

**Федеральное Государственное бюджетное образовательное учреждение  
высшего профессионального образования «Алтайский государственный  
технический  
университет им. И.И. Ползунова»**

**Андреева А.Ю.**

**Проектирование человеко-машинного интерфейса**  
**УЧЕБНОЕ ПОСОБИЕ**

**Барнаул • 2013**

Андреева А.Ю. Проектирование человеко-машинного интерфейса: учебное пособие для направления направлению 231000 «Проектирование человеко-машинного интерфейса». – АлтЮ. гос. техн. ун-т им. И.И. Ползунова. – Барнаул: АлтГТУ, 2013. – 123 с.

Учебное пособие разработано в соответствии с образовательными стандартами по дисциплине «Проектирование человеко-машинного интерфейса». В учебном пособии рассматриваются основы проектирования, разработки и развития человеко-машинных интерфейсов для эффективной работы пользователя. Данная дисциплина описывает технологии разработки и адаптации пользовательского интерфейса под широкий класс задач, а также обеспечение эффективности использования компьютерных систем.

Кроме того будут рассмотрены основные методики анализа интерфейсов, методики повышения полезности разрабатываемых и используемых программных систем, а также тенденции развития современных интерфейсов.

Рассмотрено и рекомендовано к изданию  
на заседании кафедры ПМ  
Протокол № 12 от 12.02.2013

Рецензент кандидат физ-мат. наук, доцент Крайванова В.А.

# 1 Человеко-машинное взаимодействие. Основные понятия

Наиболее часто используемые термины: эргономика и usability. Usability – качественный признак, определяющий насколько интерфейс легок в использовании.

По статистике разработка интерфейса – 50-60% кода программы, 30% денег и 40% усилий разработчика.

Большинство пользователей не разделяют функциональность программы и пользовательский интерфейс. Поскольку с точки зрения пользователя ПИ является ключевым фактором для понимания функциональности программы, плохо разработанный интерфейс резко ограничивает функциональность системы в целом. Компании, которые не стремятся провести разработку эргономичного ПИ для своих продуктов и получить все преимущества, которые обеспечивают современные технологии, ослабляют свои позиции в конкурентной борьбе.

Изменения в относительной стоимости технологий и человеческого труда также заставляют производителей ПО сосредоточить усилия на разработке ПИ. Стоимость используемых в бизнесе технологий неуклонно снижается, в то же время **стоимость времени операторов столь же неуклонно растёт**. Таким образом, инвестиции в технологии, которые позволяют повысить эффективность труда операторов (за счет снижения затрат на обучение, упрощение задач, исправления ошибок и т.п.) оказывают значительное влияние на стоимостной порог эксплуатации системы в целом.

Выделим несколько наиболее существенных преимуществ хорошего пользовательского интерфейса с точки зрения бизнеса:

- Снижение количества человеческих ошибок
- Снижение стоимости поддержки системы
- Снижение стоимости обучения
- Уменьшение потерь продуктивности работников при внедрении системы и более быстрое восстановление утраченной продуктивности
- Улучшение морального состояния персонала
- Уменьшение расходов на редизайн ПИ по требованию пользователей

- Доступность функциональности системы для максимального количества пользователей

## **1. Основные определения**

ЧМВ – наука, изучающая особенности использование компьютерных систем при решении поставленной задачи. Цель исследования: создание интерфейса, учитывающего особенности деятельности, поведения, ментальной специфики человека.

ЧМВ включает дисциплины:

- Психология, социология
- Эргономика
- Дизайн
- Информатика и разработка ПО
- Искусственный интеллект

Интерфейс – полный спектр взаимодействия между человеком и компьютером, позволяющий достигнуть поставленной цели и включает все, что помогает пользователю взаимодействовать с компьютером:

- набор решаемых задач
- используемая метафора
- элементы управления системы
- навигация
- дизайн
- документация
- обучение работе с системой

Интерфейсы должны ваять специальные инженеры-проектировщики (GUI Designer) во взаимодействии с инженерными психологами (Usability engineer). Это за границей так. У нас ещё первобытные технологии и интерфейсы лепят все, кому ни лень (программисты, веб-дизайнеры и пр.). Последние, как правило, меньше всего заботятся о пользователе и его

комфортности – с одной стороны, это не их работа, с другой стороны, они этого просто не умеют.

Как такового определения комфортности не существует -- это слишком субъективное (с одной стороны) и качественное (с другой стороны) понятие.

Однако удобство использования того или иного программного продукта можно оценить с помощью специальных методик. В этих методиках оцениваются эргономические свойства ПИ. Проектирование и последующая оценки эргономичности ПИ основывается на разрабатываемых эргономических требованиях (как общего плана, так и частных, под конкретные задачи). Статья Нельсена с определением usability -- там есть пять параметров: <http://www.zdnet.com/devhead/stories/articles/0,4413,2137671,00.html>

(Надо записать урл в адресную строку браузера -- здесь скрипт неправильно линк парсит)

## **2. Основные принципы создания интерфейса:**

Стандарт ISO/DIS 9241-14 говорит о семи необходимых свойствах пользовательского интерфейса:

1. **Соответствие задачам, решаемым пользователем.** Интерфейс соответствует задаче, если пользователю предоставлена возможность выполнения задания без появления дополнительных проблем, не обусловленных характеристиками задачи.
2. **Легкость управления (использования).** Интерфейс легок в использовании, если каждый шаг диалога понятен пользователю, либо объясняется по его запросу. Не должно возникать тупиковых ситуаций, вызывающих замешательство пользователя.
3. **Управляемость.** Организация интерфейса позволяет пользователю в течение всей работы управлять диалогом для достижения поставленной цели. Система не должна увлекаться собственными расчетами, позабыв о пользователе, сценарии работы должны быть подконтрольными.
4. **Соответствие ожиданиям пользователя** Интерфейс должен

соответствовать предыдущему опыту или образованию пользователя. Например, если пользователь хорошо знаком с некоторым интерфейсом, рекомендуется предлагаемый ему новый интерфейс выдержать в том же стиле.

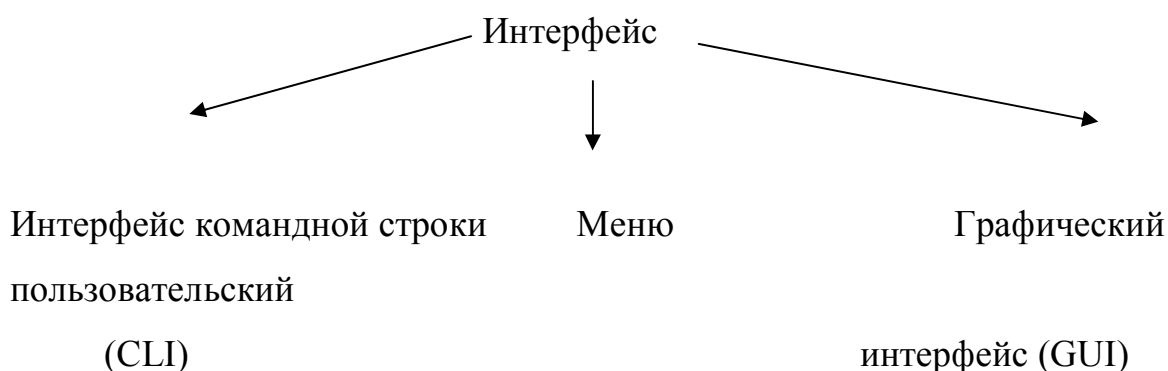
**5. Устойчивость (терпимость) к ошибкам** Интерфейс терпим к ошибкам, если несмотря на очевидные ошибочные действия пользователя требуемый результат может быть достигнут без или с минимальными корректирующими действиями. Не должно возникать фатальных ошибок, или по крайней мере, должно выдаваться развёрнутое сообщение.

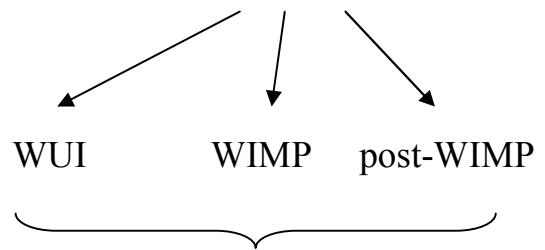
**6. Пригодность к индивидуализации.** Интерфейс индивидуализируемый (или гибкий), если допускает адаптацию к индивидуальным требованиям и профессиональной подготовке пользователя.

**7. Легкость изучения (обучения)** Интерфейс отвечает требованиям легкости обучения, если он обеспечивает последовательное обучение пользователя за минимальное время с минимальными усилиями со стороны пользователя.

- Непротиворечивость: приемы работы в различных блоках системы должны быть идентичны
- Неизбыточность: нельзя требовать от пользователя ввода данных, которые известны или могут быть получены
- Гибкость: возможность для работы как неподготовленных (через меню), так и опытных пользователей (горячие клавиши, параметры).
- Наличие обратной связи

### 3. Классификация интерфейсов





- + наличие интерпретаторов полнофункциональных командных языков
- + возможность создания скриптов
- + возможность удаленной работы
- + полный контроль пользователя над системой

– низкая обучаемость

- + хорошо продуманный графический интерфейс обеспечивает почти такую же функциональность, как командная строка
- + WYSIWYG (What You See Is What You Get)
- + высокая обучаемость, основанная на интуитивном понимании основных объектов – WIMP (Windows Icon Menu Pointer)
- высокие требования к ресурсам
- ограниченная функциональность

## Достоинства CLI

Интерпретаторы полнофункциональных командных языков обходятся десятками, в худшем случае сотнями килобайт оперативной памяти и обеспечивают очень высокую скорость и эффективность использования ресурсов системы.

