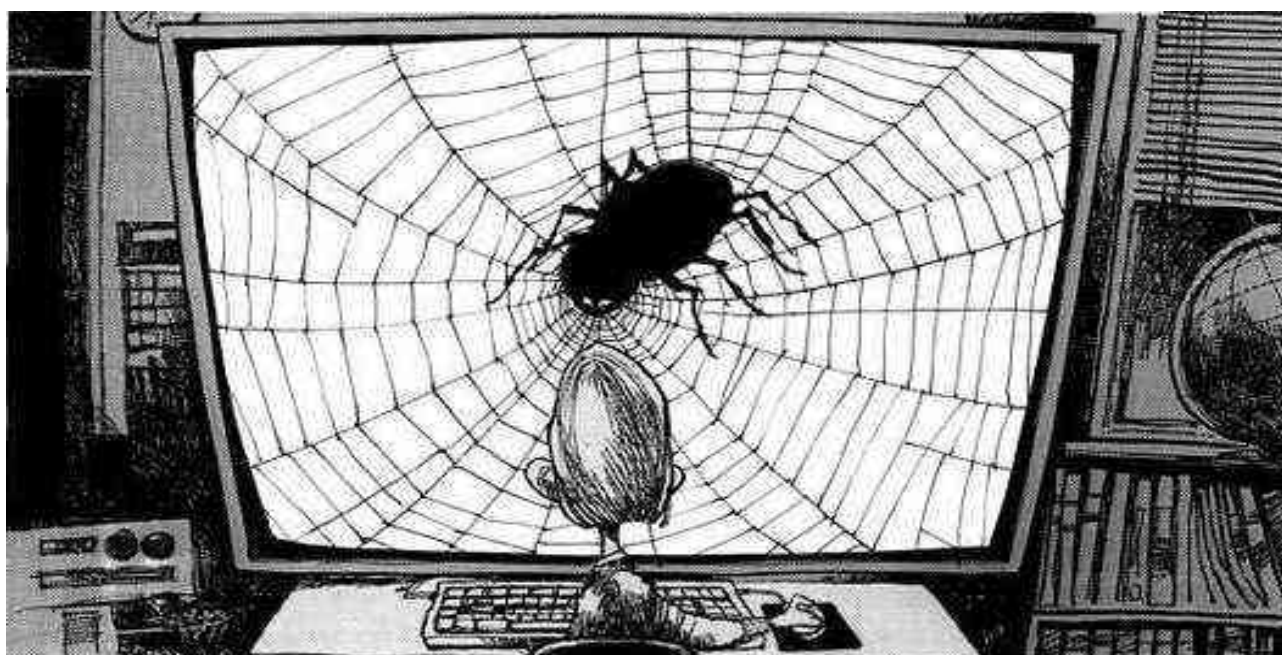




Г.Г. Массель

**ПСИХОЛОГИЧЕСКИЕ АСПЕКТЫ
ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА
СОВРЕМЕННЫХ КОМПЬЮТЕРНЫХ СИСТЕМ**



Массель Г.Г. Психологические аспекты пользовательского интерфейса современных компьютерных систем / Под ред. Л.В. Массель / ИСЭМ СО РАН. Препр. № . Иркутск, 2000. - 42 с.

Предлагаемый препринт вводит в проблему разработки интерфейсов пользователя с учетом психологических аспектов. Рассматриваются принципы проектирования и критерии эффективности интерфейсов, а также использование элементов дизайна: цветовое оформление, применение шрифтов и эффектов. Большая часть ссылок относится к источникам, взятым из Internet.

Препринт ориентирован на прикладных программистов – разработчиков пользовательских интерфейсов проблемно-ориентированных программных комплексов. Может быть полезен также специалистам, работающим в области информационных технологий, разработчикам программного обеспечения, студентам и аспирантам.

Содержание

1. Введение	4
2. Принципы проектирования и критерии эффективности интерфейсов	5
2.1. Основные принципы проектирования интерфейсов	5
2.2. Обучаемость пользователя	6
2.2.1. Исследуемые интерфейсы	7
2.2.2. Инструкции для пользователя	10
2.3. Использование метафор	14
2.4. Компоновка интерфейса	16
2.4.1. Организация пространства	16
2.4.2. Расположение на экране элементов управления	19
2.4.3. Расположение информации на экране	23
2.5. Окна диалогов	26
2.5.1. Использование различных видов диалогов	26
2.5.2. Окна диалогов с закладками	27
2.6. Выбор терминологии	30
3. Эффективность работы пользователя	31
3.1. Уменьшение задержки	31
3.2. Удобочитаемость	32
4. Элементы дизайна	33
4.1. Применение шрифтов и эффектов	33
4.2. Цветовое оформление	38
5. Заключение	40
6. Литература	41

1. ВВЕДЕНИЕ

«Он очень – очень способный, ужасно умный! Но с ним всегда так сложно, очень сложно, иногда – ну просто невозможно!..» - говорили туземцы о компьютере.

Андрей Кнышев «Уколы пера»

При разработке программного обеспечения, в первую очередь прикладных программ, теоретически, на первом месте для программиста должны быть интересы пользователя. Тем не менее часто получается, что программист пишет программу «для себя», или, вернее, «под себя». В результате логика работы с программой не понятна неподготовленному пользователю, вследствие чего он отказывается с ней работать. Программа может иметь гениальный алгоритм, очень быстро работающий код и занимать меньше места по сравнению с аналогичными программами, но эти моменты будут интересовать только программистов, потому что будут понятны только им и только ими будут оценены должным образом.

Пользователь предъявляет совершенно иные требования, чем программист, так как смотрит на программный продукт «снаружи», а не «изнутри». И, соответственно, большей популярностью будет пользоваться та программа, с которой пользователи «нашли общий язык», остальные параметры интересуют пользователей во вторую очередь. Таким образом, главное, что необходимо пользователю для работы с программой - это простота и понимание того, что программа делает.

Для обеспечения эффективной работы пользователя желательно учитывать его эмоциональные, психологические и физиологические особенности. Когда человек расстроен, раздражен или подавлен, он не сможет работать хорошо. Элементом программы, способным вызвать, или, наоборот, снять стресс,

является *пользовательский интерфейс*, т.е. среда, через которую пользователь взаимодействует с программной системой.

При проектировании интерфейсов приходится учитывать не столько физические ограничения, сколько *психологические*, которые часто вообще игнорируются. Разработчик интерфейса, применяющий психологические знания, может сделать удобной работу совершенно незнакомых ему пользователей, создавая интерфейс, удовлетворяющий не только требованиям с точки зрения вычислительных задач, но и удобный с точки зрения физических и психологических потребностей пользователя.

Таким образом, интерфейс пользователя - отдельный и очень важный компонент любой вычислительной системы, требующий предварительного проектирования. Вопросы разработки пользовательских интерфейсов становятся особенно актуальными в условиях распространения концепции открытых систем [1], в которых одним из обязательных требований является дружелюбность интерфейса.

2. ПРИНЦИПЫ ПРОЕКТИРОВАНИЯ И КРИТЕРИИ ЭФФЕКТИВНОСТИ ИНТЕРФЕЙСОВ

2.1. ОСНОВНЫЕ ПРИНЦИПЫ ПРОЕКТИРОВАНИЯ ИНТЕРФЕЙСОВ

Мы будем понимать под *пользовательским интерфейсом* программы совокупность элементов, позволяющих пользователю программы управлять ее работой и получать требуемые результаты. Фактически пользовательский интерфейс - это канал, по которому осуществляется взаимодействие пользователя и программы.

Существуют некоторые принципы, являющиеся фундаментальными при разработке и реализации эффективных интерфейсов, как традиционных графических, так и Web-интерфейсов. Достаточно часто приходится сталкиваться с недопониманием многих из этих принципов, что, в конечном счете, отражается

на эффективности работы программного продукта. Перспективы использования приложения в Internet делают применение этих принципов еще более важными. В числе первоочередных выделяют следующие принципы:

1. Эффективные интерфейсы должны быть очевидными и внушать своему пользователю чувство контроля. Необходимо, чтобы пользователь мог одним взглядом окинуть весь спектр своих возможностей, понять, как достичь своих целей и выполнить работу.

2. Эффективные интерфейсы не должны беспокоить пользователя внутренним взаимодействием с системой. Необходимо бережное и непрерывное сохранение работы, с предоставлением пользователю возможности отменить любого действие в любое время.

3. Эффективные приложения должны выполнять максимум работы, требуя при этом минимум информации от пользователя.

Следствие. Приложения должны пытаться предположить то, чего хочет пользователь. Не следует заставлять пользователя заниматься поиском информации или вызовом необходимых средств, желательно предоставлять ему доступ ко всей информации и всем средствам на каждом шаге [2].

