют пользователям быстро вызывать нужные функции мышью,
- ☐ они позволяют пользователям меньше задействовать память,
- ☐ они повышают визуальное богатство интерфейса,
- ☐ они ускоряют обучение работе с системой (по сравнению с раскрывающимся меню) благодаря своей большей наглядности.

Зато они имеют и недостаток: занимают много места на экране, так что поместить в них всё, что хочется, невозможно.

Решить эту проблему можно двояко.

Во-первых, *можно (и нужно) помещать в панель только наиболее часто используемые команды* (поддерживая это решение возможностью индивидуальной настройки панели пользователем).

Во-вторых, *панель можно сделать зависимой от контекста действий пользователя.*

Оба способа не противоречат друг другу, так что использовать стоит оба.

Окна

Текст на кнопках

Самыми частыми элементами управления, размещаемыми на панелях инструментов, являются командные кнопки, при этом их использование отличается от обычного. Дело в том, что места настолько не хватает, что очень хочется заменить текст кнопок пиктограммами. Но это не так просто.

Случай 1. *Опытный пользователь, уже знающий, где на панели находится нужная кнопка, знающий её значение, при этом выбор действия уже произведён при помощи сложившейся ментальной модели.* В такой ситуации слова пользователю уже не важны, важно отличие нужной ему кнопки от остальных. Т.е. такому пользователю даже уже все равно, что на пиктограмме изображено, лишь бы она выглядела максимально контрастно (чтобы ускорить её поиск).

Случай 2. *Опытный пользователь, обладающий сложившейся ментальной моделью, но не знающий, где конкретно расположена нужная ему кнопка и как она выглядит.* Выбор действия уже произведен, осталось только найти нужную кнопку. При этом пиктограмма оказывается ненужной, так как в качестве матрицы пользователь использовать её не может (поскольку не знает, как она выглядит). Более того, поскольку пользователь ищет слово из содержимого своей кратковременной памяти, каждая пиктограмма будет его без пользы отвлекать, при этом пользователь будет тратить время на расшифровку смысла всех попадающихся ему на пути пиктограмм.

Случай 3. *Неопытный пользователь без сложившейся ментальной модели.* Такой пользователь большую часть всего времени тратит на поиск нужной ему кнопки, а также, поскольку каждая кнопка ему внове, на постоянное улучшение своей ментальной модели системы с учетом своих новых открытий. В таких случаях пиктограммы лучше текста, но не заменяют его, так как помогают *быстрее* понять действие кнопки (в том, разумеется, случае, когда пиктограмма адекватна смыслу действия).

Окна

Таким образом, *эффективнее всего* (учитывая все аргументы за и против) *делать кнопки на панелях инструментов диалектически: самые главные кнопки нужно делать парой «пиктограмма плюс текст», а остальные в зависимости от их направленности – функции для опытных пользователей пиктограммами, а для неопытных текстом.*

В результате таких рассуждений приходится прийти к странной мысли – *сначала кнопки на панели инструментов должны состоять из текста и пиктограммы* (чтобы легко было строить ментальную модель), *затем*, когда пользователь свою модель построил, *только из текста*, а *затем*, когда пользователь окончательно обучился пользоваться системой, *только из пиктограммы.*

Разумеется, построить такую систему невозможно, так что приходится определяться. Поскольку в двух случаях из трех текст оказывается нужен (тем более что начинающие и средне продвинутые пользователи составляют большинство), удалять его из панели оказывается неправомерным.

Здесь действует ещё один закон. Поскольку кнопка с пиктограммой и текстом всегда больше кнопки с текстом или пиктограммой просто, она оказывается более эффективной в отношении скорости, поскольку в неё легче попасть мышью.

Окна

Полосы прокрутки и их альтернатива

Пользователям не нравятся горизонтальные полосы прокрутки

Когда графических интерфейсов еще не было, пользователи перемещались по документу с помощью клавиатуры. С тех далёких времен на клавиатуре остались клавиши Home и End, равно как Page Up и Page Down. В целом, пользователи были удовлетворены своей судьбой. Затем появились графические интерфейсы. Первым делом были придуманы полосы прокрутки. К сожалению, оказалось, что они работают не слишком хорошо.