Хорошая система должна предоставлять *оба* интерфейса. Например, разработчики фирмы Apple долгое время пытались избежать включения в систему командного интерпретатора но в конце концов под давлением пользователей и особенно специалистов по технической поддержке они были вынуждены реализовать командный язык *AppleScript*.

## 2 Элементы управления

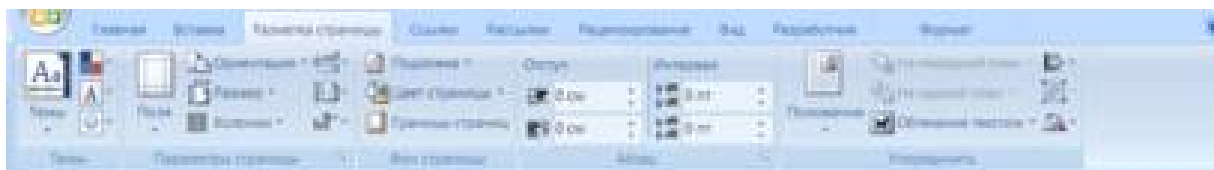
(глава написана по материалам Головач)

### 2.1 Кнопки

#### Командные кнопки

Кнопкой называется элемент управления, всё взаимодействие пользователя с которым ограничивается одним действием – нажатием. Эта формулировка, кажущаяся бесполезной и примитивной, на самом деле очень важна, поскольку переводит в гордое звание кнопок многие элементы управления, которые как кнопки по большей части не воспринимаются (об этом позже). Нажатие на такую кнопку запускает какое-либо явное действие, поэтому правильнее называть такие кнопки **«кнопками прямого действия»**. С другой стороны, из-за тяжеловесности этого словосочетания им всегда пренебрегают. С точки зрения разработчика ПО для настольных систем, командные кнопки являются чрезвычайно простыми и скучными. Иная ситуация в web-интерфейсах, где отсутствие операционной системы (откуда приходят элементы управления) и простота создания новых типов кнопок (это чуть ли не единственный элемент, с которым вообще удастся что-либо сделать) привели к тому, что нестандартные кнопки не создает только ленивый. В то же время, этот самый простой элемент управления имеет больше всего тонкостей

**В web-интерфейсах кнопка должна быть оформлена как текстовая ссылка, если она перемещает пользователя на другой фрагмент контента, и как кнопка – если она запускает действие.**



**Размеры и поля.** Как вы уже знаете, чем больше кнопка, тем легче попасть в нее курсором. Это правило по мере сил всеми соблюдается, во всяком случае, кнопок размером 5 на 5 пикселей уже практически не встретишь. Однако помимо простоты нажатия на кнопку есть другая составляющая проблемы: пользователю должно быть трудно нажать не на ту кнопку. Добиться этого можно либо изменением состояния кнопки при наведении на неё курсором, либо установлением пустого промежутка между кнопками. Первый способ приобрел существенную популярность в web-интерфейсах, второй – в десктопном ПО (он, кстати, более эффективен: одно дело, когда пользователь промахивается мимо кнопки и совсем другое – если, промахнувшись, он ещё и ошибочно нажимает на другую кнопку). Ни тот, ни



другой способы не обеспечивают стопроцентной надежности, так что при прочих равных использовать стоит оба.

**Объем.** Кнопка должна (или не должна) быть пользователем нажата. Соответственно, пользователю нужно как-то сигнализировать, что кнопка нажимаема. Лучшим способом такой индикации является придание кнопке псевдообъема, т. е. визуальной высоты. С другой стороны, этот объем плох тем, что при его использовании возникает рассогласование между обликами кнопок прямого и непрямого действия. Разумеется, никто не отменял ещё и тот факт, что псевдообъем кнопок, вообще говоря, в существенной степени есть визуальный шум. Еще с одной стороны, зачастую возникает необходимость максимально повышать шансы нажатия пользователем какой-либо отдельной кнопки (например, «О компании»), в этих случаях псевдообъем этой кнопки (при прочих плоских) сильно повышает вероятность нажатия. **Направление теней во всех элементах управления должно быть одинаковым: снизу справа**

**Состояния.** Кнопка должна как-то показывать пользователям свои возможные и текущие состояния. Количество состояний довольно велико, при этом наборы возможных состояний в ПО и в web-интерфейсах значительно различаются. Например, кнопка в Windows может иметь шесть состояний: нейтральное, нажатое, нейтральное с установленным фокусом ввода, состояние кнопки по умолчанию, кнопка по умолчанию с установленным фокусом ввода и заблокированное состояние.

В web-интерфейсах обычно используют меньший набор состояний: нейтральное, готовое к нажатию (onMouseOver) и активное (в случаях, когда набор кнопок используется для индикации навигации). Нажатое и заблокированное состояние используются очень редко, а «нейтральное с установленным фокусом ввода» старается, как может, создать браузер. **Никогда не удаляйте элементы, которые нельзя нажать, взамен этого делайте их заблокированными**

**Текст и пиктограммы.** Все руководства по разработке интерфейса с изумительным упорством требуют снабжать командные кнопки названия -ми, выраженными в виде глаголов в форме инфинитива (Прийти, Увидеть, Победить). Разработчики же интерфейса с не менее изумительным упорством не следуют этому правилу. Аргументов у них два: во-первых, все так делают, значит, это есть стандарт и ему нужно следовать, во-вторых, нет времени придумывать название.

Оба аргумента сильны. Действительно, стандарт. Действительно, нет времени. Но есть два контраргумента: во-первых, это не столько стандарт, сколько стандартная ошибка, во-вторых, думать можно и по дороге домой. Если второй контраргумент особых объяснений не требует, то сущность первого полезно объяснить. Кнопка, запускающая действие, недаром

называется командной. С её помощью пользователи отдают системе команды. Команда же в русском языке формируется посредством глагола в повелительном наклонении (никто особо не хочет слишком персонифицировать компьютер и обращаться к нему в наклонении просительном, т. е. Придите, Увидьте, Победите).

Помимо этого, у глагольных кнопок есть одно большое достоинство. По ним понятно, какое действие произойдет после нажатия. Это позволяет как-то разграничить диалоговые окна в сознании (поскольку разные диалоговые окна получают разные кнопки). В результате, из-за увеличения степени уникальности фрагментов системы, обучаться системе получается лучше, нежели с кнопками, одинаковыми везде.

Более того, вкупе со строкой заголовка окна, глагольные кнопки создают контекст, что очень полезно при возвращении к прерванной работе. Оказывается возможным не рассматривать *всё* диалоговое окно, чтобы узнать, на каком действии задача была прервана – достаточно просто прочесть надпись на кнопке. Таким образом, следует всемерно избегать создания кнопок с ничего не говорящим текстом, поскольку такой текст не сообщает пользователям, что именно произойдет после нажатия кнопки. При этом есть одна тонкость.

Существующие интерфейсы заполнены терминационными кнопками **Ок**, **Отмена** (Cancel) и **Применить** (Apply), что, собственно говоря, и позволяет разработчикам ссылаться на стандарт. Эти кнопки плохи. С первой кнопкой понятно – это не глагол, а значит, кнопка плоха. Кнопка одна и та же во всех диалогах, значит всё ещё хуже. У кнопки **ОК**, с другой стороны, есть два достоинства. Во-первых, слишком часто оказывается, что **ОК** является единственным текстом, который можно уместить в кнопке (если во всех отношениях адекватный глагол слишком длинный). Во-вторых, стандартность этой кнопки приводит к почти мгновенному её распознаванию, что позволяет ускорить работу пользователя с системой (с другой стороны, это распознавание может быть неверным, так что появляется риск человеческой ошибки

### Кнопки доступа к меню



Используются когда необходим доступ к небольшому набору команд.


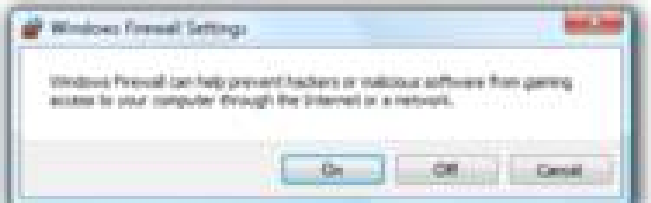
Существуют, впрочем, определенные ситуации, когда такие кнопки очень хороши. Для этого только нужно сделать так, чтобы кнопка была одновременно и командной кнопкой, и показывала меню. Для этого нужно сделать две вещи. Во-первых, нужно разделить кнопку на две области, одна из которых запускает действие, а другая открывает меню. Во-вторых, нужно организовать такой контекст, при котором результат нажатия на кнопку всегда будет понятным. Например, это очень хорошо работает с кнопками **Вперед** и **Назад**. Другой пример: иногда бывают ситуации, когда действий может выполняться несколько, но чаще всего нужно только одно. В этом случае пользователи очень быстро обучаются этому действию, имея доволь -но простой доступ к остальным. В таком исполнении кнопки доступа к меню работают замечательно.

## Чекбоксы и радиокнопки

Первое, что необходимо сказать про чекбоксы и радиокнопки, это то, что они являются **кнопками отложенного действия**, т. е. их нажатие не должно инициировать какое-либо немедленное действие. С их помощью пользователи вводят параметры, которые скажутся после, когда действие будет запущено иными элементами управления. Нарушать это правило опасно, поскольку это серьезно нарушит сложившуюся ментальную модель пользователей. В этом заключается общность чекбоксов и радиокнопок, теперь поговорим о различиях.