Единственный путь установить предпочтения пользователя - это тестирование. Анализ или рассуждения «с позиций пользователя» могут быть применены на предварительных этапах, но не могут заменить тестирование.

В соответствии с перечисленными принципами определяются такие критерии эффективности интерфейсов, как обучаемость пользователя, использование метафор, удобная компоновка элементов интерфейса, окна диалогов, применяемая терминология.

2.2. ОБУЧАЕМОСТЬ ПОЛЬЗОВАТЕЛЯ

В идеале программные продукты не должны иметь кривую обучения: пользователи хотели бы сразу начинать работать с системой и быстро дости-

гать определенного мастерства. Но на практике все программы, независимо от сложности, имеют кривую обучения.

Чтобы пользователь понял, для чего нужна программа, желательно уже в названии отразить суть и назначение программы, а в окне «About» («Описание программы») изложить это более подробно. Если пользователь будет иметь предварительные сведения о назначении программы, то ему будет проще разобратся, как с ней работать.

При построении интерфейса важно угадать привычки пользователя и его реакцию (действия) при первом контакте с программой. Можно предположить, что вначале он будет пытаться понять, как добиться от программы того, что ему нужно. Надеяться на то, что пользователь будет читать документацию, которая часто написана непонятно, не стоит. На HELP тоже рассчитывать не стоит, так как и ее читают редко. Поэтому очень важно, чтобы рабочий интерфейс компоновкой внешних элементов подсказывал пользователю, как и что делать. Иными словами, интерфейс должен быть исследуемым, то есть должен допускать возможность его изучения путем имитации пользователем различных ситуаций [3].

2.2.1. Исследуемые интерфейсы

Для обеспечения безопасности и целостности пользовательского интерфейса в него необходимо вводить четкие указания для пользователя и предусматривать возможность нелогичных действий с его стороны (в программистском сленге для обозначения этого есть термин «дуракоустойчивость» или «защита от дурака»). На самом деле, нежелательные последствия – это, конечно, в первую очередь результат непредусмотрительности программиста. Необходимо продумывать возможные стратегии поведения пользователя, чтобы не позволять ему «попадать в ловушку» (например, неожиданный выход из системы) или идти единственным путем: у него должен быть хотя бы небольшой выбор (например, чтобы не повторять каждый раз уже отработанную цепочку). Это позволит как новичку, так и пользователю, выполняющему работу, быстро

пройти дальше. Одновременно это обеспечит возможность человеку, который захочет исследовать интерфейс, поиграть во «что – если».

Рассмотрим подробнее требования, которым должны соответствовать исследуемые интерфейсы.

1) *Не следует* помещать в главное меню пункты, срабатывающие сразу после нажатия. Пользователи привыкли, что любой пункт главного меню в программах всегда содержит «подменю». Помещение в главное меню пункта, который срабатывает сразу при нажатии на него, делает программу непредсказуемой. Чаще всего это бывает пункт «Выход». Разработчики программ, видимо, испытывают трудности с его размещением в меню. На самом деле «Выход» из программы лучше располагать самым нижним пунктом левого подменю, независимо от того, подходит он туда по смыслу или нет. Пользователи начинают искать его в первую очередь именно здесь.

2) *Делайте действия обратимыми.* Люди исследуют интерфейс не только с помощью навигации. Иногда они хотят знать, что случится, если они сделают что-то потенциально опасное. Иногда они не хотят этого, но такое может произойти случайно. Если действия обратимы, пользователь может исследовать интерфейс и не бояться за последствия.

3) *Всегда предоставляйте отмену (undo).* Если Ваша программа не предоставляет возможности отмены, Вам неизбежно придется создавать массу диалогов с вопросами типа «Вы действительно уверены?». Это, безусловно, замедляет работу, но отсутствие таких диалогов (если Вы опустили «отмену») будет замедлять работу еще больше. Специальные исследования показали, что люди в опасной среде делают не больше ошибок, чем в отзывчивой и более ясной среде, но работают гораздо медленнее и осторожнее, чтобы избежать необратимых ошибок.

4) *Всегда предоставляйте запасной выход.* Пользователи не должны чувствовать себя пойманными в ловушку. Путь назад должен всегда быть четко обозначен. Однако делайте так, чтобы остаться в программе было легче. Более

ранние программы создавались так, чтобы их трудно было покинуть. С приходом Internet чаще появляются программы, в которых трудно остаться. Браузеры напоминают гирлянды объектов и опций, которые не имеют ничего общего с самой программой. Если пользователь имеет на экране набор элементов, в котором 49 элементов ведут напрямую к разрушению работы и только один-два способны реально помочь – такой интерфейс не является исследуемым.

5) *Предлагайте пользователю постоянные наглядные подсказки*, чтобы он чувствовал себя «как дома» (очень эффективны «всплывающие» подсказки). Стабильные визуальные элементы не только позволяют людям быстро перемещаться, но и работают как указатели, давая пользователю ощущение психологического комфорта. Если в ставший привычным пользовательский интерфейс начать вносить некоторые изменения, то его, как и любой язык в такой ситуации, будет труднее понимать. Люди в большинстве своем консервативны по природе, не любят отказываться от привычек и привыкать к чему-то новому, особенно когда этого нового много. Этот фактор необходимо учитывать. Поэтому, если разрабатывается программа, аналог которой уже существует и которым пользуются многие, то имеет смысл посмотреть, как организовано взаимодействие с пользователем в уже существующей программе, и позаимствовать ее основные принципы, тогда новая программа не вызовет отторжения, так как сделана в привычном для пользователя стиле. Она будет более привычной для пользователя, а, главное, ему будет легче перейти к новой программе, если она лучше аналогичной.

5) *Стремитесь к единообразию интерфейса в разных подсистемах (частях программы)*. Необходимо тщательно продумать и осознать схему программы, приведя ее к системе (структуре, выстроенной по определенным правилам), и реализовать пользовательский интерфейс в соответствии с этой системой. Пользователю достаточно будет «схватить» суть, чтобы по аналогии разобраться в остальном. Следует избегать лишних элементов: чем меньше их будет, тем проще пользователю будет разобраться.

6) *Обеспечивайте защиту работы пользователя.* Удостоверьтесь в том, что пользователь никогда не потеряет свою работу в результате либо своей ошибки, либо неустойчивости соединения, или по какой-то другой причине, кроме неизбежных, таких, как неожиданная потеря питания [2].

2.2.2. Инструкции для пользователя

Одним из критериев быстрой обучаемости работе с программой может служить возможность оказания помощи пользователю при его работе с системой. Инструкции для пользователей обычно выводятся в виде подсказок либо справочной информации. Характер и количество инструкций должны соответствовать опыту работы пользователя с системой и его намерениям. Справочная информация должна появляться тогда, когда она требуется, и в приемлемой форме. Основное внимание обращается на количество и качество имеющихся инструкций; характер выдаваемых сообщений об ошибках; подтверждение каких-либо действий системы.

Сообщения об ошибках

Сообщения об ошибках часто агрессивны и грубы, останавливают текущую работу и требуют подтверждения, прежде чем позволят продолжить ее. С другой стороны, многие сообщения об ошибках туманны, не способны довести значимую информацию и по временам попросту неверны, в потенциале принося незаслуженные огорчения и моральный ущерб пользователю. Сообщение об ошибке должно точно объяснить, в чем заключается ошибка и какие действия следует предпринять, чтобы ее устранить, а не ограничиваться общими фразами типа «синтаксическая ошибка» или таинственными кодами типа «OS1» [4].

У людей есть эмоции и чувства - у компьютеров их нет. Когда один кусок кода отклоняет ввод другого, последний не хмурится и не страдает. Процессору все равно, даже если вы выключите компьютер.

Каждый опытный программист знает, что если модуль А передал ошибочные данные модулю В, модуль В должен сразу же отбросить эти данные, сгенерировав подходящее сообщение об ошибке. Отсутствие такого взаимодействия означает серьезную ошибку в проектировании модулей. Но люди не модули программы, поэтому разработчик должен переосмыслить само понятие «ошибочных данных». Когда данные получены от человека, программа должна предположить, что они правильные, потому что человек важнее, чем программа. Вместо того, чтобы отклонить ввод, программа должна его проанализировать и понять. Программа может предотвратить возможные проблемы, гарантируя невозможность введения ошибочных данных.

Чаще всего сообщение об ошибке - это явно выраженное извещение о том, что разработчик приложения оказался не способен предвидеть потребности или возможности пользователя.

Ничто из того, что происходит в компьютере, не может быть достаточной причиной для оправдания унижения человека. Не имеет значения то, насколько важна для вас целостность базы данных, она не стоит того, чтобы оскорблять пользователя. Если целостность данных такая важная вещь для вас, вы должны позаботиться о методах ее поддержания без унижения пользователя.