Проблема полос прокрутки заключается в следующем: для маленьких документов они не очень нужны, поскольку пользователям, держащим руки на клавиатуре, гораздо легче переместиться к нужному фрагменту с помощью клавиш со стрелками. Напротив, в больших документах малое перемещение ползунка приводит к существенному сдвигу области просмотра, так что после перемещения нужно еще и подправляться либо клавиатурой, либо стрелками на полосе прокрутки.

Далее была придумана «дополнительная стоимость» полосок – размер ползунка был сделан пропорциональным отношению видимой части документа ко всему его объёму. Это очень хорошая и полезная функция, поскольку она позволяет использовать полосы прокрутки не только как элемент управления, но и как индикатор размера документа, благодаря чему степень бесполезности полосок значительно снижается. Помимо этого было придумано довольно много других дополнительных стоимостей, так, например, на полоске прокрутки можно отображать границы разделов документа.

Окна

Полосы прокрутки и их альтернатива

Полосы прокрутки без индикации размера документа практически бесполезны

С появлением мышей с колёсиками, полосы прокрутки смело можно делать тоньше

Тем не менее, всё это так и не сделало полосы прокрутки здорово лучше: как и раньше, полосы не столько помогают перемещаться по документу, сколько показывают то, что не весь документ помещается на экране. Решение этой проблемы пришло с несколько непривычной стороны, во всяком случае, графический пользовательский интерфейс не пригодился – *была придумана мышь с колёсиком прокрутки*. Решение это чуть ли не идеальное, поскольку не требует от пользователя переносить внимание с документа на элемент управления. Конечно, для перемещения по большим документам колесо не слишком эффективно (палец устаёт), но малые и средние перемещения получаются замечательно, тем более что процент больших документов невелик.

Таким образом, полосы прокрутки стали ещё более бесполезны, поэтому *относиться к ним надо не как к данности, но как к еще одному элементу управления, который можно использовать, а можно и не использовать*. При этом есть как аргументы в пользу использования, так и существенный аргумент против него – *полоски прокрутки занимают много места на экране*. Ладно еще, когда на экране одна полоска, а что делать, если их три или более?

Окна

Полосы прокрутки и их альтернатива

К сожалению, вовсе не использовать полосы прокрутки в ПО затруднительно, MS Windows User Experience прямо заставляет разработчика ими пользоваться. В интернете ситуация иная – никто никого не заставляет. Осталось разобраться, как же сделать пролистывание документа идеальным.

Если всё-таки приходится оставлять полосы прокрутки, крайне желательно добиться нескольких свойств полосок:

- ❑ *Размер ползунка должен показывать общий объем пролистываемого документа.*
- ❑ *Стрелки на полосах должны быть спаренными, т.е. обе стрелки должны находиться рядом, а не на разных сторонах полоски. Это один из случаев, когда логичность интерфейса вступает в противоречие с эффективностью. Если при перелистывании была допущена ошибка, спаренные кнопки позволяют минимизировать перемещение курсора к стрелке, ведущей в обратную сторону.*
- ❑ *Если невозможно сделать динамическое изменение области просмотра при пролистывании, необходимо показывать текущее местоположение пользователя во всплывающей подсказке.*
- ❑ *Необходимо обеспечить обработку погрешности перемещения курсора.* Когда пользователь курсором перемещает ползунок, а смотрит в это время на документ, курсор может сойти с полосы. До определённого момента (смещение на 30-70 пикселей) система должна такое смещение игнорировать.

Окна

Альтернативные элементы управления

*Производительность
пользователей порой
можно повысить вдвое,
просто изменив
расположение элементов
управления, не меняя сами
эти элементы.*

В качестве альтернативных элементов управления используются **кнопки со стрелками**, т.е. фактически полосы прокрутки, из которых вырезано самое главное.

Это не очень хороший элемент, потому что он совершенно линейен: когда пользователь нажимает на кнопку со стрелкой, документ листается с фиксированной скоростью, изменить которую пользователь не в силах. Это приводит либо к медленному пролистыванию, либо к низкой точности.