- радиокнопки *всегда* должны находиться в рамке группировки, а для чекбоксов это необязательно;
- в группе не может быть меньше двух радиокнопок;
- в группе радиокнопок как минимум одна радиокнопка должна быть проставлена по умолчанию;
- и чекбоксы и радиокнопки желательно расставлять по вертикали, поскольку это значительно ускоряет поиск нужного элемента

**Текст подписей.** Каждая подпись должна однозначно показывать эффект от выбора соответствующего элемента. Поскольку радиокнопки и чекбоксы не вызывают немедленного действия, формулировать подписи к ним лучше всего в форме существительных, хотя возможно использование глаголов (если изменяется не свойство данных, а запускается какое-либо действие). Подписи к стоящим параллельно кнопкам лучше стараться делать примерно одинаковой длины. Все подписи обязаны быть позитивными (т. е. не содержать отрицания). Повторять одни и те же слова, меняя только окончания подписей (например, «Показывать про -белы» и «Показывать табуляции»), в нескольких кнопках нельзя, в таких случаях лучше перенести повторяющееся слово в рамку группировки. Если подпись не помещается в одну строку, выравнивайте индикатор кнопки (кружок или квадрат) по первой строке подписи.

Правильно	Неправильно
	

**Взаимодействие.** Это может показаться невероятным, но до сих пор в web-интерфейсах 99% чекбоксов и радиокнопок реализованы неправильно. Дело в том, что создатели языка HTML, ничего не понимавшие в проектировании интерфейсов, были поначалу искренно уверены в том, что в этих элементах управления нажимается только визуальный индикатор переключения, т. е. кружок или квадратик. На самом деле это совершенно не так! Нажимать должна быть ещё и подпись, просто потому, что закон Фитса («Быстрый или точный») однозначно требует больших кнопок. Но в web-интерфейсах всего этого нет, поскольку в HTML конструкция чекбоксов и радиокнопок просто не позволяла делать нажимаемыми подписи. Сейчас это стало технически возможным (через тег Label), но по инерции и вполне понятной лени никто чекбоксы нормальными не делает.

## Списки

Все часто используемые списки функционально являются вариантами чекбоксов и радиокнопок. Скорость доступа к отдельным элементам и наглядность в них принесены в жертву компактности (они экономят экранное пространство, что актуально, если количество элементов велико) и расширяемости (простота загрузки в списки динамически изменяемых элементов делает их очень удобными при разработке интерфейса, поскольку это позволяет не показывать пользователю заведомо неработающие элементы). Списки бывают пролистываемыми и раскрывающимися, причем пролистываемые могут обеспечивать как единственный (аналогично группе радиокнопок), так и множественный выбор (a la чекбокс); раскрывающиеся же работают исключительно как радиокнопки. Но сначала необходимо рассказать об общих свойствах всех списков.

**Ширина.** Ширина списка как минимум должна быть достаточна для того, чтобы пользователь мог определить различия между элементами. В идеале, конечно, ширина всех

элементов должна быть меньше ширины списка, но иногда это невозможно. В таких случаях не стоит добавлять к списку горизонтальную полосу прокрутки, лучше урезать текст элементов. Для этого нужно определить самые важные фрагменты текста (например, для URL это начало и конец строки), после чего все остальное заменить отточием (...).

Поскольку нужно максимально ускорить работу пользователей, необходимо сортировать элементы. Идеальным вариантом является сортировка по типу элементов. Если же элементы однотипны, их необходимо сортировать по алфавиту, причем списки с большим количеством элементов полезно снабжать дополнительными элементами управления, влияющими на сортировку или способ фильтрации элементов. Если можно определить наиболее популярные значения, их можно сразу расположить в начале списка, но при этом придется вставлять в список разделитель, а в систему – обработчик этого разделителя.

**Пиктограммы.** Уже довольно давно в ПО нет технических проблем с выводом в списках пиктограмм отдельных элементов. Однако практически никто этого не делает. Это плохо, ведь пиктограммы обеспечивают существенное повышение субъективной привлекательности интерфейса и сканируются быстрее «голого» текста.

## Раскрывающиеся списки

Самым простым вариантом списка является раскрывающийся список. Помимо описанных выше родовых достоинств списков, раскрывающиеся списки обладают одним существенным достоинством. Оно заключается в том, что малая высота списка позволяет с большой легкостью визуальнo отображать команды, собираемые из составляющих. Рис. 36. Пример визуальной сборки команды из составляющих. Данный метод значительно проще для понимания, нежели, например, ввод положительного значения для смещения вверх и отрицательного значения для смещения вниз без поддержки раскрывающимся списком. Справа на иллюстрации крутилка (см. стр. 74). Раскрывающийся список, как правило, вызывает две проблемы, одна появляется преимущественно в ПО, другая – в web-интерфейсах. Первая проблема заключается в том, что иногда отсутствие места на экране не позволяет использовать ни чекбоксы с радиокнопками, ни пролистываемые списки множественного выбора. Приходится делать раскрывающийся список, в котором помимо собственно элементов есть «мета#элемент», включающий все элементы из списка. Этому элементу часто не дают названия, оставляя строку списка пустой, что неправильно, поскольку требует от пользователя слишком глубокого абстрагирования. Такой мета#элемент нужно снабжать названием, например, **Все значения** или **Ничего**. В web-интерфейсах проблема иная. Раскрывающийся список часто используется как навигационное меню. Это изначально неправильно, поскольку содержимое такого меню не видно сразу и уж тем более им трудно индентифицировать пользователям, в каком разделе сайта они находятся. Это, впрочем, не главное. Большая проблема заключается в том, что список снабжают скриптом,

который запускается сразу по выбору значения. **Пролистываемые списки**

Другим, более сложным вариантом списка является пролистываемый список. Пролистываемые списки могут позволять пользователям совершать как единственный, так и множественный выбор. Одно требование применимо к обоим типам списков, остальные применимы только к одному типу.

**Размер.** По вертикали в список должно помещаться как минимум четыре строки, а лучше восемь. Напротив, список, по высоте больший, нежели высота входящих в него элементов, и соответственно, содержащий пустое место в конце, смотрится неряшливо. Требование выводить полосы прокрутки в больших списках кажется моветоном, но забывать о нем не следует.

**Списки единственного выбора.** Список единственного выбора является промежуточным вариантом между группой радиокнопок и раскрывающимся списком. Он меньше группы радиокнопок с аналогичным числом элементов, но больше раскрывающегося списка. Соответственно, использовать его стоит только в условиях «ленивой экономии» пространства экрана.

**Комбобоксы** Комбобоксами (combo-box), называются гибриды списка с полем ввода: пользователь может выбрать существующий элемент, либо ввести свой. Комбобоксы бывают двух видов: раскрывающиеся и расширенные. Оба типа имеют проблемы. У раскрывающегося комбобокса есть проблемы. Во-первых, такие комбобоксы выглядят в точности как раскрывающиеся списки, визуально отличаясь от них только наличием индикатора фокуса ввода (да и то, только тогда, когда элемент выделен). Это значит, что полноценно пользоваться ими могут только сравнительно продвинутые пользователи. В этом нет особой проблемы, поскольку комбобоксом все равно можно пользоваться, как обычным списком.

## Поля ввода

**Размеры.** Основная часть требований к полям ввода касается размера. Понятно, что размер по вертикали должен быть производным от размера вводимого текста – если текста много, нужно добавить несколько строк (нарушением этого правила регулярно грешат форумы, заставляющие пользователей вводить сообщения в поля ввода размером с ноготь). С размерами по горизонтали интереснее. Конечно, ширина поля должна соответствовать объему вводимого текста, поскольку гораздо удобнее вводить текст, который *видишь*. Менее очевидным является другое соображение: ширина поля ввода не должна быть больше объема вводимого в поле текста, поскольку частично заполненное поле выглядит как минимум неряшливо.

**Ширина поля ввода не должна быть больше максимальной длины строки**

Отдельной проблемой является ограничение вводимого текста. С одной стороны, ограничение хорошо для базы данных. С другой стороны, всегда найдутся пользователи, для которых поле ввода с ограничением вводимых символов окажется слишком маленьким. Поэтому этот вопрос нужно решать применительно к конкретной ситуации. определить самую большую ошибку, которую разработчики допускают при создании полей ввода. Всякий раз, когда ширина поля ввода больше максимального объема вводимого в него текста, и при этом объем вводимого текста ограничен, пользователи неприятно изумляются, обнаружив, что они не могут ввести текст, хотя место под него на экране имеется. Соответственно, вообще нельзя делать поле ввода шире максимального объема вводимого в них текста.

**Подписи.** Вопрос «где надо размещать подписи к полям ввода?» является одним из самых популярных среди программистов: битвы сторонников разных подходов, хоть и бескровны, но значительны. Аргументов и подходов тут множество, моё личное мнение заключается в том, что, поскольку восприятие подписей занимает определенное время, которого жаль, лучше всего действует следующее простое правило: **в часто используемых экранах подписи должны быть сверху от поля (чтобы их было легче не читать), в редко же используемых подписи должны быть слева (чтобы всегда восприниматься и тем самым сокращать количество ошибок).**

Подписи к полям ввода имеют определенное отличие от других подписей. В полях ввода подписи можно размещать не рядом с элементом, а внутри него, что позволяет экономить пространство экрана. Подпись при этом выводится в самом поле ввода, точно так же, как и текст, который в него нужно вводить. Необходимо только отслеживать фокус ввода, чтобы при установке фокуса в поле убирать подпись. Это решение, будучи нестандартным, плохо работает в ПО, но неплохо работает в web-интерфейсах. Если очень жалко экранное пространство, этим методом стоит пользоваться.

## Крутилки

Крутилка (spinner, little arrow) есть поле ввода, не такое универсальное, как обычное, поскольку не позволяет вводить текстовые данные<sup>1</sup>, но зато обладающее двумя полезными возможностями. Во-первых, чтобы ввести значение в крутилку, пользователю не обязательно бросать мышь и **переносить руку на клавиатуру** (в отличие от обычного поля ввода). Поскольку перенос руки с места на место занимает сравнительно большое время (в среднем почти половину секунды), к тому же ещё и сбивает фокус внимания, отсутствие нужды в клавиатуре оказывается большим благом.

