

Глава 5

Золотые правила построения интерфейсов

В этой главе мы опишем рекомендации, которые выкристаллизовались за годы изучения проблем человеко-машинного взаимодействия. Многие специалисты считают их наиболее существенными при построении интерфейсов, поэтому мы и назвали их "золотыми правилами". Эти правила могут также использоваться для экспертной оценки существующих интерфейсов.

5.1. Правила Нильсена-Молиха

Вначале сформулируем девять правил Нильсена–Молиха (Nielsen, Molich).

1. **Простой и естественный диалог.** Простота означает, что не должно присутствовать не относящейся к теме или редко используемой информации. Старайтесь добиваться максимального следования задачам пользователя, минимизируя сложность поиска соответствия между семантиками задачи и интерфейсом. Важно представить точно ту информацию, в которой нуждается пользователь, следуя принципу "лучше меньше да лучше". Вся редко используемая информация должна быть спрятана. Естественность означает порядок, соответствующий задаче. Информация, которая выводится на экран, должна появляться в естественном порядке, т.е. в порядке, соответствующем ожиданиям пользователя. Вся взаимосвязанная информация должна группироваться в одном месте.

В одной программе для тестирования студентов была сделана следующая вещь. Перед началом теста студент должен был выбрать свою специальность из выпадающего списка. Каждый элемент списка содержал код специальности и её название. Порядка двухсот специальностей в списке были упорядочены по коду – так было проще сделать, потому что код шёл в начале строки. Но для пользователя-студента, который знал примерное название своей специальности, но был абсолютно не в курсе её кода, такой порядок был просто ужасен. Приходилось последовательно просматривать весь список, прокручиваемый в маленьком окошке, и тщательно выискивать в нём знакомые слова. Это пример несоответствия порядка предоставления информации решаемой задаче.

2. **Говорите на языке пользователя.** Используйте слова и понятия из мира пользователя. Не используйте специфических инженерных терминов. Разработчики одной программы для юристов в США, проводя тестирование интерфейса с пользователями, столкнулись с тем, что те вместо слова "параметр" постоянно употребляли слово "периметр". Термин "параметр" казался совершенно обычным для разработчиков и постоянно использовался в инструкциях к программе, но оказалось, что этого слова нет в словаре юристов (американских). Слово "опция"

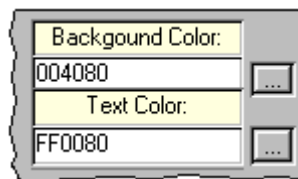
было

им

лучше

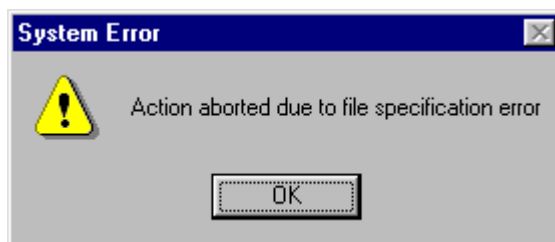
знакомо.

В качестве другого примера приведём такой фрагмент интерфейса:



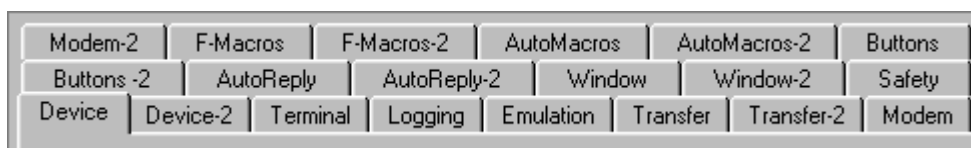
Разработчики не учли тот факт, что для большинства людей представление цвета в виде RGB-компонент и их шестнадцатеричная кодировка совершенно непонятные вещи. Лучшим и довольно простым решением было бы показать вместо кода (или наряду с ним) образец цвета.