Гораздо эффективнее **малюсенький джойстик**, часто встречающийся в ноутбуках. Сущность этого элемента проста: на экране располагается объект, нажатие на который меняет курсор на изображение направленных в разные стороны стрелок. Если пользователь перемещает курсор с нажатой кнопкой мыши в сторону, документ в эту же сторону и прокручивается, причем скорость прокрутки пропорциональна расстоянию, на которое перемещен курсор. Важно только не забывать его блокировать, когда пролистывать нечего. Такой элемент управления в настоящее время реализован в MS Windows и доступен по нажатию средней кнопки мыши. Структура окна Структура и само устройство окна или экрана является, пожалуй, самым существенным фактором, влияющим на качество интерфейса в этом окне.

Окна

Большинство руководств по проектированию интерфейсов, перечисляя требования к структуре окна, ограничиваются замечанием, что *терминационные кнопки (т.е. командные кнопки, управляющие окном, например Ок или Закрыть) должны быть либо снизу окна, либо в правой его части.*

Это хорошо, но мало. На самом деле всё сложнее.

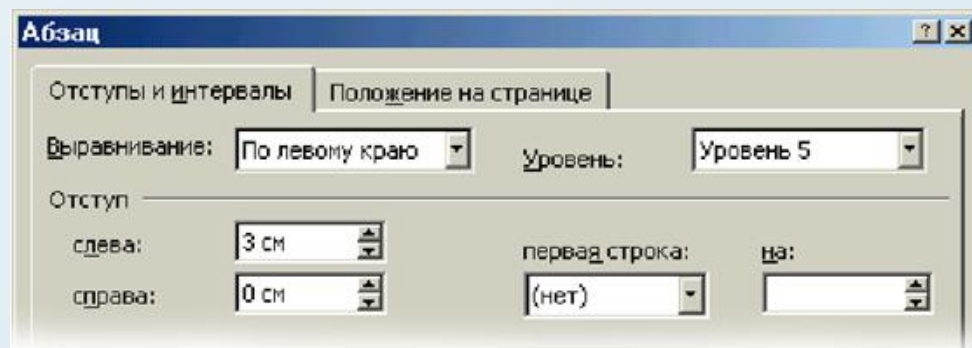
- ❑ Во-первых, *окно должно хорошо сканироваться взглядом*, т.е. его основные части должны быть сразу видны и заметны. Как правило, в окнах с малым количеством элементов управления проблем со сканированием не возникает.
- ❑ Во-вторых, *окно должно читаться, как текст*. При прочих равных, окно, все элементы управления которого можно без труда связно прочесть, будет лучше запомнено и быстрее обработано мозгом (поскольку не придется преобразовывать грамматику окна в грамматику языка).
- ❑ В-третьих, *оно должно удовлетворять закону «релевантное – вперед»*. Чаще всего используемые элементы должны быть расположены в левой верхней части экрана, реже используемые – в правой нижней части. Обратите внимание, что окно сканируется взглядом точно так же, как происходит процесс чтения – сначала взгляд переходит в левый верхний угол, затем перемещается вправо, затем переходит на следующую «строку» и т.д. Поэтому, например, вертикальный элемент управления, разрывающий эти воображаемые строки на части, будет всегда замедлять сканирование окна (и вызывать недовольство у пользователей).

Окна

В интерфейсе всегда должно быть реализовано правило: *сначала выбор параметров, а затем действие.*

Окно должно читаться, как текст

Пример читаемого окна



Читается он следующим образом: «Текст выравнивается по левому краю, уровень пятый, отступ слева 3 см, справа 0 см, первая строка нет, на 5 и так далее».

На этом примере прекрасно видны все неопределенности в окне: например, не говоря уже о том, что непонятно, чего именно пятый уровень, видно, что подписи к «первая строка» и к «на», расположенные сверху, разрывают единый по смыслу элемент управления на два разных. При этом один элемент управления должен однозначно преобразовываться в единый фрагмент предложения, а единая группа элементов – в целое предложение.

Окна

Увеличение площади

Площадь экрана ограничена, напротив, количество элементов управления, которых может понадобиться уместить в едином функциональном блоке (т.е. окне), не ограничено ничем.

В этом случае приходится использовать **вкладки**. Чтобы правильно их использовать, нужно соблюдать определенные требования.