Во всяком случае, ввод значения в крутилку с клавиатуры достаточно редок, т. е. пользователи воспринимают крутилки целиком и полностью положительно. Во-вторых, при вводе значения мышью система может позволить пользователям вводить только **корректные данные**, причем, что особенно ценно, в корректном формате. Это резко уменьшает вероятность человеческой ошибки. Таким образом, использование крутилок для ввода любых численных значений более чем оправдано.

## Ползунки

Как и ранее описанные элементы управления, ползунки позволяют пользователям выбирать значение из списка, не позволяя вводить произвольное значение. Возникает резонный вопрос: зачем нужен ещё один элемент управления, если аналогичных элементов уже полно. Ответ прост: ползунки незаменимы, если пользователям надо дать возможность выбрать значение, стоящее в хорошо ранжирующемся ряду, если: значений в ряду много, нужно передать пользователям ранжируемость значений.

Ползунки имеют интересный аспект. Их можно также использовать для выбора текстовых параметров, но только в случаях, когда эти параметры можно понятным образом отранжировать. Случаев таких немало, например, «завтрак», «обед» и «ужин», при отсутствии внешней связи ранжированию поддаются вполне.

## Меню

В настоящее время систем, которые не использовали бы меню в том или ином виде, практически не осталось. Объясняется это просто. Меню позволяет **снизить нагрузку на мозги пользователей**, поскольку для выбора команды не надо вспоминать, какая именно команда нужна и как именно её нужно использовать — вся (или почти вся) нужная информация уже содержится на экране. Вдобавок, поскольку меню ограничивает диапазон действий пользователей, появляется возможность в значительной мере изъять из этого диапазона **ошибочные действия**.

Более того: меню показывает пользователям объем действий, которые они могут совершить благодаря системе, и тем самым **обучают пользователей** (в одном из исследований было даже обнаружено, что меню является самым эффективным средством обучения<sup>1</sup>). Таким образом, в большинстве систем меню является объективным благом (они неэффективны, в основном, в системах с внешней средой или течением времени).

**Устройство меню.** На эффективность меню наибольшее влияние оказывают **устройство** отдельных элементов и их **группировка**. Несколько менее важны другие факторы, такие как выделение элементов и стандартность меню. Самым важным свойством хорошего элемента меню является его название. Название должно быть самым эффективным из возможного. В



отличие от кнопок в диалоговых окнах, элементы главного меню практически никогда не несут на себе контекста действий пользователя, просто потому, что в любой момент времени доступны все элементы. Это значит, что к наименованию элементов меню нужно подходить весьма тщательно, тщательней, нежели ко всему остальному.

Впрочем, помимо тщательности (и таланта, к слову говоря) нужно ещё кое-что. Обязательно нужно убедиться, что выбранное название понятно целевой аудитории. Сделать это просто – пользователю нужно сообщить название элемента и попросить его сказать, что этот элемент меню делает. Нелишне заметить, что функциональность, не отраженная названием элемента, с большой степенью вероятности не будет найдена значительной частью аудитории. Поэтому не стоит уместать в диалоговое окно какую-либо функцию, если её существование в этом окне невозможно предсказать, глядя на соответствующий элемент меню. **Не делайте элементов меню, часть функциональности которых не влезает в текст элемента**

Особо стоит остановиться на склонении текста. В отличие от диалоговых окон, в которых кнопки прямого и отложенного действия выглядят и действуют по-разному, в меню нет четкой разницы между этими элементами. Единственным способом разграничения этих элементов является текст, так что нужно очень тщательно подходить к тому, чтобы элементы, запускающие действия, были глаголами в форме инфинитива (как команд -ные кнопки).

Впрочем, часто глагол приходится выкидывать вообще, чтобы переместить значимое слово ближе в начало текст элемента. Нужно это, чтобы повысить скорость распознавания. Повысить её можно всего одним способом: главное (т. е. наиболее значимое) слово в элементе должно стоять в элементе первым. Обратите внимание, что короткий текст элемента, без сомнения, быстро читаясь, совершенно необязательно быстро распознается. Поэтому не стоит безудержно сокращать текст элемента: выкидывать нужно все лишнее, но не более.

**Пиктограммы в меню.** Пиктограммы в меню, если они повторяют пиктограммы в панели инструментов, обладают замечательной способностью обучать пользователей возможностям панели. Помимо этого они здорово ускоряют поиск известного элемента и точность его выбора, равно как и общую разборчивость меню. Таким образом, пиктограммы в меню объективно хороши (только стоят дорого, к сожалению). Это очевидный факт. Теперь менее очевидный: пиктограммы лучше работают, когда ими снабжены не все элементы. Когда все элементы имеют пиктограммы, разборчивость каждого отдельного элемента падает: в конце концов, пикто -граммы всех ненужных в данное время элементов являются визуальным шумом. Когда же пиктограммами снабжены только самые важные элементы, их разборчивость повышается (а разборчивость остальных не понижается), при

этом пользователям удастся легче запоминать координаты элементов («элемент сразу под второй пиктограммой»). **Не снабжайте пиктограммами все элементы меню, снабжайте только самые важные**

**Переключаемые элементы.** Особого внимания заслуживают случаи, когда меню переключает какие-либо взаимоисключающие параметры, например, показывать или не показывать палитру. Тут есть несколько возможных способов. Можно поместить перед переключателем галочку, показывая, что он включен (если же элемент снабжен пиктограммой, можно её утапливать). Заранее скажу, что это лучший метод. Можно не помещать галочку, зато инвертировать текст элемента: например, элемент **Показывать сетку** превращается в **Не показывать сетку**. Это плохо по многим причинам. Во-первых, в интерфейсе желательно не употреблять ничего негативного: в меньшей степени потому, что негативность слегка снижает субъективное удовлетворение; в большей степени потому, что она снижает скорость распознавания текста (главное слово не первое, нужно совершить работу, чтобы из отрицания вычислить утверждение). Во-вторых, если изъять «не» и переформулировать одно из состояний элемента, пользователям будет труднее осознать, что два разных элемента на самом деле есть один элемент. Таким образом, галочка предпочтительнее. **Всегда формулируйте текст в интерфейсе без использования отрицаний**

**Предсказуемость действия.** Пользователей нужно снабжать чувством контроля над системой. Применительно к меню это значит, что по виду элемента пользователи должны догадываться, что произойдет после выбора. Сделать это невероятно трудно, поскольку на экране нет места под такие подсказки. Можно сделать только одно, но сделать это нужно обязательно: нужно показать пользователям, какой элемент запускает действие или меняет параметр, а какой открывает окно с продолжением диалога. Почти во всех ОС стандартным индикатором продолжения диалога является многоточие после текста элемента, так что пользоваться этим признаком стоит везде, включая интернет.

Также необходимо показывать, какой элемент срабатывает сразу, а какой открывает элементы меню нижнего уровня (в любой ОС это делается автоматически, в web-интерфейсах нужно не забывать делать это вручную). Это же правило касается и гипертекстовых ссылок вообще (они тоже меню). Пользователи испытывают значительно большее чувство контроля, когда имеют возможность предсказать, куда их ссылка приведет (при этом снижается количество ошибочных переходов). Таким образом, нестандартные ссылки (т. е. ссылки на другой сайт, на почтовый адрес, на файл, на узел FTP, на долго загружающуюся страницу и т.д.) полезно снабжать характерными для них признаками, например, ссылку на почтовый адрес пиктограммой письма.

## **Группировка элементов**

Второй составляющей качества меню является группировка его элементов. В большинстве меню группировка оказывает не меньшее значение при поиске нужного элемента, нежели само название элемента, просто потому, что даже идеальное название не сработает, если элемент просто нельзя найти. Чтобы уметь эффективно группировать элементы в меню, нужно знать ответы на три вопроса: зачем элементы в меню нужно группировать, как группировать элементы и как разделять группы между собой.

**Зачем элементы в меню нужно группировать.** Меню, группы элементов в котором разделены, сканируется значительно быстрее обычного, поскольку в таком меню больше «точек привязки» (так же, как и в меню с пиктограммами). К тому же наличие явных разделителей многократно облегчает построение ментальной модели, поскольку не приходится гадать, как связаны между собой элементы. Наконец, в объемных меню группировка элементов облегчает создание кластеров в кратковременной памяти, благодаря чему всё меню удастся пометить в КВП.

**Как группировать элементы.** Каждый знает, или, во всяком случае, догадывается, что элементы в меню нужно группировать максимально логично. Пospорить с этим утверждением нельзя, но от этого его проблематичность не уменьшается.

**Взаимоисключающие элементы желательно помещать в отдельный уровень иерархии**

Дело в том, что существует множество типов логики. Есть логика разработчика, который знает все функции системы. Есть логика пользователя, который знает только меньшую часть. При этом практика показывает, что эти типы логики в значительной мере не совпадают. Поскольку польза -ватели важнее, нужно сгруппировать меню в соответствии с их логикой.

Для этого используется очень простой и надежный метод, называемый карточной сортировкой.

### **Карточная сортировка**

Карточная сортировка это достаточно новый способ формирования информационной архитектуры проекта. Данный метод техника user-centred design для увеличения параметра "нахождаемости" системы. Процесс заключается в сортировании наборов карточек, который маркируются в соответствии с частью контента или функционала. Фактически на выходе получается ментальная модель пользователя, повествующая проектировщикам о том, как пользователи будущего продукта видят его структуру.