Существуют несколько способов избежать сообщений об ошибках: 1) сделать ошибки невозможными; 2) дать позитивную обратную связь; 3) проверить, но не редактировать [4].

Наилучший способ избежать сообщений об ошибках - это сделать так, чтобы пользователь не смог их совершить. Вместо требования ввести данные с клавиатуры предоставьте пользователю список возможных вариантов выбора, например, вместо требования ввести код региона вручную, предоставьте пользователям возможность выбрать из списка доступных кодов регионов или даже карту.

Прекрасный способ избежать сообщений об ошибках - это сделать программу достаточно умной для того, чтобы избежать лишних вопросов к поль-

зователю. Многие сообщения об ошибках говорят что-то вроде «Неверные данные. Пользователь должен ввести XXXX». Почему же программа не может, зная, что должен ввести пользователь, ввести XXXX сама? Вместо того, чтобы запрашивать у пользователя имя файла на диске (давая возможность задать несуществующее имя), программа может запомнить, с какими файлами велась работа в последний раз, и предоставить их список.

Еще один метод избежать сообщений об ошибках для программы, когда пользователь вводит неправильные данные, состоит в том, чтобы принять, что они неправильные, потому что программа, а не пользователь плохо информирована.

Если, например, пользователь вводит счет-фактуру для несуществующего номера клиента, программа может принять эти данные и сделать себе специальную заметку, которая показывает, что пользователь должен исправить ее. Затем она будет наблюдать, чтобы удостовериться, что пользователь ввел необходимую информацию до конца отчетного периода. Так в действительности работает большинство людей. Они не вводят «неверных» номеров. Просто люди обычно вводят информацию в такой последовательности, которую программа принять не может.

Разработчики предполагали, что запись о клиенте должна уже существовать перед выпиской счета-фактуры, но это не догма. Программа может воспринимать счета-фактуры независимо от записей о клиентах и попросту считать, что недостающая информация будет введена позже. Если человек забывает ввести недостающую информацию до конца месяца, программа может переместить незаконченные документы на неопределенный счет. Программа не должна прерывать работу и останавливаться с сообщением об ошибке. В конце концов, все документы на неопределенном счете наверняка будут составлять лишь малую долю всего объема продаж и, таким образом, не будут являться значительным фактором для бизнеса.

Сообщения, подтверждающие какое-либо действие системы

Сообщения, подтверждающие какое-либо действие системы, требуются для того, чтобы пользователь мог еще раз убедиться в том, что система выполнила или будет выполнять требуемое действие.

Подтверждать ввод данных следует тогда, когда эти данные приведут к необратимым действиям системы (например, удаление записи из файла) или когда вводится код и пользователю необходимо проверить по соответствующему описанию, нужна ли запись выбрана (например, о конкретном энергоресурсе или административной области). Также может оказаться желательным подтверждение, что было выполнено некоторое действие (например, добавлена запись в файл). Может оказаться полезным не только подтверждение выполнения какого-то действия, но и вывод сообщения о состоянии системы (чтобы пользователь знал, где он находится). Это можно реализовать, отведя на экране специальную область, в которой будет отображаться путь, «пройденный» пользователем до текущего состояния [5].

Механизм статуса используется, чтобы держать пользователя в курсе того, что происходит. Управление программой не может быть применено без наличия значительной информации. Механизмы статуса жизненно необходимы для предоставления информации человеку, чтобы он мог соответственно реагировать на изменение условий. Простой пример: человек, не получая информации о статусе, в течение периодов ожидания, пока процесс действительно не завершится, будет напрягаться и уставать без необходимости, так что в следующий раз у него может не хватить физических и умственных ресурсов на это.

Информация о статусе должна быть актуальной и легкодоступной. Пользователи не должны искать информацию о статусе. Они должны иметь возможность, одним взглядом окинув свою рабочую среду, получить хотя бы приближенное понятие о ходе работы и состоянии системы. Информация о статусе может быть, например, такой: иконка папки «Входящие» может иметь три со-

стояния - пустая, наполненная и переполненная. Однако здесь нельзя пересердствовать. Иконка «мусорной корзины» на Макинтоше долгие годы изображала неминуемую опасность взрыва, когда хотя бы один документ находился внутри, так что люди старались очищать корзину сразу же после каждого документа. Это не только превратило одношаговую операцию в двухшаговую (перетащить в корзину, очистить корзину), но и лишило смысла саму идею мусорной корзины - возможность отмены удаления.

2.3. ИСПОЛЬЗОВАНИЕ МЕТАФОР

В разработке интерфейса часто употребляются *метафоры*, чтобы помочь пользователю познакомиться с приложением, используя ассоциации с уже имеющимся знанием. Некоторые из наиболее известных - это метафора *бухгалтерской книги* VisiCalc, метафора *рабочего стола*, впервые примененная Хегох, и метафора *чековой книжки* из Quicken.

Метафоры понимаются интуитивно. Люди схватывают смысл метафорического элемента управления в интерфейсе, мысленно отождествляя его с каким-либо другим процессом или предметом, на познание которого они уже затратили время и силы. Эффективность этого метода велика, потому что она использует такое свойство человеческого ума, как способность делать логические выводы (выводы «по аналогии»). Слабая сторона метафор заключается в том, что у человека, который с ней сталкивается, может не быть знаний или логических способностей, необходимых для совершения отождествления. Метафоры не могут нести ответственность за то, как их понимают. Правильно выбранные метафоры позволяют пользователям быстро ухватить все детали концептуальной модели. Неправильно приложенные метафоры только ухудшают пользовательскую ценность приложения. Поэтому следует очень внимательно относиться к выбору той или иной метафоры [6].

Метафоры обычно являются чем-то знакомым, но часто добавляют и новые свойства. Например, в Windows95 есть объект «Портфель». Как и настоя-

щий портфель, его цель - помочь нам сделать электронные документы более переносимыми. Он так и делает, действуя, однако, не как транспортировщик, а как синхронизатор: документы в портфеле на рабочем столе и в портфеле на дискете обновляются автоматически при вставке дискеты в дисковод.

Преимущества использования метафор:

- пользователю легче понимать и интерпретировать изображение на экране;
- не нужно каждый раз заглядывать в руководство, чтобы узнать, как выполняется то или иное действие, по крайней мере, некоторые действия должны естественно следовать из метафоры;
- у пользователя возникает чувство психологического комфорта, характерного для встречи с чем-то знакомым.

Метафора выбирается (разрабатывается), как правило, в процессе концептуального дизайна интерфейса, который включает три основные задачи:

- 1) разработка системы интерфейсных элементов, своего рода алфавита взаимодействия, изучив который пользователь сможет легко выполнять необходимые действия;
- 2) выбор способа изображения как отдельных элементов, так и их групп;
- 3) выбор общего изобразительного стиля, легко узнаваемого и приятного для глаз [7].

Концептуальный дизайн интерфейса должен базироваться на идее *интерфейсной среды*. В сущности, на время работы с системой пользователь погружается в среду интерфейса подобно тому, как, приехав в лес, человек погружается в среду дикой природы.

Понятия среды и метафоры близко связаны. Если среда по виду будет напоминать пользователю что-то знакомое, он сможет быстрее приспособиться к ней. Метафора может продиктовать все изобразительные решения интерфейса. Но следует опасаться фотографической схожести интерфейсной среды и выбранной метафоры. Все же компьютерная среда искусственна, и полностью

повторить все элементы взаимодействия из физического мира не удастся. Фотографическая похожесть может спровоцировать пользователя на то, чтобы пользоваться этой средой в точности так, как той, которую она напоминает. В первый же раз, когда пользователь наткнется на различие, он испытает тяжелый психологический шок, который может привести к полному отторжению системы. В этом секрет непопулярности многих компьютерных игр с прекрасным изобразительным рядом, тогда как популярные «Тетрис» или «Lines» имеют очень простую условную среду, обеспечивающую психологический комфорт пользователя.

Еще один важный принцип построения дизайна интерфейса – баланс между интерактивными возможностями программы и сложностью ее изобразительного ряда. В интерфейсе необходимо обеспечить баланс между функциональными возможностями программы, возможностями манипуляции ею и ее изобразительным рядом. Простая программа не должна сложно управляться или сопровождаться изощренной графикой. Сложная картинка может предупреждать о сложности программы (что не означает обязательного наличия у сложной программы изощренной графики и сложных путей взаимодействия). Это правило может нарушаться в программах, предназначенных для профессиональной деятельности. Сложность лучше показывать постепенно, подобно наращиванию уровней в компьютерных играх.

2.4. КОМПОНОВКА ИНТЕРФЕЙСА

2.4.1. Организация пространства

Наше восприятие определяется особенностями высшей нервной системы, психики, содержанием сознания. Кроме обстановки, в которой происходит процесс восприятия, на него оказывают активное влияние социально-демографические характеристики человека, его образовательный, культурный уровень, предшествующий жизненный опыт, степень знакомства с объектом восприятия [8].