Помимо *умещения максимального количества элементов управления в диалоговом окне*, вкладки могут выполнять еще одну роль, а именно *скрывать от неопытных пользователей не очень нужную им функциональность*. Проблема заключается в том, что когда нужно просто уместить в окно побольше элементов, вкладки скрывают от пользователей функциональность, возможно, что и нужную.

Нежелательно размещать на закрытых вкладках элементы, которые пользователям обязательно понадобятся, даже если эти элементы и не нужны постоянно (в этом случае правило про релевантность должно отступать). Разумеется, это не касается опытных пользователей.

В Интернете и в остальных операционных системах, которым Microsoft была не указ, *кнопки, увеличивающие размер окна и открывающие продвинутые элементы управления, сохранились в полном объеме*. Учитывая тот факт, что никаких пользовательских проблем с ними не замечено, можно смело рекомендовать их и для использования в Windows, тем более что они позволяют добиться определенного брендинга.

Окна

Число вкладок

Самым эффективным решением является комбинация второго и третьего способов: основные экраны реализуются в форме вкладок, а дополнительные вызываются через раскрывающийся список. Это позволяет обеспечить максимальное количество наглядности и скорости работы.

Теоретически *число вкладок может быть сколь угодно большим*. На практике оно ограничивается двумя факторами:

- ❑ во-первых, объемом кратковременной памяти,
- ❑ во-вторых, размером области, в которые ярлыки вкладок могут помещаться.

Дело в том, что если ширина всех ярлыков будет больше ширины окна, придется либо делать несколько строк ярлыков, либо скрывать часть из них, пока пользователь не нажмет специальную кнопку. И то и другое плохо.

1. Несколько строк ярлыков плохо по двум причинам.

- ❑ Во-первых, из-за большого количества мелких деталей (границ ярлыков), вся конструкция довольно медленно сканируется, т.е. трудно найти нужную вкладку.
- ❑ Во-вторых, при последовательном обращении к нескольким вкладкам из разных рядов эти ряды меняются местами, т.е. каждый раз нужно заново искать нужную вкладку.

И то и другое крайне негативно сказывается на субъективном удовлетворении и скорости работы.

2. Скрывать часть ярлыков тоже нехорошо.

3. Можно просто убрать вкладки, заменив их раскрывающимся списком.

Этот способ тоже не слишком хорош, поскольку не слишком визуален и к тому же сравнительно медлителен.

Окна

Объем содержимого

*Не старайтесь
рассортировать
элементы так, чтобы в
каждой вкладке их был
одинаковое количество*

Фактически, каждая вкладка представляет собой отдельное диалоговое окно внутри другого диалогового окна. Поэтому странной выглядят попытки (встречающиеся огорчительно часто) рассортировать элементы управления так, чтобы во всех вкладках их было поровну. Делать это ни в коем случае нельзя. Один экран должен содержать только те элементы, которые в нем нужны и которые пользователь может в этом экране ожидать.

В диалоговом окне с вкладками терминационные кнопки обязательно должны располагаться вне области вкладок.

Окна

Перемещение в пределах окна

Любая форма ввода, которой часто пользуются, обязана корректно работать с Tab, при этом желательно, чтобы она работала и с горячими клавишами

Помимо навигации между экранами, существует еще и навигация внутри отдельных экранов.

Пользователям необходимо дать возможность максимально быстро переходить к необходимым элементам управления.

Для этого у них есть два способа – мышь и клавиатура.

1. С мышью все более-менее понятно: закон оптимизация диалогового окна, уменьшающая дистанцию перемещения курсора, всегда приводит к росту (хотя и небольшому) производительности пользователей.

2. С клавиатурой сложнее. Пользователь может перемещаться между элементами управления двумя разными способами: клавишей Tab и горячими клавишами. Перемещаться клавишей Tab медленно, но зато для этого не нужно обращаться к памяти или высматривать клавиатурную комбинацию для нужного элемента. Напротив, горячие клавиши позволяют быстрее перемещаться вглубь экрана, но требуют запоминания клавиш.

Таким образом, пользователи, которые часто вводят данные в какой-либо экран, стараются использовать клавишу Tab и только изредка пользуются горячими клавишами.