**Как разделять группы между собой.** Существует два основных способа разделять группы: между группами можно помещать пустой элемент (разделитель) или же размещать отдельные группы в разных уровнях иерархии. Второй способ создает более четкое разделение: в меню **Файл**, например все элементы более близки друг другу (несмотря на разделители), чем элементы других меню. В то же время выбор конкретного способа

диктуется результатами карточной сортировки, так что интерес представляет только вопрос «как должны выглядеть и действовать разделители». Для разграничения групп традиционно используют полосы. Это надеж -ное, простое решение, другой разговор, что с дизайнерской точки зрения полосы плохи, поскольку представляют собой визуальный шум. Гораздо правильное, но и труднее, использовать только визуальные паузы между группами, как это сделано, например, в MacOS X.

**Глубина меню.** Наличие многих уровней вложенности в меню приводит к там называемым «каскадным ошибкам»: выбор неправильного элемента верхнего уровня неизбежно приводит к тому, что все следующие элементы также выбираются неправильно. При этом широкие меню больше нравятся пользователям. Поэтому большинство разработчиков интерфейсов стараются создавать широкие, а не глубокие меню. К сожалению, у широких меню есть недостаток: они занимают много места. Это значит, что, начиная с определенного количества элементов, меню физически не сможет оставаться широким, оно начнет расти в глубину.

Возникает проблема, которую надо решать. Итак, проблема заключается в том, что велика вероятность каскадных ошибок. Чтобы снизить их число, нужно повысить вероятность того, что пользователи будут правильно выбирать элементы верхних уровней. Чтобы повысить эту вероятность, нужно заранее снабдить пользователей контекстом.

При перемещении по меню пользователь действует по определенному алгоритму:

- 1 Выбирая элемент первого уровня, он выбирает элемент, «нужность» которого кажется ему максимальной.
- 2 После выбора он видит список элементов второго уровня, при этом он оценивает вероятность соответствия всех элементов второго уровня его задаче и выбирает наиболее вероятный элемент. При этом в уме он держит контекст, т. е. название элемента первого уровня.
- 3 Если ни один из элементов не кажется пользователю достаточно вероятным, пользователь возвращается на первый уровень.
- 4 Если какой-то элемент удовлетворяет пользователя, он выбирает его и получает список элементов третьего уровня. Действия из второго и третьего шагов повторяются с новыми элементами меню.

Видно, что действия пользователя при поиске нужного элемента отчетливо цикличны, при этом на каждом шаге есть вероятность ошибок. С каждым новым уровнем меню объем контекста, который приходится держать в голове, непрерывно возрастает. При этом, если пользователь всё -таки не находит нужного элемента, весь этот контекст оказывается ненужным. Хранение же контекста, даже не засчитывая усилия, затрачиваемые на выбор

элемента, есть довольно существенная работа. Её объем лучше уменьшить.

Есть другой метод, и этот метод есть, пожалуй, лучшее, что дал интернет науке о проектировании интерфейсов: в качестве аннотации к элементу можно показывать наиболее популярные элементы следующего уровня. этом случае пользователь может сформировать контекст элемента, не перемещаясь внутрь этого элемента, при этом вероятность ошибочного перехода значительно снижается. Помимо уменьшения числа ошибок, такая система позволяет ускорить доступ к наиболее популярным элементам второго и последующих уровней.

В целом, ширина и глубина меню являются, пожалуй, наименее значимыми факторами. Гораздо важнее хорошая группировка, при этом как группировку, так и структуру дерева меню, всё равно лучше определять карточной сортировкой. Применительно же к раскрывающимся меню действует ещё один ограничитель глубины. Раскрывающиеся меню довольно тяжелы в использовании, поскольку требуют от пользователей достаточно тонкой моторики. Поэтому главное меню с более чем тремя уровнями вложенности просто невозможно.

## **Контекстные меню**

Преимущество контекстных (всплывающих) меню заключается в том, что они полностью встраиваются в контекст действий пользователей: не нужно переводить взгляд и курсор в другую область экрана, практически не нужно прерывать текущее действие для выбора команды. При этом они не занимают места на экране, что всегда ценно. С другой стороны, из-за того, что они не находятся всё время на экране, они практически неспособны чему-либо научить пользователя. **Не делайте контекстные меню единственным способом вызова какой-либо функции**

Поскольку основной причиной появления контекстных меню является стремление максимально повысить скорость работы пользователей, на их размер и степень иерархичности накладываются определенные ограничения. Если меню будет длинным, пользователям придется сравнительно долго возвращать курсор на прежнее место, так что привлекательность нижних элементов окажется под вопросом. Поэтому лучше сокращать размер контекстных меню до разумного минимума (порядка семи элементов). К тому же не надо забывать, что главное меню *не всегда* перекрывает выделенный (т. е. актуальный объект), а контекстное меню – *почти всегда* (как-никак оно вызывается на самом объекте).

В большинстве же случаев перекрытие актуального объекта нежелательно (сбивается контекст). Мы не можем сделать в этой ситуации ничего, кроме как уменьшить размер меню, в расчете, что маленькое меню будет перекрывать малое количество информации.

Разумеется, если точно известно, что оперируемый объект совсем уж мал, сокращать объем меню бесполезно.

Другая особенность контекстных меню – иерархия. В обычном меню иерархия имеет хотя бы одно достоинство: при обучении она позволяет упорядочивать элементы меню и тем самым делать его понятнее. В контекстных же меню обучающая функция не играет никакой роли, поскольку такими меню пользуются только опытные пользователи. Иерархия элементов теряет свое единственное достоинство, не теряя ни одного недостатка. Поэтому делать иерархические контекстные меню можно, ничего плохого в этом нет, но необходимо сознавать, что вложенными элементами почти никто не будет пользоваться (тем более что вложенность сбивает контекст действий). Система сначала должна показывать максимально релевантную информацию, затем всё остальное

Последнее отличие контекстных меню от обычных заключается в том, что в них очень важен порядок следования элементов. В главном меню не обязательно стремиться к тому, чтобы наиболее часто используемые элементы были самым первыми – все равно курсор придется возвращать к рабочему объекту, так что разницы в дистанции перемещения курсора практически нет. В контекстном же меню ситуация обратная – чем дальше нужный элемент от верха меню, тем больше придется двигать курсор. Поэтому правило релевантности в таких меню действует в полной мере.

## Окна

### Элементы окна

Окна, помимо областей с элементами управления, имеют некоторые общие элементы, главными из которых являются строки заголовка окна, строки статуса, панели инструментов и полосы прокрутки.

### Строка заголовка окна

У каждого окна есть строка заголовка. Человеку не свойственно обращать внимание на обыденность, особенно если эта обыденность не находится в фокусе его внимания (а строка заголовка как раз в нем не находится). Поэтому пользователи строкой заголовка интересуются весьма мало. В результате, пользователи 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. Дело в том, что текст и, в меньшей степени, пиктограмма заголовка играют важную роль в ПО (они заведуют переключением задач) и очень важную в web-интерфейсах (заведуют

навигацией). С переключением задач всё просто и сложно одновременно. Просто, поскольку правило тут простое «Релевантное выводится в первую очередь».

Поскольку пользователю нужен именно конкретный документ конкретной программы, а вовсе не программа просто (мы уже определили, что окна документов, не попадающие в переключатель задач, нехороши), названия документов, как более релевантные, нужно выводить в первую очередь. Наоборот, сложность состоит в том, что из-за жесткости интерфейса Windows много не сделаешь. Тем не менее, сокращать название программы нужно безусловно. Иная ситуация в web-интерфейсах. 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 на верх экрана.

При этом в web-интерфейсах нет проблемы с текстом заголовка – что хотим, то и пишем (стараясь не обращать внимания на то, что к этому прибавится название браузера). Строка статуса является, пожалуй, самым недооцененным элементом Строка статуса интерфейса (во всяком случае, способы её использования в web-интерфейсах существенно портят статистику). В то же время она заслуживает лучшей участи. Почему-то распространено мнение, будто строка статуса предназначена для того, чтобы информировать пользователей о значении тех или иных элементов интерфейса. Подразумевается, что если пользователь подводит курсор к какому-либо элементу, в строке статуса появляется краткое его описание.

На самом деле строка не может этого делать вообще: дело в том, что курсор находится в одном месте, а подсказка появляется совсем в другом, пользователю при этом приходится читать подсказку либо переводя взгляд, либо периферийным зрением. Разумеется, никто в таких условиях читать подсказку не будет, причем те, кто уверен, что строка статуса есть место для подсказки, чувствуют это прекрасно. Неудивительно, что разработчики строку статуса игнорируют. В действительности строка статуса предназначена для двух вещей: она может быть либо собственно строкой статуса, т. е. отображать текущее состояние системы, либо быть панелью инструментов для опытных пользователей (или же делать и то, и другое). Разберем это подробнее.

**Отображение текущего состояния системы.** Практически каждая система имеет свойства, либо зависящие от документа, либо изменяющиеся со временем. Например, в иллюстративных программах объекты имеют какие-либо свойства, причем не все эти свойства показываются. Другой пример: когда система долгое время занята, она должна показывать пользователю индикатор степени выполнения. И, наконец, самый простой



пример: пользователь текстового процессора имеет право знать, на какой странице документа он сейчас находится. Эффективнее всего выводить всё это в строке статуса.

**Панель инструментов для опытных пользователей.** Зачастую система обладает функциональностью, которая с одной стороны важна, а с другой – способна свести с ума неподготовленного пользователя. Обычно это касается не столько собственно функций, сколько режимов работы системы, о недостатках которых уже говорилось.

## Панели инструментов

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. **Панель инструментов нежелательно делать единственным способом вызова функции**

В настоящее время нет технической проблемы с помещением в панели произвольных элементов управления (остался только один ограничитель – размер помещаемых элементов), так что последние преграды, мешавшие делать сложные панели, исчезли. Этим стоит пользоваться, поскольку это позволяет экономить время, уходящее на открытие и закрытие диалоговых окон, и повышать интегральное качество взаимодействия с системой (пользователям *нравится* пользоваться сложными панелями).