Создавая искусственную среду, человек использует такие принципы организации мира (по аналогии с преобразованием окружающего мира), как ритм, симметрия, пропорции, контраст и др. Природные и созданные человеком объекты действительности порождают в его сознании устойчивые образы, сопровождаемые определенными эмоциями. Эти символические ассоциативные значения геометрических фигур, линий составляют алфавит иконических знаков пространства внешнего мира.

Матричные структуры зрительного анализатора предпочитают воспринимать геометрические фигуры типа круг, крест, квадрат. Этот врожденный механизм позволяет всем индивидуумам осуществлять семантическую обработку наиболее значимой информации молниеносно и всегда бессознательно. На основе этого алфавита иконических знаков, закрепленного эволюционно, формируется персонифицированный внутренний мир понятий, важных для восприятия пространства. Иконические знаки древней символики, такие, как черта, крест, круг, квадрат, по существу, являются готовым инструментом для гармонического деления пространственно воспринимаемых форм [9].

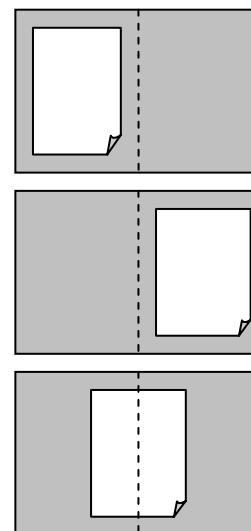
Для европейцев характерно читать слева направо, благодаря этой моторно закрепленной привычке глаза человека и по экрану движутся так же. Как результат, левая часть экрана воспринимается связанной с прошлым, с тем, что уже было прежде. Правая часть экрана связана с будущим, с завершением. Середина же ассоциируется с текущим моментом.

Пример

Смещение акцента влево от середины будет восприниматься как движение навстречу зрителю. Нагруженная левая часть создает ощущение движения вспять, в прошлое, навстречу «нормальному» ходу событий. Перенос центра масс в левую часть экрана может вызвать иллюзию вращения композиции против часовой стрелки.

Нагруженная правая часть экрана воспринимается как «убегание» от зрителя в пространстве или устремленность в будущее - во времени. В любом случае такой вариант композиции трактуется как девиз «Вперед!» Массивная правая часть может создать иллюзию вращения по часовой стрелке.

Акцент на середину экрана сосредоточивает внимание на текущем моменте и воспринимается как «здесь и сейчас».



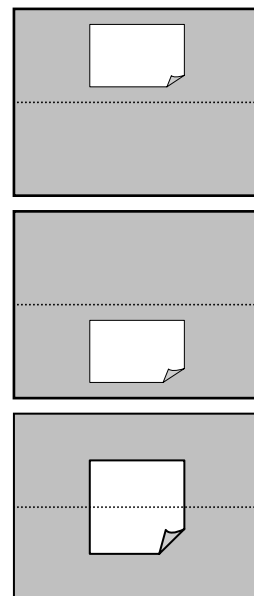
Горизонталь, проходящая по середине экрана, отождествляется с линией горизонта, когда человек стоит на горизонтальной поверхности, а его взгляд параллелен земле.

Пример

Преобладание массы в верхней части экрана создает впечатление неустойчивости, смутной угрозы, эмоциональное ощущение напряженности. Объяснение этого эффекта заключается в подсознательном опасении, что нависающая над головой масса может обрушиться.

Массивная же нижняя часть экрана, наоборот, дает ощущение статичности, основательности, надежности и стабильности.

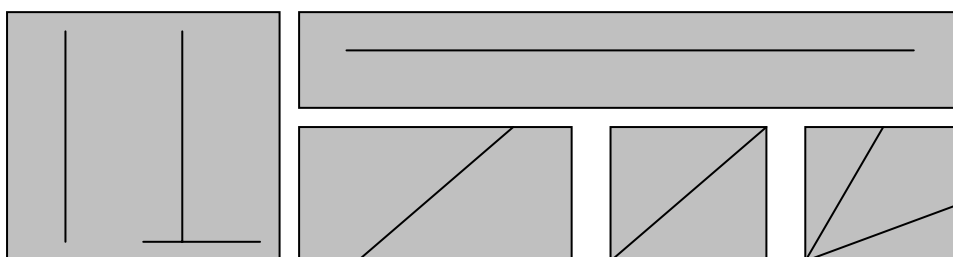
Размещение центра масс по середине экрана вызывает ощущение сбалансированности, уравновешенности, усредненности. Словесные выражения, соответствующие такому положению, «здравый смысл» или «обыденная повседневность».



Вертикаль при отсутствии членений воспринимается как нечто несоизмеримое, бесконечное, легкое, устремленное ввысь, если она имеет утолщения в нижней части или стоит на горизонтальном основании – как нечто более устойчивое. Горизонталь ассоциируется с надежностью, стабильностью.

Диагональ символизирует динамику. Наиболее мощное и энергичное движение передает диагональ «из угла в угол». Чем ближе к прямому углу между диагональю и горизонтальным основанием, тем более резкое и порывистое движение она передает. Приближение к вертикали делает этот рывок все короче, судорожнее, сближая его с напряженным стоянием вертикали. Сближение же с горизонталью передает движение более протяженное, медленное и спокойное.

Пример



Соответственно, композиция, построенная на четких горизонталях и вертикалях, тяготеет к устойчивости, статичности, торжественности, а композиция, базирующаяся на диагоналях, - к движению, изменчивости, нестабильности. Горизонтальная композиция будет выглядеть более основательной, тяжеловесной, чем вертикальная. Справедливость этого утверждения в большой степени зависит и от соотношения сторон прямоугольников, и от размещения текста в них [10].

2.4.2. Расположение на экране элементов управления

Очень *распространенная ситуация*: имеется поле с элементами, над которыми необходимо производить некоторые операции, например, поле ввода текста в окне и управляющие кнопки к нему, либо поле для рисования и набор инструментов к нему и т.д. В этой ситуации возможны различные варианты расположения кнопок относительно поля.

Хотя непосредственное управление программой делается не руками, а мышью, это не влияет на особенности человеческого восприятия. Поэтому было бы более естественно располагать управляющие элементы снизу или справа, то есть так, как они располагаются в реальном мире. Однако, в силу ряда причин, кнопки стали размещать не совсем естественно: сверху и слева. Так за «быстрыми» кнопками закрепилось положение сверху, как и меню. Слева, в некоторых случаях, размещают инструменты и прочие управляющие элементы. Снизу и справа остались лишь управляющие кнопки типа ДА, НЕТ, ОТМЕНА, СПРАВКА и т.д. Желательно придерживаться этой схемы, так как она уже стала привычной для многих пользователей, либо сделать так, чтобы панели управления были «съемными» и могли перемещаться по желанию в произвольные места. Однако предоставлять пользователю слишком большие возможности по конфигурированию интерфейса тоже не следует, так как это может привести к тому, что пользователь просто запутается и испытает определенный дискомфорт от внесенных им изменений (в конечном итоге он придет к мысли

о том, что интерфейс, сконфигурированный разработчиком программы, все равно был лучше, так как разработчику лучше известно, «как надо»).

Другая ситуация: текст, поясняющий поле, и само поле. Здесь однозначно: текст слева, поле справа, других вариантов нет. Уточнение: Checkbox и Radiobutton - особый случай, так как поясняющая надпись фактически является продолжением поля, и их относительное расположение будет корректным [3].

Расположение большого числа элементов

Как правило, такая ситуация возникает при необходимости настройки программы (Options). Здесь, по крайней мере, следует отметить случай, когда на площади окна не помещаются все необходимые элементы и приходится использовать страницы.

Примером не очень удачного решения может служить элемент типа Pagecontrol (так он называется в Delphi) в Windows, состоящий из страниц корешками вверх. Суть этого решения состоит в том, что если корешки не помещаются на одной линии, то они располагаются в нескольких рядах и выбор какой-либо страницы приводит к перестановке этих корешков, так что пользователь путается и не помнит, на какой странице он был (названия страниц, как правило, не запоминаются за ненадобностью). При использовании этого элемента (или любого аналогичного) желательно уместить все корешки в одну линию. С точки зрения «правильности» расположения корешков элемент с верхним расположением предпочтительней для людей, работающих (или работавших) с картотекой. Если картотеку не учитывать, то правильнее было бы использовать элемент переключения страниц с нижним расположением корешков, так что какой элемент использовать - дело вкуса и особенностей контингента людей, для которых предназначена программа. Наверное, не стоит напоминать, что форматирование на всех страницах должно быть единообразным (т.е. привязанным к неявной таблице), ничто так не раздражает пользователя, как явные «ляпы» [3].