Ещё один пример иллюстрирует использование инженерного языка, непонятного пользователю:



Для программиста всё ясно: он проанализировал статус завершения системного запроса и выдал сообщение об ошибке в спецификации файла. Вероятно, пользователь допустил ошибку в имени файла или неверно указал путь, или забыл про расширение. Программисту не важна конкретная причина, но пользователь остаётся в недоумении.

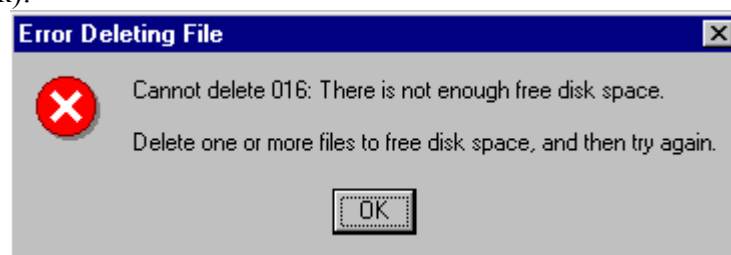
3. **Минимизируйте загрузку памяти пользователя.** Не заставляйте пользователя помнить вещи от одного действия к следующему. Оставляйте информацию на экране до тех пор, пока она не перестанет быть нужной. В этом месте хочется упомянуть о чрезмерном использовании закладок, как это сделано в программе Zoc:



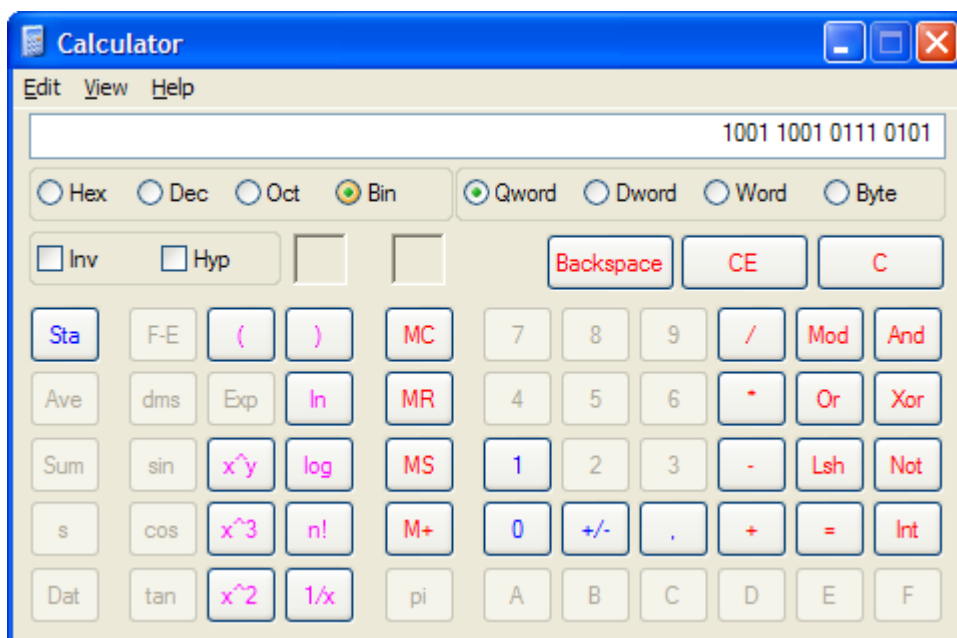
После каждого использования закладки переупорядочиваются и, если вы что-то пытаетесь отыскать, перебирая их, нужно держать в памяти, какие закладки вы уже открывали. Хорошим стилем считается делать только один ряд закладок.

4. **Будьте последовательны.** У пользователей должна быть возможность изучить действия в одной части системы и применить их снова, чтобы получить похожие результаты в других местах. Вспомним о примере с программой CDCору, в которой кнопка с изображением дискеты использовалась не для сохранения файла, как это мог ожидать пользователь, а для преобразования формата.
5. **Обеспечьте обратную связь.** Дайте пользователю возможность видеть, какой эффект оказывают его действия на систему. Мы уже обсуждали проблему обратной связи при анализе интерфейса CDCору.