**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. 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.

## 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 (благо, они не знали альтернатив). Затем появились графические интерфейсы. Первым делом были придуманы полосы прокрутки. К сожалению, оказалось, что они работают не слишком хорошо. **Пользователи ненавидят горизонтальные полосы прокрутки** Нечего и говорить, что пользователи избегают пользоваться полосками прокрутки (тем более что курсорные клавиши никто с клавиатуры не убирал).

Фактически, чуть ли не единственным применением этих полосок является перемещение «примерно к нужному фрагменту» при работе с большими документами. Разумеется, такое положение вещей никого особенно не радовало. Поэтому было 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. При перетаскивании – индикация страницы и название раздела. **Полосы прокрутки без индикации размера документа практически бесполезны** 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.

## **5 WEB-интерфейсы**

### **5.1 Особенности WEB-пространства**

Большинство людей не читают текст полностью а быстро просматривают в поисках нужной информации, информативные подзаголовки очень сильно сэкономят им время. Старайтесь выбрать подзаголовки максимально соответствующие следующему за ними тексту.

#### **Советы:**

##### **1. Разбивайте текст на небольшие параграфы**

Чтобы упростить чтение, разбивайте большие блоки текста на параграфы. За компьютером читатель редко концентрируется исключительно на тексте, обычно, параллельно работает IM клиент, открыты блоги, кричат дети. Если текст разбит на параграфы, читателю проще вернуться к нужному месту текста, после того как он отвлекся от экрана.

К тому же, если текст разбит на небольшие параграфы, читателю кажется, что он потратит на него меньше времени чем на тот же текст без разбиения.

##### **2. Используйте подзаголовки**

##### **3. Уберите все лишнее «Второй черновик = Первый черновик - 10%» — Стивен Кинг**

Многословность и красноречие хороши в поэзии, но малопригодны для текста в веб.

Точность, прямота и краткость — вот что нам нужно.

Например:

«Если вы хотите получить доступ к нашему замечательному разделу содержащему техническую информацию и советы, мы просим вас нажать кнопку на вашем внешнем устройстве типа мышь»

Далеко не столь удобно и приятно для пользователя как простая ссылка:

«Техническая поддержка».

#### **4. Выделяйте самое важное**

Не ленитесь выделить важную часть текста жирным, курсивом или вынести ее в сайдбар. Как и подзаголовки, это позволяет сэкономить время нужное читателю для поиска нужной информации или определения релевантности страницы.

#### **5. Используйте читабельный шрифт**

Красивый, нестандартный шрифт, конечно, позволит вам выделиться, но если из-за этого пользователь не может ничего прочесть, то от этого нет толка. Уникальные шрифты подойдут для баннеров или графики, шрифт основного текста сайта должен быть простым и достаточно крупным (особенно актуально последнее, нестандартные шрифты, к счастью, плохо поддерживаются браузерами, а со слишком мелкими приходится сталкиваться почти каждый день).

### **5.2. Содержание и структура WEB-страницы**

#### **5.3 Ссылки**

Гиперссылки позволяют людям переходить от страницы к странице, от сайта к сайту. Столь важный инструмент нельзя обойти вниманием при разработке дизайна, чтобы посетители могли легко ориентироваться на сайте, ссылки должны хорошо выделяться, быть абсолютно ясными и информативными.

##### ***Правила***

1. Текстовые гиперссылки должны хорошо выделяться на фоне остального текста.
2. Наведение мышки на ссылку должно вызывать эффект подсветки.
3. Текст ссылки должен быть, по возможности коротким, но достаточно длинным, чтобы четко описать следующее:
  - куда вы попадете.
  - что увидите
  - что произойдет
4. Гиперссылки на различные документы должны четко различаться.

5. Гиперссылки вызывающие неожиданные для пользователя действия должны об этом предупреждать, например:
- ссылки на файлы
  - ссылки, открывающие или закрывающие окна.

### ***Точный размер в гиперссылках***

Достаточно распространенная практика указывать точный размер в ссылке на файл: [PDF \(46,764 байт\)](#)<sup>3</sup> Все что нужно пользователю это знать как долго будет загружаться файл, несколько секунд или минут.

Нет никаких причин для указания более чем двух цифр в размере файла. В предыдущей ссылке достаточно указать 47KB. Старайтесь использовать килобайты или мегабайты для указания размеров файлов: 4.7KB, 47KB, 470KB, 4.7MB и т.д.

### ***Оформление гиперссылок***

Де факто установился стандарт согласно которому ссылки должны использовать синий подчеркнутый текст, который становится красным при нажатии, и пурпурным для посещенных ссылок.

Мне кажется, сбалансированным решением следовать стандарту де факто и использовать для ссылок цвет #00f или #00c, и красный с подчеркиванием при наведении на них мышки, в этом случае достаточно функциональны как отдельные ссылки в тексте, так и большие группы ссылок.

Наиболее читабельный способ, при использовании черного текста на белом фоне, выделять ссылки синим цветом. При таком оформлении они хорошо различимы в тексте, и достаточно контрастны, чтобы не ухудшать читабельность.

Напрашивается вопрос о доступности таких ссылок для людей, которые плохо различают цвета. Изображение представленное ниже показывает, что даже при отсутствии цвета сохраняется достаточный контраст между черным и синим, чтобы сделать гиперссылки заметными. При использовании подчеркивания ссылки выделяются немного сильнее, хорошим компромиссом может быть подчеркивание при наведении мышки.

**Подчеркивание** хорошо работает для ссылок в тексте, они выделяются сильнее, чем при использовании только цвета.

Но по мере роста **количества ссылок** подчеркивание теряет свои преимущества, хуже всего, если оно используется для списков ссылок.

Обратите внимание, что чтение списка ссылок без подчеркивания быстрее и проще. Для лучшего разделения тем немного увеличен интервал между элементами списка.

## 5.4 Навигация

Вот **три рекомендации**, которые позволят вам улучшить навигацию сайта для тех посетителей, которые пришли прямо на его внутреннюю страницу:

- **Сообщите пользователям, куда они попали, и как они могут перейти, на другие страницы сайта.** Для этого включайте в каждую страницу следующие три элемента:
  - Название компании или логотип в левом верхнем углу
  - Прямую ссылку на главную страницу
  - Средство поиска (лучше всего - в правом верхнем углу)
- **Сориентируйте пользователя по месту.** Сообщите ему, где он сейчас находится на сайте. Если у сайта иерархическая информационная структура, помещайте на старнице цепочную ссылку. Также разместите ссылки на другие ресурсы, которые каким-либо образом связаны с текущей страницей. Не засыпайте посетителя ссылками на ресурсы, которые не связаны с данным контекстом.

**Никогда не подразумевайте, что посетитель попал на данную страницу, преодолев весь путь от начала сайта.**

### Типы навигации

#### **Одноуровневый список**

Одноуровневая панель навигации (или навигационная панель в виде одноуровневого списка ссылок) представляет собой панель навигации, на которой размещены равноправные ссылки на разделы ресурса.

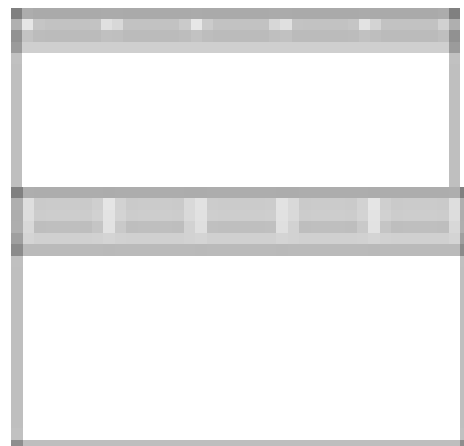
Этот тип панелей навигации условно можно условно разделить на три подтипа:

- горизонтальный одноуровневый список;
- вертикальный одноуровневый список;
- одноуровневая панель необычной формы.

### ***Горизонтальный одноуровневый список***

Панель навигации в виде горизонтального одноуровневого списка располагается вверху страницы. На некоторых сайтах горизонтальную панель располагают в двух местах: вверху и внизу страницы.

Рекомендуемое количество ссылок на одноуровневой горизонтальной панели — 5-7 штук (именно такое количество ссылок посетитель может с легкостью держать в голове). При большем количестве ссылок одного уровня пользователю сложнее ориентироваться в них. К тому же, такое количество ссылок обычно приходится размещать в два ряда, что создает дополнительные проблемы при оформлении страниц.



#### **Достоинства:**

1. После загрузки страницы все элементы списка полностью размещаются в верхней части страницы. Когда пользователь читает основной текст, меню исчезает из поля зрения и не занимает лишнего места.

#### **Недостатки:**

1. На панели нельзя разместить много элементов.
2. При большом количестве элементов списка их приходится размещать в несколько строк.

### ***Вертикальный одноуровневый список***

Панель навигации в виде вертикального одноуровневого списка чаще всего располагается в левой части страницы. При этом под панелью остается пустое место, которое часто используется под баннеры, формы опросов и т.п.

Рекомендуемое количество ссылок на панели все то же: 5-7 штук. Если ссылок все же больше, это не создает ощутимых проблем при дизайне страницы. Однако при слишком большом количестве ссылок часть их выходит за границу просматриваемой области, что вынуждает пользователя использовать полосу прокрутки.



#### **Достоинства:**

1. В вертикальной одноуровневой панели навигации можно безболезненно увеличивать количество ссылок, тогда как при горизонтальном расположении это приводит к проблемам верстки (для нового элемента может не хватить места).

#### **Недостатки:**

1. Под панелью навигации часто остается много неиспользованного пространства.

2. При большом количестве элементов списка они не все видны на странице одновременно (для просмотра некоторых элементов необходимо прокручивать страницу по вертикали).