Меню

Известно, что объем человеческой памяти составляет 7 ± 2 сущности. Поэтому меню, содержащее число пунктов, не превышающее эти цифры, будет более удобным для пользователей [11]. Это довольно гибкая концепция, так как сущности могут быть как «маленькими», так и «большими». Это свойство может быть очень полезным, так как Вы сможете помочь пользователям организовывать информацию в сущности, удобные для запоминания, просто группируя ее в отдельные блоки.

В меню, если нельзя назвать первый пункт «Файл», то следует назвать его, исходя из соображений системности, например, в качестве первого элемента меню можно выбрать тот предмет (объект), на работу с которым нацелена программа. Желательно придерживаться общепринятого, как уже говорилось выше, порядка расположения пунктов меню и порядка расположения элементов в самих пунктах. Пользователь должен, ничего не зная о программе, быстро найти в меню то, что ему нужно. Например, пробуя что-либо сделать, пользователь знает, что отменить свое действие можно с помощью команды Edit / Undo. Названия пунктов тоже следует подбирать стандартные, чтобы не запутать пользователя и не заставлять его менять привычки.

Быстрые клавиши, такие, как «Ctrl+v» и др., тоже лучше использовать так, как привыкли пользователи.

Быстрое меню, вызываемое обычно с помощью правой кнопки мышки, также следует делать с учетом привычек. Так, если в программе предусмотрена работа через буфер (cut, copy, paste и др.), то следует в соответствующем порядке разместить эти функции в быстром меню, поскольку его все равно по привычке будут вызывать при копировании текста [3].

Кнопки

Кнопки должны быть хотя бы примерно стандартных размеров (например, размеры кнопок, предлагаемые Delphi по умолчанию, лучше не менять). Если используются кнопки разных идеологий (обычные, быстрые и др.), то же-

лательно объединять их каким-либо признаком, например, высотой, цветом и т.п. Визуальный порядок воспринимается более спокойно и естественно, поэтому элементы, которые возможно сделать одинаковой высоты, следует такими и делать.

Закон Фиттса

Этот закон гласит, что время, необходимое для достижения цели, является функцией расстояния до объекта и его размера. Хотя на первый взгляд закон кажется логичным, он чаще всего игнорируется. Хотя, согласно этому закону, самые быстро достижимые цели на любом экране - это его четыре угла, используется это свойство далеко не всегда.

Используйте края, верх, низ и углы экрана. Панель инструментов с кнопками в один ряд, расположенная у самого края экрана, будет во много раз быстрее, чем два ряда иконок тоже у края экрана, но с обводкой шириной в один пиксель, не реагирующей на щелчки мышью.

Используйте большие объекты для важных функций - большие кнопки быстрее, так как их легче находить и по ним легче попасть мышкой [2].

Перегруженность элементами управления

Перегруженность окон программы элементами управления встречается довольно часто. Когда разработчики программ просто не знают, как сделать по-другому, они располагают всю необходимую в данный момент, по их мнению, информацию на одном окне. Окно при этом превращается в мешанину элементов управления и надписей. Даже опытный пользователь будет работать с такой программой с трудом, не говоря уже о новичке.

Программы предназначены для того, чтобы помогать пользователю делать свою работу. Пользователь не должен тратить свое время на саму программу. Перегруженность элементами управления приводит к тому, что человек отвлекается от своей основной задачи и принимается рассматривать окно программы, пытаясь понять, что здесь где и как с этим работать. Большое количество посторонних элементов заслоняет необходимую информацию. В ре-

зультате производительность работы с программой резко снижается. Кроме того, пользователю в этом случае практически невозможно создать у себя в голове модель поведения этого окна, и позднее ее использовать. Каждый раз при работе с этим окном процесс познания начинается сначала.

Необходимо всегда стремиться к минимизации числа элементов управления на экране: чем их меньше, тем лучше интерфейс. Существуют так называемые программы-агенты, выполняемые в фоновом режиме. Они собирают и накапливают информацию, на основе которой выполняют определенные действия. Примером может быть программа-агент, собирающая информацию о пользовательских привычках и предпочтениях при работе с компьютером, которая в соответствии с этим сама настраивает рабочую среду, выдает напоминания и т.д. [12].

2.4.3. Расположение информации на экране

Плотность расположения данных зависит от конкретного пользователя и задачи. Например, данные на экране, предназначенные для работы опытного пользователя, располагают плотнее, чем данные, предназначенные для работы неопытного. Однако существуют некоторые правила, регулирующие плотность расположения данных на экране с помощью интервалов:

- оставлять пустым приблизительно половину экрана;
- оставлять пустую строку после каждой пятой строки таблицы;
- оставлять 4 или 5 пробелов между столбцами таблицы.

Эти правила определяют лишь общие принципы, иногда они могут быть нарушены, но делать это нужно вполне сознательно [5].

На экране должна находиться только та информация, которая действительно необходима пользователю на данном этапе работы. Тот факт, что доступна или запомнена вместе с необходимой и другая информация или что существует место для размещения большого объема информации на экране, не имеет значения. Требования пользователя являются решающим фактором, по-

этому разработчик должен так продумать процесс доступа к нужной информации, чтобы помочь пользователю выполнить его задачу.

В меню следует помещать список только тех пунктов, из которых пользователь может сделать выбор, независимо от того, традиционное это меню или строка команд, описанная для электронной таблицы. Меню смешанных диалогов также выводят полностью все команды меню независимо от того, доступна ли какая-нибудь команда в данной точке программы или нет. В обоих случаях варианты, недоступные пользователю, нужно выделить на экране (например, другим цветом).

Важно, чтобы на экране отображалась вся информация, относящаяся к решаемой на данном этапе задаче. Не следует заставлять пользователя запоминать информацию на одном экране, чтобы позднее воспользоваться ею для обработки информации на следующем экране. Если вся информация исходного документа не помещается на одном экране, некоторые элементы данных могут повторяться на других экранах для сохранения последовательности обработки. Если данные выводятся на несколько экранов, то делить их нужно так, чтобы сохранялась четкая граница раздела (если таковая имеется). Нельзя нарушать логические связи в группе данных; логически связанные данные должны представляться на экране отдельной группой. Этого можно достичь, оставляя по несколько пробелов с каждой стороны группы либо проводя горизонтальные или вертикальные линии, либо используя различные цвета для полей разных групп.

Один и тот же тип информации должен проявляться всегда в одном и том же месте экрана. Важно учитывать и эстетические характеристики. Легче следить за данными, красиво расположенными на экране, при этом повышается безошибочность работы.

Нужно также располагать на одном экране все данные, необходимые для принятия конкретного решения. Фрагменты текста должны располагаться на экране так, чтобы взгляд пользователя сам перемещался по экрану в нужном направлении. Количество блоков должно быть достаточным. Содержимое по-

лей не должно «прижиматься» к краю экрана, а располагаться около горизонтальных или вертикальных осей. Меню, содержащее относительно небольшой объем информации, должно смещаться в левую верхнюю часть экрана. Для подчеркивания симметрии содержимое и наименования полей, относящиеся к одной группе, должны выравниваться по вертикали. По возможности необходимо выравнивать все логически связанные группы данных.

Важно правильно определить точку начала отсчета. Естественнее всего начать из левого верхнего угла и перемещаться слева направо и сверху вниз. Хотя расположение материала на экране может задавать другую последовательность перемещения, общепринятую последовательность следует менять только обоснованно.

Если расположение данных нужно согласовывать как в пределах одной программы, так и между несколькими программами, важно располагать их на экране соответственно одному и тому же шаблону.

Вспомогательные сообщения требуют тщательной подготовки, так как необходимо, чтобы они полностью соответствовали возникшей проблеме.

В пунктах меню и надписях на кнопках ключевое слово должно быть первым.

Пример

Вымышленный текстовый редактор:

Первый пример в действительности более информативен и точен: никто не "вставляет" колонтитул, если он будет помещен после всех других колонтитулов. И никто не "вставляет" содержание, если оно уже есть. Вместо этого оно обновляется.

Тем не менее, второй пример более эффективен по времени. Почему? Потому что дополнительная информация в первом примере не перевешивает преимущества просматривать взглядом только одно слово в каждом пункте меню, чтобы найти нужный.

неправильно

Вставить
Разрыв страницы
Добавить колонтитул
Обновить содержание

правильно

Вставить
Разрыв страницы
Колонтитул
Содержание

2.5. ОКНА ДИАЛОГОВ

2.5.1. Использование различных видов диалогов

Существуют четыре традиционные структуры диалога, каждая из которых основывается на аналогии с определенным типом взаимодействия между людьми и в той или иной степени отвечает перечисленным основным критериям. Эти структуры обеспечивают различный уровень поддержки пользователя [5].

Меню. Предполагается, что меню следует использовать в тех случаях, когда пользователь неопытен или пользуется преимущественно методом ввода путем указания нужного объекта из ограниченного множества данных, например, при выборе задачи для исполнения.