6. **Обеспечьте хорошо обозначенные выходы.** Если пользователь попадает в часть системы, которая его не интересует, у него всегда должна быть возможность быстро выйти оттуда, ничего не повредив.
7. **Обеспечьте быстрые клавиши и ярлыки.** Элементы быстрого доступа могут помочь опытным пользователям избегать длинных диалогов и информационных сообщений, которые им не нужны.
Мы обсуждали отсутствие ярлыка для быстрого запуска редактора формул в Microsoft Word 2003. Эта проблема, правда, решается при установке плагина MathType.
8. **Хорошие сообщения об ошибках.** Хорошее сообщение об ошибке помогает пользователю понять, в чём проблема и как это исправить.
Вот пример одного из самых курьёзных сообщений об ошибке (для знающих английский язык):



9. **Предотвращайте ошибки.** Всегда, когда вы пишете сообщение об ошибке, вы должны спросить себя, можно ли избежать этой ошибки?
Хороший пример построения интерфейса, предотвращающего ошибки – программа Calc:



При переключении в двоичный режим интерфейс делает недоступными числовые кнопки, кроме 0 и 1, а также делает недоступными некоторые функции, имеющие смысл только для чисел с плавающей точкой. Этот простой приём позволяет минимизировать возможные ошибки пользователя.

В последнее время вдобавок к этим девяти правилам часто формулируют десятое правило: **снабдите программу системой помощи.**

5.2. Принципы организации графического интерфейса

А теперь рассмотрим несколько принципов организации графического интерфейса.

Принцип кластеризации. Организуйте экран в виде визуально разделённых блоков с похожими элементами управления, предпочтительно с названием для каждого блока. Элементы управления включают меню, диалоговые боксы, экранные кнопки и любые другие графические элементы, позволяющие пользователю взаимодействовать с компьютером. Подобные команды должны быть в одном меню: это позволяет им быть визуально близко и идти под одним заголовком. Команды, относящиеся к некоторой конкретной области функциональности, могут также быть показаны в диалоговых боксах, опять таки в визуально определяемых блоках. Тот же самый принцип распространяется на специальные управляющие экраны с многочисленными кнопками и значками, такие как сенсорные панели. Кнопки для конкретной функции должны группироваться вместе, а затем выделяться цветом, обрамлением или окружающим пробелом. Примеры такого подхода можно встретить в любом хорошо разработанном интерфейсе, см., например, диалог "Печать" в Microsoft Word.

Существует две важные причины для кластеризации. Во-первых, она помогает пользователю находить нужную команду. Если вы ищете возможность изменить формат абзаца, то вам легче найти соответствующий диалоговый бокс в меню "Формат", а не в случайно распределённом по экрану наборе из сотни кнопок. Во-вторых, группирование команд позволяет пользователю приобрести концептуальную организацию знаний о программе. Полезно, например, знать что начертание и размер являются атрибутами шрифта, в то время как интервал и отступ – атрибутами абзаца.

Принцип "видимость отражает полезность". Делайте часто используемые элементы управления заметными, видимыми и легко доступными; и наоборот, прячьте или сжимайте редко используемые элементы. Пример реализации этого принципа в современных системах – панели инструментов, т.е. наборы иконок для часто используемых функций. Обоснование этого принципа заключается в том, что человек может быстро находить что-то среди небольшого числа объектов. Для редко используемых функций можно позволить поиск в длинных списках.