#### **Двухуровневый список**

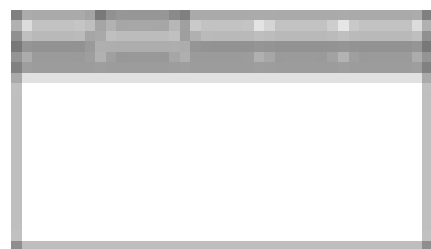
Двухуровневая панель навигации представляет собой панель, в которой каждой ссылке первого (корневого) уровня соответствует несколько ссылок второго уровня.

Все рассуждения относительно горизонтального или вертикального расположения панели навигации с многоуровневым списком аналогичны предшествующим.

#### ***Двухуровневый список с фиксацией***

В данном виде навигационной панели в каждый момент времени отображаются все ссылки первого уровня (одна из них является активной), а также ссылки второго уровня, которые соответствуют активной (выбранной) ссылке из первого уровня.

Таким образом, для получения доступа к любому элементу второго уровня, не соответствующего выбранному элементу первого уровня, посетитель должен сначала выбрать соответствующий элемент первого уровня (при этом произойдет обновление веб-страницы), а затем выбрать нужный элемент второго уровня (еще одно обновление страницы).



По способу размещения на странице этот вид панели также делится на два подвида: вертикальные и горизонтальные. Все соображения по способу размещения аналогичны приведенным ранее для одноуровневого списка.

#### **Достоинства:**

1. Всегда видно, в какой части сайта мы находимся.

2. По сравнению с одноуровневой панелью, позволяет разместить значительно большее количество ссылок.

### **Недостатки:**

1. При активном перемещении по разделам сайта происходит слишком большое количество перезагрузок страниц.

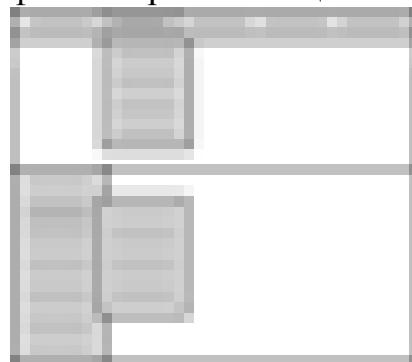
Одна из разновидностей двухуровневого списка с фиксацией — когда ссылки первого и второго уровня расположены подобно осям координат. Это расположение ссылок очень наглядное, но занимает много места на странице. Такую панель я назвал 2d-панелью навигации.



Она, например, использовалась на сайте компании РусАрт (сейчас Individ). К сожалению, на данный момент старый сайт компании недоступен.

### **Динамический двухуровневый список**

Панель навигации в виде динамического двухуровневого списка более всего напоминает стандартное меню графического интерфейса. При активации элемента верхнего уровня раскрывается подменю, состоящее из соответствующих элементов второго уровня.



Наиболее часто встречается горизонтальное и вертикальное расположение таких панелей навигации (соображения по этому поводу аналогичны приведенным ранее для панелей с одноуровневым списком), хотя иногда встречаются и другие решения.

### **Достоинства:**

1. Позволяет быстро получить доступ к любому элементу.
2. Занимает мало места на странице.

### **Недостатки:**

1. Многие реализации динамических навигационных панелей не

### **Развернутый двухуровневый список**

Идея применения развернутого двухуровневого списка — показать на странице все ссылки второго уровня. При этом ссылки первого уровня могут играть роль заголовков (иногда они не являются ссылками)



### **Достоинства:**

1. Посетитель сразу видит все ссылки панели навигации.



2. Т.к. ссылки сгруппированы, это облегчает посетителю поиск нужного элемента.

3. Данными типом навигационной панели так же просто пользоваться, как и панелью с одноуровневым списком.

#### **Недостатки:**

1. Панель занимает много места на странице.

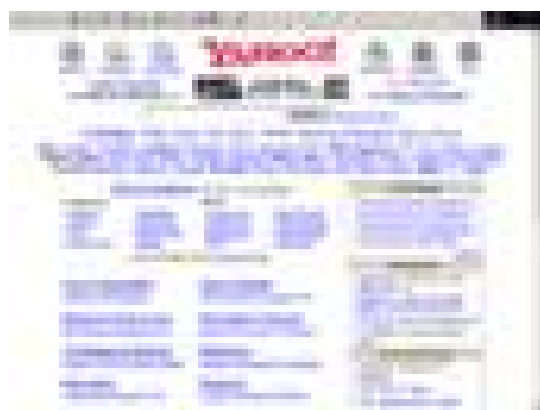
#### **Полуразвернутый двухуровневый список**

Панель навигации в виде полуразвернутого двухуровневого списка представляет собой список из ссылок первого уровня, рядом с которыми приведены несколько наиболее важных (или популярных) ссылок второго уровня. Этот вид навигационных панелей наиболее часто применяется на сайтах больших информационных ресурсов, обладающих разветвленной структурой, разделы которых неравнозначны по популярности (полезности) среди пользователей.

**Достоинства:** Эффективное использование площади сайта. Все наиболее существенные ссылки доступны с главной страницы.

**Недостатки:** Одни элементы найти очень просто, а другие — совсем нет.

#### **Пример:**



[www.yahoo.com](http://www.yahoo.com)

Самый известный в мире каталог интернет-ресурсов использует в качестве панели навигации полуразвернутый двухуровневый список.

*Ссылки, которые ведут напрямую к внутренним страницам*

*сайта, улучшают его юзабилити, так как в отличие от простых ссылок, они ведут посетителя непосредственно к цели. Веб-сайты должны использовать глубокие ссылки как можно чаще, и при этом следовать трем правилам.*

#### **Ссылки на дочерний элемент (Down-to-Child Links)**



Ссылки, обеспечивающие доступ к более специфичной информации; позволяют пользователю углубляться в толщу интересующего его раздела.

Ниже представлены собранные в мае 2002 года статистические данные о сайтах-лидерах, занимающихся электронной коммерцией. Все

исследованные сайты имели этот тип ссылок. Это число получено путем подсчета всех страниц, где ссылки на дочерние элементы могли бы быть.

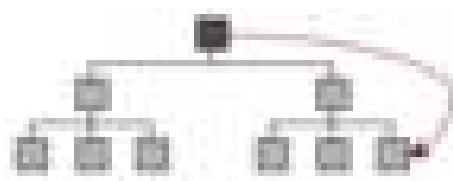
60% В области основного содержимого

27% Слева

7% Справа

3% Сверху

### Ссылки на "внучатый" элемент (Down-to-Grandchild Links)



Ссылки, позволяющие опуститься на два уровня ниже текущей страницы; дают возможность быстрого перехода к подразделам текущей категории.

15 % сайтов (из 75 – 2002 год) имели этот вид ссылок. Эта частота вычислена по количеству страниц, где ссылки на "внучатые" элементы могли встречаться.

Итак, мы рассмотрели некоторые типы навигационных панелей. В следующей части статьи мы поговорим о других элементах панелей навигации, а также об особенностях размещения панелей на странице.

### Другие элементы навигационных панелей

#### Выпадающий список

Выпадающий список представляет собой стандартный элемент графического интерфейса. Обычно его использование оправдано, когда необходимо предоставить возможность выбора одной ссылки из большого количества одноуровневых ссылок (когда они занимают слишком много места или вообще не помещаются на экран).



**Достоинства:** Можно поместить очень много одноуровневых элементов (в частности, благодаря использованию полосы прокрутки).

**Недостатки:** Возможно, выпадающий список несколько непривычен в качестве элемента навигации.

#### Раскрывающийся список

Раскрывающийся список имеет много общего с вертикальной динамической панелью навигации. Разница в том, что подуровни раскрываются вниз, сдвигая другие элементы



списка.

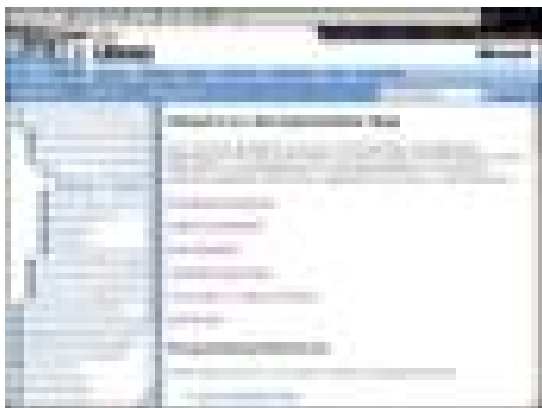
Обычно раскрывающийся список используется для отображения сложной структуры сайта, или, например, для перемещения по разделам большого документа.

**Достоинства:** С помощью раскрывающегося списка удобно отображать структуру очень больших документов.

**Недостатки:**

Если раскрывающийся список используется для перемещения по сайту с очень большим числом страниц, список с такой структурой может занимать значительное количество памяти (при этом посетителю сайта придется очень долго ждать загрузки списка). Для обеспечения нормальной (быстрой) работы пользователя, используется технология динамического скачивания структуры списка, когда в начале закачивается структура нескольких верхних уровней, а загрузка подуровней, соответствующих выбору пользователя, производится по мере необходимости.

**Пример:**



<http://msdn.microsoft.com/library/default.asp>

На сайте компании Microsoft в разделе “MSDN online Library” в качестве второстепенной панели навигации для перемещения по справочной библиотеке используется раскрывающийся список. Это позволяет достаточно быстро получить доступ к любому документу библиотеки, как бы глубоко он ни находился.

**Расположение навигации на странице**

Ранее мы рассмотрели различные типы навигационных панелей. Сейчас перейдем к обсуждению вопроса о выборе оптимального места расположения панели навигации на странице.

**Вертикальная панель — слева или справа?**

Существует два варианта расположения вертикальной панели — либо справа, либо слева страницы.

Обычно вертикальная панель навигации располагается слева. Такое расположение управляющих элементов более привычно для пользователя. В частности, это объясняется тем, что при чтении взгляд двигается слева направо, поэтому все, что расположено слева, воспринимается как сущность более высокого уровня.