Язык команд. Предполагается, что эта структура будет использоваться подготовленным пользователем там, где задачи обработки не имеют иерархической структуры и не требуется много данных на входе.

Формы (шаблоны). Удобны там, где можно заранее определить стандартную последовательность вводимых данных, например, при обработке транзакций в виде таблиц. Ее целесообразно использовать для ввода табулированных данных.

Структура диалога типа «вопрос-ответ». Представляет собой разумный компромисс для различных уровней подготовки пользователей. Ее можно использовать вместо любой из приведенных выше структур, но особенно хорошо она подходит там, где диапазон входных величин слишком велик для структуры типа меню или слишком сложен для структуры на основе языков команд, либо когда последующий вопрос зависит от ответа на текущий вопрос.

Несмотря на то, что большинство систем имеют в своей основе структуры диалога типа «вопрос-ответ», меню или структуры на базе команд, редко удастся построить диалог для всей системы, используя только одну структуру. Для разных частей диалога нужны различные структуры в зависимости от их конкретных характеристик. Другими словами, большинство диалогов должны

базироваться на смешанных структурах, объединяющих в себе несколько основных структур.

2.5.2. Окна диалогов с закладками

Диалоги с закладками группируют несколько связанных функций вместе, теоретически делая работу пользователя более продуктивной с помощью более быстрого доступа к функциональности. В то же время, разработчик решает еще более важную проблему - сложность и большое число подразделов на верхнем уровне меню, перенося некоторую функциональность в диалог с закладками.

Закладки - это украшение. Разработчикам они нравятся потому, что они могут организовать и разместить сложный набор операций четко и ясно. Пользователям они на первый взгляд нравятся по тем же причинам, но, работая с программой, пользователь думает о своей задаче, а не об интерфейсе. Сложная семантика, которая делает возможным реализацию диалога, становится опасной, когда внимание пользователя не направлено на нее.

Окна диалогов с закладками создаются исходя из следующих предположений:

- 1) пользователь обычно пользуется функциями, сгруппированными в диалоге вместе (если нет, тогда, возможно, нет никакого преимущества в их группировке);
- 2) порядок использования трудно предположить (если это сделать можно, тогда последовательность экранов будет лучше).

Все это хорошо в теории, но на практике часто получается, что старые проблемы заменяются новыми. В большинстве случаев излишнее стремление к практичности может привести к тому, что пользователи теряются и не могут понять, когда изменения в диалоге вступят в силу. Как правило, путаница происходит, когда пользователь меняет что-то более чем на одной закладке, перед тем как нажать **Ок** или **Отмена** [13].

Табулированные диалоги могут затруднять понимание того, когда именно наступит эффект от сделанных в них изменений. Большинство многостраничных диалогов следуют правилу, применяемому в стандартных диалогах: изменения вступают в действие по нажатию пользователем кнопки ОК, но иногда этому правилу следуют от случая к случаю. Например, в диалоге «Параметры» из Microsoft Word 6.0 некоторые опции специально запрограммированы вступать в действие, когда пользователь переключится на другую страницу. Например, в группе опций «Show» (Показывать) все опции за исключением опции «Picture Placeholders» (пустые рамки рисунков) начинают действовать, когда пользователь переключится на другую страницу. В дополнение, когда пользователь после изменения одной из этих опций перейдет на другую страницу, кнопка «Cancel» (Отмена) превратится в «Close» (Заккрыть)*. Это означает, что указанные изменения не могут быть отменены, разве что вручную установить их оригинальные значения. Прочие опции на этой закладке и на других закладках диалога подвержены такому непоследовательному применению правил. Непоследовательность такого рода не дает возможность пользователю предугадать поведение приложения, затрудняет обучение работе с этим и с другими приложениями и оставляет негативный опыт общения с компьютером [14].

Когда пользователи взаимодействуют с программой, они формируют для себя некую ментальную модель ее работы. Эта модель увеличивается и совершенствуется в результате наблюдения за своими действиями и реакцией на них программы. У неопытного пользователя часто создается нечеткая ментальная модель поведения программы. Так, когда изменения происходят «в темноте» (без объяснения), пользователю трудно понять, как и почему они произошли. Когда пользователи делают ошибки в окнах диалога с закладками, они часто не понимают, как и почему это случилось. Это приводит к двум последствиям: либо пользователь «достраивает» недостающие правила сам, либо переста-

* В Office 97 Pro исправлено.

ет работать с этими функциями, потому что ментальная модель показывает, что они ненадежны.

Windows 95 располагает кнопки Ok и Отмена в визуальной другой области, чтобы показать, что кнопки воздействуют на все закладки. Однако риск существует и в этом случае - различие довольно слабое.

Пользователи ожидают, что изменения вступят в силу, когда они переключаются с одной закладки на другую (хотя на самом деле они срабатывают после нажатия на Ok). Например, пользователи электронной таблицы (привыкшие работать с Excel.4) открывают диалоговое окно «Формат ячейки» в Excel.5 и меняют настройки на разных страницах. Затем они переходят на другую закладку, на которой нет нужных им настроек, нажимают на кнопку Отмена и с удивлением видят, что ничего не изменилось.

Частью проблемы является то, что нажатие Ok сохраняет изменения, которые пользователи больше не видят. Одно из решений состоит в том, чтобы спрашивать пользователя, сохранять изменения или нет всякий раз, когда они переходят с закладки на закладку, но этот способ неприемлем, если пользователи часто осуществляют переходы. Другая идея - напоминать пользователю, что изменения не работают, пока он не нажал Ok.

Кроме того, закладки являются отвлекающими сами по себе, и может возникнуть проблема времени, которое отнимается у пользователя от его основной задачи. Пользователи должны думать о своей работе, а не о манипулировании интерфейсом. Как только они попадают на окно с закладками, они на два уровня отдаляются от своей работы - один уровень - это сами настройки, а другой - закладки. Преодолевая семантику закладок, пользователь может уйти далеко от своей работы.

Окна диалогов с закладками довольно эффективны для сложных и не используемых часто операций. Однако использование их для простых и частых операций ведет к раздражению и потере производительности [13].

2.6. ВЫБОР ТЕРМИНОЛОГИИ

Программисты при разработке программ часто забывают, что пользоваться ими будут обычные люди, которые не знакомы со спецификой программирования. Терминология, используемая во многих приложениях, часто порождает у пользователя чувство, будто интерфейс написан на иностранном языке. Большинство людей в действительности толком не представляют себе, что такое, например, «база данных» или понятие «записи». Файлы и манипуляции с ними тоже сложны для пользователей.

Такая терминология пугает пользователей, в результате чего снижается эффективность их работы. Практически всегда в подобных случаях можно назвать вещи более понятными именами. Программа должна говорить с пользователями на их языке. Применение жаргона допустимо, но только при условии, что он используется в среде пользователей, а не специалистов в области вычислительной техники. Разработчик может назвать задачу «коррекцией существующего файла», но если пользователь называет ее «обработкой формы 3», то именно так и надо назвать решение этой задачи, вызываемой диалоговым процессом [5, 12].

Умолчания

Не стоит использовать слова *«по умолчанию»*. Лучше заменить их на: *«Стандартный»*, *«Использовать Принятые Установки»*, *«Восстановить начальные установки»* или другие более специфичные термины, описывающие то, что действительно произойдет [2].

Установки «по умолчанию» должны быть легко сбрасываемыми. Поля, содержащие такие значения, рекомендуется выделять, чтобы пользователь мог быстро и просто заменить их на новые. Установки «по умолчанию» должны быть «интеллектуальными».

3. ЭФФЕКТИВНОСТЬ РАБОТЫ ПОЛЬЗОВАТЕЛЯ

При разработке интерфейса необходимо стремиться к производительности пользователя, а не компьютера. Человеческое время стоит гораздо больше, чем машинное, и хотя может показаться, что увеличение производительности машины должно привести к увеличению производительности человека, часто происходит наоборот. Поясним это на несколько отвлеченном, но выразительном примере: что займет меньше времени - нагрев воды в микроволновой печи в течение 1 минуты и 10 секунд или в течение 1 минуты и 11 секунд? С точки зрения печи, 1 минута и 10 секунд - единственно верный ответ. Однако с точки зрения человека 1 минута и 11 секунд быстрее. В первом случае пользователь должен нажать кнопку «1» дважды, затем найти взглядом кнопку «0», переместить палец на нее и нажать ее один раз. Во втором случае пользователь просто нажмет одну и ту же кнопку - кнопку «1» - три раза подряд. Обычно для нахождения кнопки «0» требуется более чем одна секунда. Таким образом, в первом случае вода нагревается быстрее, но «готовится» дольше, что и делает решение «111» более эффективным. Для поиска другой кнопки требуется не только время, но и довольно высокий уровень познавательной работы мозга. Пока эта работа происходит, главная задача пользователя - приготовить еду - остается в стороне. Чем дольше она остается в стороне, тем больше времени потребуется, чтобы к ней вернуться.