Принцип интеллектуальной последовательности. Используйте похожие экраны для похожих функций. Это похоже на заимствование, но в этом случае вы заимствуете что-то из одной части дизайна и применяете это к другой части. Обоснование этого принципа очевидно: раз пользователи выучили расположение элементов управления на экране (например, где находится кнопка "Помощь"), они могут легко приложить это знание к другим экранам внутри той же системы. Этот подход позволяет вам сосредоточить усилия на разработке лишь нескольких привлекательных работоспособных экранов, а затем их слегка модифицировать для использования в других частях приложения. Делайте, однако, это со смыслом: экраны не должны выглядеть одинаково, если в действительности они должны отражать совершенно другие вещи. Предупреждение о критической ошибке в системе реального времени должно иметь вид, значительно отличающийся от экрана помощи или информационного сообщения.

Принцип "цвет как приложение". Не полагайтесь на цвет как носитель информации; используйте его умеренно, чтобы лишь акцентировать информацию, передаваемую

другими средствами. Цвет значительно проще использовать неправильно, чем верно. Для разных людей разные цвета означают разные вещи, особенно эти вариации сильны между различными культурами. Так, красный цвет означает опасность в странах европейской культуры, смерть – в Египте, жизнь – в Индии и счастье – в Китае. Дополнительная проблема в том, что некоторые люди не могут различать цвета: около 7 % взрослых страдают от той или иной формы дальтонизма.

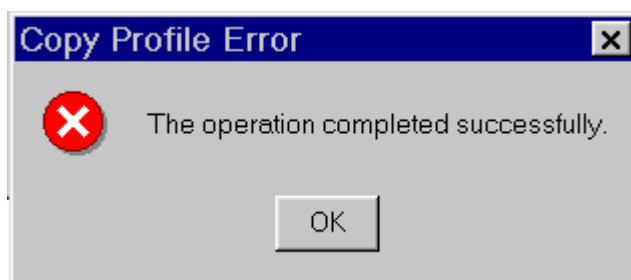
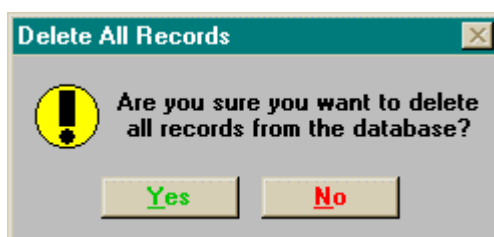
Хорошее правило для большинства интерфейсов – разрабатывать их в чёрно-белом варианте, убедиться, что они работоспособны, а затем добавить минимальный цвет для их окончательного облика. Цвет, безусловно, полезен, когда необходимо выделить предупреждение или информационное сообщение, но обязательно дайте дополнительные ключи для пользователей, не способных воспринимать изменения цвета.