Окна

Последовательные окна

Особым вариантом окон являются **действия, выполняющиеся в последовательности сменяющих друг друга окон** (мастера). Чтобы осознать правила, применимые к ним, полезно определить причины, вызвавшие появление таких окон.

- ❑ Во-первых, существуют действия, для которых либо естественна, либо желательна жесткая последовательность. Эффективнее разбить действие на несколько разных экранов.
- ❑ Во-вторых, существуют действия, которые всегда будут вызывать проблемы у пользователей, либо потому, что эти действия сложны, либо потому, что нужны они редко (так что пользователям нет резона учиться). При этом единое окно для выполнения действия также оказывается неэффективным, поскольку объем справочной информации, которую в него нужно вместить, слишком невелик. В таких случаях разделение действия на последовательность экранов позволяет снизить насыщенность отдельных экранов и тем самым освободить место для справочной информации.

Окна

Переход между экранами

Пользователи должны получить возможность переходить не только на следующее окно в последовательности, но и на предыдущие окна. Менее очевидным является другое требование к мастерам: **переход должен быть максимально легким.**

Задача раскладывается на две составляющие:

- ❑ во-первых, нужно реализовать *возможность свободного перемещения по последовательности*. Если экранов немного (3-5), то вполне можно ограничиться стандартными кнопками Назад и Далее. Если же экранов много, переход по этим кнопкам будет, как минимум, медленным. В таких случаях *разумно дополнять кнопки раскрывающимся списком* (при этом, возможно, исключая из него ещё не пройденные экраны), либо, если есть место, *снабжать мастера списком всех экранов (отмечая текущий и пройденные экраны)*.
- ❑ Вторая составляющая – *четкость перехода*. Для пользователей мастер, т.е. последовательность экранов, кажется единым экраном, содержимое которого меняется. Эту иллюзию нужно поддерживать, поскольку она позволяет не сбивать контекст действий пользователя и поддерживать внимание на «сюжетно-важной» области экрана. Для этого *размер и расположение окна мастера, а также расположение и облик всех повторяющихся элементов (таких как терминационные кнопки) нужно выдерживать неизменными на протяжении всей последовательности*.

Окна

Контекст

В отличие от единого окна, в котором выполняется действие, в мастерах **необходимо поддерживать контекст действий пользователя.**

Поскольку ранее сделанная работа скрыта, пользователи могут потерять контекст, что может замедлить действие (контекст придется восстанавливать). Степень потери контекста зависит от количества экранов, времени, которое пользователи проводят за отдельными экранами и от времени реакции системы. И если *количество экранов в мастере редко превышает шести* (а это небольшое число), то время, проведенное на пройденных экранах и, особенно, реакция системы (особенно в интернете), могут быть значительными.

Единственным же средством поддержания контекста является *вывод текущего состояния данных в процессе выполнения мастера*. Как правило, обычный текстовый список с предыдущими установленными параметрами работает плохо (к тому же редко вмещается в экран), визуализировать же изменения трудно, если вообще возможно.

Таким образом, *лучше избегать длинных последовательностей.*

Окна

Вывод справочной информации

Благодаря обилию пустого места *мастера замечательно подходят к выводу справочной информации в самом интерфейсе.*

Справочную же информацию нужно выводить двух типов описания текущего шага:

- ☐ краткое
- ☐ более развернутое.

С развернутым описанием все просто. Где-нибудь снизу экрана (чтобы не сбивать фокус внимания пользователей) выводится один или два абзаца, рассказывающие стандартный набор: что, зачем и почему.

С кратким же описанием сложнее. К сожалению, устоявшийся облик мастеров не имеет большого и заметного заголовка (этой проблемы, к счастью, нет в интернете, где вообще нет ничего устоявшегося).

Это неправильно. У каждого окна последовательности должен быть ясно видимый и бросающийся в глаза заголовок. *При этом в отличие от обычных заголовков окна, он должен быть написан не описательно, но командно (сделайте то-то и то-то).*

Microsoft, в некоторых своих продуктах широко использующая мастера вообще рекомендует считать заголовки важнейшими элементами мастеров. Особо подчеркивается, что заголовки экранов должны быть созданы и сформулированы до начала проектирования экранов, при этом содержимое экранов не должно выходить за рамки смысла заголовков. Пospорить с Microsoft в данном случае затруднительно.