3.1. УМЕНЬШЕНИЕ ЗАДЕРЖКИ

Желательно, чтобы время отклика было сопоставимо с реакцией человека. Люди теряют терпение всякий раз, когда вынуждены ждать отклика от системы. Задержку часто можно «спрятать» от пользователя через многозадачность, позволяя им продолжать выполнять другую работу, пока производятся фоновые вычисления. В любом случае пользователь должен иметь подтверждение того, что программа не «зависла», а продолжает работать.

Способы уменьшения видимости задержки:

- 1) обеспечение сопровождения щелчков на всех кнопках визуальной или звуковой обратной связью в пределах 50 миллисекунд;
- 2) изменение формы курсора на песочные часы везде, где действие занимает от 1/2 до 2 с;
- 3) анимированный курсор, не позволяющий пользователям думать, что система «умерла»;
- 4) вывод сообщений, показывающих потенциальную длительность ожидания результата, для любого действия более чем 2 секунды;
- 5) демонстрация действительной длительности с помощью анимированного индикатора;
- 6) показ занимающих пользователя сообщений, чтобы он был проинформирован и занят во время ожидания завершения долгих процессов;
- 7) привлечение внимания пользователя после завершения длительных (более 10 секунд) задержек звуковым сигналом или заметной визуальной индикацией;
- 8) необходимо отлавливать повторные щелчки на одной и той же кнопке или объекте, во время длительного ожидания люди иногда нажимают на одну и ту же кнопку дважды, замедляя процесс еще больше [2].

3.2. УДОБОЧИТАЕМОСТЬ

Текст для чтения должен иметь высокий контраст. Основное требование к сочетанию цветов для текста и фона – достаточный контраст между ними, необходимый для комфортного, не утомительного чтения. Контраст этот должен выражаться в различной яркости цветов, так как разница только в тоне или насыщенности не позволит сознанию различать текст и фон с достаточным автоматизмом, а для текста с небольшим кеглем его тональная окраска или степень насыщенности вообще с трудом различимы.

При размещении текста на цветном фоне недопустима ситуация, когда буквы и фон имеют оттенки одинаковой или похожей яркости. Проверить, насколько удачно подобраны оттенки по яркости, довольно просто: надо посмотреть, как будет выглядеть изображение в черно-белом варианте. Если текст хорошо различим, то все в порядке, если же он сливается с фоном, это предупредительный сигнал того, что над видом текста надо поработать еще.

В возрасте 45 лет практически у всех людей зрачок становится менее гибким и пропускает меньше света в глаз. Поэтому решения относительно размера и контраста шрифта надо принимать с учетом этой человеческой особенности.

Рекомендуют избегать размещения текста на сером фоне. Раньше считалось, что сочетание черного текста на белом фоне воспринимается лучше всего, однако современные исследования показали, что оно стоит только на четвертом месте по удобочитаемости. По степени ухудшения восприятия цветовые сочетания располагаются следующим образом:

1) синий на белом; 2) черный на желтом; 3) зеленый на белом; 4) черный на белом; 5) зеленый на красном; 6) красный на желтом; 7) красный на белом; 8) оранжевый на черном; 9) черный на пурпурном; 10) оранжевый на белом; 11) красный на зеленом [15,16].

4. ЭЛЕМЕНТЫ ДИЗАЙНА

4.1. ПРИМЕНЕНИЕ ШРИФТОВ И ЭФФЕКТОВ

Большое разнообразие шрифтов и различных эффектов оформления рассеивает внимание, заставляя взгляд метаться по экрану. При этом легко пропустить важную информацию. Чувствуя это, человек прилагает дополнительные усилия для концентрации внимания, что приводит к быстрой утомляемости [17,18].

Шрифты

Каждый шрифт оказывает влияние на общий характер текстового материала.

Текстовые шрифты

С засечками. Шрифты с засечками хорошо смотрятся при малом размере символов и лучше всего подходят для основного текста. Это обусловлено тем, что засечки помогают рассмотреть форму символов, различать буквы с засечками легче. Такие шрифты используются в документах официального характера.

Рубленые. Простота и элегантность шрифтов без засечек делает их наиболее подходящим средством для различных текстовых выделений и заголовков.

Пример

Шрифт с засечками – удобен для чтения
Рубленый шрифт – хорош для заголовков

Декоративные шрифты

Стилизованные под рукописные. Использование рукописных шрифтов требует особой осторожности, так как они затрудняют чтение. Следует учитывать, что такие шрифты обладают сильным ассоциативным воздействием, создавая впечатление небрежности. Поэтому нецелесообразно их применение при выведении информации технического характера, текстов большого объема и особенно в тех случаях, когда необходимо быстрое прочтение.

Архаические. Например, готические или старославянские. Эти шрифты, как и «рукописные», оказывают весьма сильное ассоциативное воздействие, которое существенно различается в зависимости от возраста, профессиональной деятельности и прочих личностных факторов. Кроме того, применение подобных шрифтов требует соблюдения особых орфографических правил. В

наше время текст, написанный подобным образом, способны воспринимать лишь немногие люди, особенно при быстром чтении.

Начертания шрифтов

Каждый шрифт имеет несколько начертаний – светлое, курсивное, полужирное и полужирный курсив. Это позволяет при использовании двух – трех шрифтов иметь достаточный выбор средств для выделения текстовых элементов.

Полужирное начертание. Придает тексту большую весомость и выразительность, поэтому хорошо подходит для заголовков и важных сообщений. Применять его следует очень умеренно, поскольку его избыток делает текст слишком плотным, темным и малопривлекательным.

Курсив. Используется для выделения слов в тексте или сообщения им шутиwego или иронического содержания. Курсив придает тексту разговорный оттенок.

Существуют дополнительные начертания – контурное, подчеркнутое и с тенями. Они затрудняют восприятие текста, поэтому использование этих начертаний должно быть ограничено. Подчеркивание снижает читаемость текста, так как закрывает нижние элементы символов.

Пример

Фразы, набранные контурным шрифтом,
или шрифтом с тенью, а также
подчеркнутые фразы читаются с трудом

Эффекты

Благодаря компьютерам появилось множество способов преобразования шрифтовых элементов в графические. Несмотря на заманчивые перспективы творческого обращения со шрифтами, настоящий успех возможен только при четком представлении того, для чего производится то или иное действие. Важ-

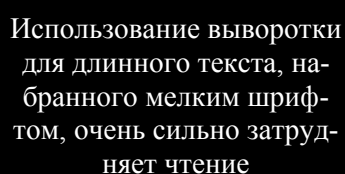
нейшим критерием оценки является читаемость текста, если его прочтение станет затруднительным, заключенный в нем смысл будет теряться. Различные эффекты могут использоваться для выделения важной информации, привлечения внимания пользователя к ней. Однако применение большого количества различных эффектов является дополнительной нагрузкой на психику пользователя и влечет за собой быструю утомляемость.

Светлый текст на темном фоне (Выворотка). Очень хорошо подходит для привлечения внимания, но сильно утомляет глаза. Большие тексты, набранные вывороткой, вызывают эффект психологической усталости, при котором сильный раздражитель приводит к отказу от восприятия всего текста. Поэтому использовать выворотку следует в небольших количествах, а так же следить за тем, что бы шрифт был достаточно крупным. Нецелесообразно оформлять таким образом самую информативную или трудную для восприятия часть текста. Если текст длинный и набран мелким шрифтом, выворотка только затрудняет чтение [9,17].

Пример



ВЫВОРОТКА



Использование выворотки
для длинного текста, на-
бранного мелким шриф-
том, очень сильно затруд-
няет чтение

Объемный текст. Может существенно ухудшить читаемость текста, так как изменяет форму букв, что затрудняет их восприятие. Однако как способ привлечения внимания этот прием подходит великолепно.

Мерцание. Самый сильный способ привлечения внимания, как, впрочем, движение вообще. Именно поэтому применять его следует с особой осторожностью.

Существует еще множество различных способов обработки шрифтов, однако рассмотрение их не является нашей целью. Основное правило использования различных эффектов заключается в том, что их применение должно

быть оправдано насущной необходимостью. Удобство восприятия текста гораздо важнее дизайнерских изысков.

Выравнивание текста

Способ выравнивания влияет не только на величину пространства, занятого текстом, но и на его читаемость [18].

Выравнивание по левому краю. Исследования читаемости текстов показывают, что большинство людей предпочитают выравнивание по левому краю, то есть когда первые буквы всех строк находятся на одной линии по вертикали, а сами строки при этом имеют разную длину. Такое оформление придает тексту неофициальный, современный и «открытый» характер. Читать такой текст очень удобно, так как одинаковые интервалы между словами облегчают выделение групп слов при чтении, а малое количество переносов и одинаковое расположение начала каждой строки автоматизируют процесс чтения и облегчают восприятие смысла текста.