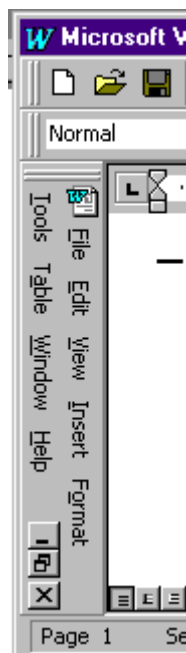
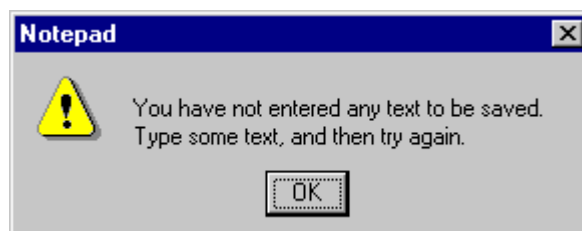
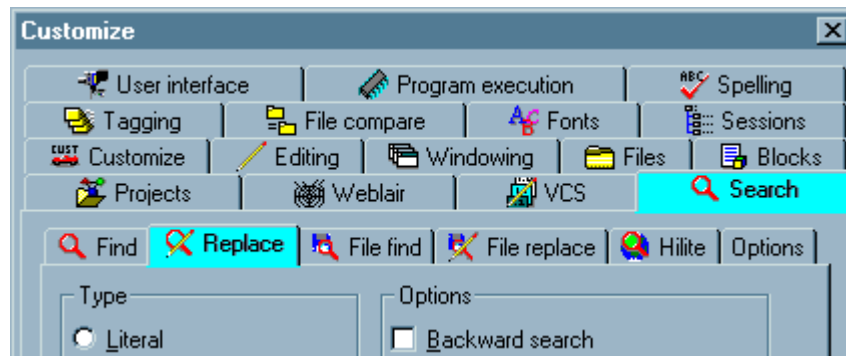
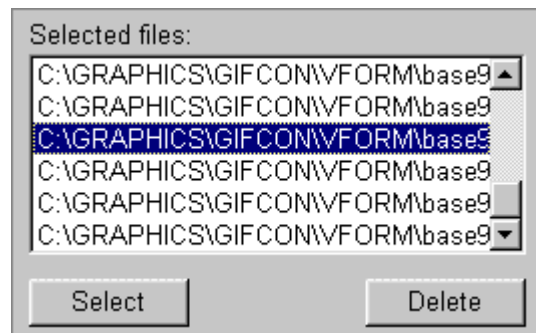
Если только вы не опытный графический дизайнер, придерживайтесь принципа минимума цвета для создания привлекательного интерфейса. Постарайтесь придерживаться серых тонов в большей части системы, дайте немного ярких красок для логотипа или метки для придания индивидуальных отличий вашему продукту. Помните, что многие пользователи могут, и часто это делают, изменить цвет окон, подсветок и других системных объектов. Стройте ваш продукт так, чтобы он работал с пользователем, а не боролся с ним.

Принцип уменьшения беспорядка. Не помещайте на экран слишком много всего. Этот неформально определяемый принцип – хорошая проверочная точка, чтобы подтвердить, что ваш дизайн отражает перечисленные выше принципы. Если видимы только наиболее часто используемые элементы, все они сгруппированы в небольшое число визуальных кластеров, использован минимум цвета, то экран должен получиться графически привлекательным.

Это также хороший принцип для вещей, с которыми мы особенно не имели дела. Например, тип и размер шрифта: принцип уменьшения беспорядка предполагает, что одного или двух стилей вполне достаточно. Не пытайтесь наделять каждое меню собственным шрифтом или работать с большим набором размеров. Как правило, пользователи заметят не столько различия, сколько беспорядок.

В заключение приведём несколько плохих примеров, нарушающих принципы, изложенные в этой главе. Хорошие примеры у всех перед глазами. ОС Windows, как бы к ней ни относиться, в огромной степени соответствует принципам хорошего дизайна интерфейса.



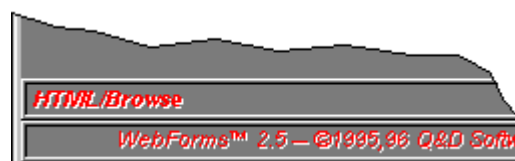


Whenever your local SMS Administrator sends you an actual software Package, the SMS Package Command Manager will appear (usually at network login time) displaying the available Package(s). The following screenshots display scenes similar to what you will see when you receive an actual SMS Package.

To start the demonstration, click the "CLICK HERE TO CHANGES" button of the screen.



Form Title -- (appears above URL in most browsers and is used by WWW search)		Background Color:
Q&D Software Development Order Desk		FFFBF0
Form Heading -- (appears at top of Web page in bold type)		Text Color:
Q&D Software Development Order Desk <input checked="" type="checkbox"/> Center		000080
E-Mail responses to (will not appear on)	Alternate (for mailto forms only)	Background Graphic
dversch@q-d.com		
Text to appear in Submit button	Text to appear in Reset button	<input type="radio"/> Mailto
Send Order	Clear Form	<input checked="" type="radio"/> CGI
Scrolling Status Bar Message (max length = 200 characters)		
****WebMania 1.5b with Image Map Wizard is here!****		
<< Prev Tab		Next Tab >>



Думається, коментарии излишни.