Выравнивание по формату. Производится как по левому, так и по правому краю, и все строки имеют одинаковую длину. Принято считать, что такой текст читается хуже из-за большого количества переносов и образующихся местами больших пробелов между словами. Тем не менее, такой метод позволяет повысить плотность размещения текста, в результате для передачи того же объема информации требуется меньше места.

Выравнивание по центру. Центрированный текст носит официальный характер, поэтому его часто используют в деловых бумагах. Центрирование больших блоков текста не желательно, так как, читая такой текст, приходится каждый раз отыскивать взглядом начало новой строки.

Выравнивание по правому краю. Применение такого способа расположения текста влечет за собой те же неудобства, что и центрирование, то есть вынуждает читающего при переходе к очередной строке каждый раз оста-

навливаться и искать ее начало. Поэтому такое выравнивание лучше использовать только в заголовках.

4.2. ЦВЕТОВОЕ ОФОРМЛЕНИЕ

Пользователям нравится цветное изображение. Люди могут различать тысячи оттенков цвета, но работать могут лишь с ограниченным числом одновременно. На разных экранах один и тот же цвет может иметь разные оттенки. Некоторые комбинации цвета, например голубой цвет символов на красном фоне, неприятны для глаз. Лучший способ узнать, как будет выглядеть цвет, - это посмотреть его на экране.

Воздействие цвета на человека во многом определяется личностными характеристиками воспринимающего субъекта. Цвет, с одной стороны, является одним из простейших средств привлечения внимания, а с другой, очень сильным раздражителем, способным затруднить восприятие. Таким образом, кроме самостоятельного значения цвета, при его использовании необходимо учитывать следующие факторы:

- характер создаваемого программного продукта;
- психологические характеристики группы пользователей, зависящие от ее социального и профессионального состава;
- средства и технологии передачи цвета [9].

Работать с цветом вообще нужно аккуратно. Важно учитывать, что люди связывают с различными цветами особые представления. Цвета, используемые в системе, должны соответствовать этим представлениям. Правильно примененный цвет может, например, передавать тонкие различия между однородными элементами. Неправильное применение цвета будет мешать работать с программой.

Цветные элементы могут быть использованы в качестве визуальных направляющих. Повторяющиеся разделы, элементы управления, отвечающие за

сходные функции, часто выделяются определенным цветом, для того что бы облегчить поиск нужной информации.

Неправильное использование цветов может отталкивать от приложения, не только придавая ему непрофессиональный вид, но и отвлекая пользователя от работы с программой. Хорошим образцом верного выбора цветового решения может послужить панель инструментов в Microsoft Word, использующая ограниченный набор цветов приглушенных тонов. Несмотря на это, панель Word'a дает большую информацию на небольшом пространстве, потому что для различия между функциями опирается в первую очередь на форму. Разумный выбор цветов оставляет меньшую вероятность для непредусмотренной интерпретации из-за личных или национально-культурных цветовых ассоциаций пользователя [14].

Особого внимания требует использование красного цвета. Например, использование в системе красного для вывода сообщений о подтверждении любых действий пользователя и зеленого цвета для вывода сообщений об ошибках может запутать пользователя. Это связано с тем, что у большинства людей красный цвет ассоциируется с опасностью (исключением является использование красного цвета, определяемое соглашениями, принятыми в конкретной предметной области, – так, в электроэнергетике красным цветом обозначаются линии электропередачи мощностью 500 кВ).

Как правило, большое количество красного цвета в каком-либо месте экрана привлекает внимание, настораживает, заставляет пользователя думать, что что-то не так. Применение красного цвета требует особой осторожности, им не следует злоупотреблять, чтобы не ввести пользователя в «стрессовое» состояние [12].

Рекомендуются следующие правила использования цвета:

- 1) используйте минимальное количество цветов, не более трех или четырех на одном экране;
- 2) для больших прямоугольников используйте цвет фона;

- 3) используйте яркие цвета для выделения данных, а более спокойные тона для фона;
- 4) для выделения двух областей используйте для одной черный цвет или цвет из одного из концов спектра, а для другой области возьмите белый цвет или цвет из середины спектра;
- 5) используйте цвет в соответствии с представлениями пользователя о нем;
- 6) поэкспериментируйте с различными цветовыми оттенками на реальном экране [5].

5. ЗАКЛЮЧЕНИЕ

В предлагаемом препринте поставлена задача попытаться собрать воедино и систематически изложить разрозненные рекомендации по проектированию пользовательских интерфейсов с учетом психологических аспектов. Большая часть публикаций при этом взяты из Internet.

Психологические аспекты проектирования пользовательских интерфейсов по содержанию пересекаются с вопросами восприятия, дизайна и организации рекламы, но цели, которые при этом ставятся, отличаются от рекламных целей. Если в рекламе главное – привлечь и удержать внимание, то при проектировании интерфейсов главная задача - облегчить процесс восприятия и переработки информации.

Масштабы распространения компьютеров, все возрастающая интенсификация человеческого труда требуют повышения внимания к проектированию интерфейсов, с тем, чтобы по возможности способствовать устранению или уменьшению стресса, который испытывает человек, особенно неподготовленный, при работе с компьютером.

Как правило, в специальной программистской литературе уделяется недостаточно внимания учету психологических факторов. Данный препринт предназначен для того, чтобы частично восполнить этот пробел.

6. ЛИТЕРАТУРА

1. Филинов Е. Выбор и разработка концептуальной среды открытых систем // Открытые системы. - 1995. - №6. – С.71-77.
2. Тогназини Б. Основные принципы проектирования интерфейсов // «Центр практических программ»: <http://hci.psychology.ru/toader/articles/>
3. Астапов М. Пользовательский интерфейс программ // «Королевство Delphi» Виртуальный клуб программистов: <http://delphi.vitpc.com/article/interface.htm>
4. Купер А. Остановите сообщения об ошибках! // «Центр практических программ»: <http://hci.psychology.ru/toader/articles/>
5. Коутс Р., Влейминк И. Интерфейс «человек – компьютер»: Пер. с англ. – М.: Мир, 1990. – 501с., ил.
6. Купер А. Миф о метафоре // «Центр практических программ»: <http://hci.psychology.ru/toader/articles/>
7. Донской М. Пользовательский интерфейс // Книжная полка Д. Сатина: <http://www.personal.rtsnet.ru/~dsatin/Library/00006.shtml>
8. Общая психология: Курс лекций для первой ступени педагогического образования. / Сост. Рогов Е.И. – М.: ВЛАДОС, 1995. – 448с.
9. Волкова В.В. Дизайн рекламы: Учебное пособие. – М.: Книжный дом «Университет», 1999. – 144с.
10. Лидин К.Л., Меерович М.Г. Общие принципы композиции в скрин дизайне // «Интерфейсы для всех»: <http://interface.bluebird.ru/stat.htm>
11. Солсо Р.Л. Когнитивная психология. – Пер. с англ. – М.: Тривола, 1996. – 600с., ил.
12. Седельников А. Пять распространенных ошибок при разработке интерфейсов программ // «Центр практических программ»: <http://hci.psychology.ru/toader/articles/>
13. Окна диалогов с закладками // «Центр практических программ»: <http://hci.psychology.ru/toader/articles/> (Статья с сайта User Interface Engineering).
14. Isys Information Architects Inc Зал позора интерфейсов: <http://ssmu.tomsk.ru/~iarc/mshame.htm>
15. Платонова М. Психология цвета // Маша Платонова – картины, баннеры, статьи: http://www.566.org/~masha/psigilogiya_cveta.html
16. Гермогенова Л.Ю. Эффективная реклама в России. Практика и реализация. – М.: РусПартнер ЛТД, 1994. – 205с.
17. Буковецкая О.А. Дизайн текста: шрифт, эффекты, цвет. – М.: ДМК, 1999. – 304с., ил.
18. Паркер Р. Как сделать красиво на бумаге. – Пер. с англ. – СПб: Символ-Плюс, 1999. – 336с., цв. ил.

Галина Геннадьевна Массель

Психологические аспекты пользовательского интерфейса современных компьютерных систем.

Препринт ИСЭМ СО РАН, 2000 - №

Утверждено к печати Институтом систем энергетики им. Л.А. Мелентьева СО РАН.

Редактор Г.Г.Боннер
Подписано к печати под №

Формат 60*84*1/16
Уч.изд.л.
Тираж экз. 70
Заказ №

Лицензия ПЛД № 40-61 от 31.05.99г.
Ризограф ИСЭМ СО РАН
664033, Иркутск, Лермонтова, 130