Когда вертикальная панель навигации расположена справа, она несколько хуже воспринимается как управляющая панель, однако при чтении на нее чаще попадает и задерживается взгляд посетителя сайта. Таким образом, правостороннее расположение панели навигации сильнее привлекает внимание пользователя и стимулирует его на дальнейшее ознакомление с материалами сайта.

## **Горизонтальная панель — дизайн или функциональность?**

Для панели навигации, расположенной горизонтально, одним из критичных параметров является размер. При прочих равных условиях, больший размер панели навигации предоставляет больше возможностей для дизайнерских изысков. Однако при большом размере горизонтальной панели пользователю практически каждую страницу приходится прокручивать вниз, чего при меньшем размере панели могло бы и не потребоваться.

В связи с этим часто используется следующее решение. На главной странице сайта горизонтальная панель имеет значительно больший размер, чем на всех остальных страницах сайта. При этом мы имеем возможность в полной мере представить фирменный стиль компании, а пользователь не будет испытывать неудобства при продолжительном знакомстве с материалами сайта.

### **Панели необычной формы**

Панели необычной формы столь необычны (извините за каламбур), что невозможно дать им четкое определение. Как правило, это панели с небольшим количеством ссылок (часто одноуровневые), которые распложены в некотором нелинейном порядке и тесно сплетаются с другими графическими элементами оформления сайта.

Панели необычной формы часто используются на сайтах дизайнерских студий.

**Достоинства:** Позволяют сделать интересный дизайн (для этого они и предназначены).

#### **Недостатки:**

1. Как правило занимают неоправданно много места.
2. Такие панели навигации как правило менее удобно использовать.

### **Фреймовый дизайн**

Под фреймовым дизайном мы будем подразумевать не только дизайн с использованием технологии фреймов языка HTML, а любой дизайн, в котором используется несколько самостоятельных областей текста, одна или более из которых может прокручиваться независимо от остальных.

У сайтов с фреймовым дизайном меню обычно располагается в одном фрейме, а соответствующая страница с текстом — в другом. Поэтому при прокрутке текста меню остается на том же месте.

#### **Подвижная или неподвижная граница?**

При фреймовом дизайне можно разрешить пользователю перемещать границу между фреймами.

#### **Подвижная граница**

**Достоинства:** Пользователь может выбрать, сколько места будет занимать текст и, соответственно, панель навигации.

**Недостатки:** Лишние действия со стороны пользователя.

### **Неподвижная граница**

**Достоинства:** Пользователь не отвлекается на различные манипуляции с границей фреймов.

**Недостатки:** Невозможно убрать или уменьшить размер панели навигации.

## **5.5 Хлебные крошки**

**Хлебные крошки** (англ. Breadcrumb)— элемент навигации по сайту, представляющий собой путь по сайту от его «корня» до текущей страницы, на которой находится пользователь. Навигация, которой пользуются опытные покупатели чаще, чем основным меню сайта. И навигация, о которой тупо забывают большинство разработчиков интернет-магазинов.

По негласным правилам юзабилити — пользователь должен видеть где он в данный момент находится непосредственно в навигации. Т.е. мы обязаны показать пользователю как глубоко он залез в дебри архитектуры сервиса.

Когда ты наглядно видишь где находишься в данный момент, и имеешь возможность сразу «откатиться» на предыдущий этап — это замечательно. Лично у меня сразу возникает доверие к сервису.

### **Где их стоит использовать:**

1. В блог-площадках, в которых вложенность достигает четырех пунктов, а иногда и поболее.
2. На форумах — успешно применяется, проверено временем. Все популярные форумные движки уже имеют их автоматически встроенными.
3. На крупных корпоративных сайтах.
4. На крупных 2.0 сервисах, с большой вложенностью.
5. На информационных сайтах — часто, чтобы добраться до статьи, нужно отсеять много категорий, и хлебные крошки в этом определенно помогают.

6. Также идеально подходит для интернет-магазинов, при обозначении категорий и подкатегорий товара.

## 2 замечания:

1. **Не стоит** использовать хлебные крошки, если вложенность навигации менее двух порядков, иначе этот способ из хорошего в мгновение ока превращается в очень плохой, ведь если вложенность одного порядка — тогда непонятна логика этого манипулятора.

2. Если использовать этот способ навигации, нельзя делать на него акцент — в большинстве случаев он всего-лишь дополнение к основной навигации, и не должен перетягивать внимание на себя, но в то же время быть заметным.

Помимо этого, данная навигация удобна для пользователей тем, что они могут без труда подняться вверх по иерархии, используя соответствующие ссылки между разделителями! Это очень удобно и обычно используется людьми!

Еще одним способом применения «хлебных крошек» является визуализация процесса какой-либо операции. К этому способу обычно прибегают в клиентских зонах сайтов. Визуализируется же там процесс готовности клиентского заказа, например. Очень наглядное и удачное применение, скажу я Вам!

### Разделитель между звеньями (для горизонтальных цепочек)



65% Стрелка вправо



9% Вертикальная черта



8% Двоеточие



4% наклонная черта



3% Стрелка влево



3% Маркер



3% Видоизменённый текст

## 5.6 Использование многоаспектной классификации

В отличие от обычной (простой) иерархической схемы, многоаспектная классификация даёт пользователю возможность искать нужный элемент более чем по одной категории. Например, некоторые покупатели драгоценностей могут быть более заинтересованы в просмотре специфических категорий (серьги, ожерелья), в то время как другим более интересен просмотр по материалам (золото, серебро). "Материал" и "тип" - примеры граней/аспектов; серьги, ожерелья, золото и серебро - примеры содержимого граней.

**Многоаспектная классификация была использована в том или ином виде при создании 69 % сайтов.**

В четырёх товарных категориях (Computers, Gifts, Kitchen Ware, Music/Video) все сайты (из числа этих 69 %) использовали многогранную классификацию. В товарной категории "Office Supplies" ни один из сайтов многогранную классификацию не применял.

### **Расширенный поиск против многогранной классификации**

Многогранная классификация может быть использована в интерфейсе расширенного поиска и/или быть **составной частью навигационного интерфейса** всего сайта.

### **Карта сайта и индекс: что это такое и для чего это нужно?**

Карты сайта и индексы - формы дополнительной навигации по сайту. Они позволяют пользователю перемещаться по сайту, не используя глобальную навигацию. Представляя визуальную структуру сайта и взаимоотношения его частей, карта может помочь заблудившемуся и растерянному пользователю перейти на нужную страницу. Карты распространены намного шире, чем алфавитные списки (индексы), но и те и другие по праву занимают своё собственное место, выполняя свойственные им уникальные функции.

## 5.7 Карта сайта и индекс сайта

Карта сайта - не что иное, как модель структуры сайта, целиком размещённая на одной странице. Одним взглядом пользователь может охватить сразу все категории и разделы сайта. Визуализация структуры может быть

выполнена как реальная графическая карта или на основе только текста. Текстовые карты легче поддерживать, чем графические из-за дополнительных сложностей по внесению изменений в последние. Динамические карты, т.е. такие, с которыми пользователь должен выполнить какие-то действия для получения информации (например, "[гиперболические деревья](#)"), не столь эффективны, как простые упорядоченные иерархические списки. В создании карты сайта самое главное - сделать ее точной, понятной для просмотра и простой для понимания.

Карты отображают либо общую (только верхние уровни), либо подробную (все уровни) структуру сайта. Элементы, включённые в карту сайта, должны соответствовать элементам глобальной навигации. Названия одного и того же элемента в карте, глобальной навигации (а иногда и на одной и той же странице) должны быть одинаковыми. **Названия ссылок, которые являются сходными, но не одинаковыми, заставляют пользователя думать о том, попал ли он на нужную страницу или нет. Это заставляет лишнего пользователя лишнего раз задумываться о простейших вещах.**

В отличие от карты сайта, которая показывает его структуру, индекс сайта обеспечивает доступ к определённым страницам и разделам. Индексы наиболее удобны для поиска чего-то конкретного, уже известного пользователю. Как писал Фред Лейз (Fred Leise), "через индексы пользователь получает доступ к содержимому сайта, опираясь на собственный словарь понятий, а также он может найти вещи, обсуждаемые, но не упоминаемые в тексте". Кроме того, индексы обеспечивают более тонкую степень детализации материалов сайта, так как они в отличие от карт могут указывать на конкретные упоминания о людях, местах и вещах, а не на общие разделы. Индексы материалов сайта предоставляют возможность работать с материалами методом "от частного к общему", "снизу вверх".

Индекс сайта должен работать точно так же как работают обычные индексы, расположенные в конце книг. Фактически, копируя при проектировании вашего собственного индекса формат книжных индексов, вы сможете повысить его полноценность и удобство использования. Если ваша аудитория знает, как использовать одно (книгу), она поймёт, как использовать



другое (индекс). Следование графическим стандартам, принятым при составлении книжных индексов, (например, выравнивание по левому краю списка подразделов данной главы), так же поможет улучшить удобство чтения и использования индекса вашего сайта.

Индексы бывают нескольких разновидностей. Первый тип - индекс охватывает все разделы сайта. Также существуют индексы товаров, отсортированные по алфавиту или по инвентарному номеру. Такие списки-индексы в особенности полезны на сайтах, представляющих большое количество продуктов или сложные их наборы. Сайты университетов, подобные Гарвардскому или Йельскому, могли бы разместить индекс своих дочерних сайтов- сайтов отдельных факультетов, колледжей, клубов, входящих в состав университета.

Как вам узнать, что вашему сайту нужна карта или индекс, а может и то и другое? На маленьких сайтах, скорее всего, не нужно ни то, ни другое. В большинстве случаев глобальная навигация таких сайтов позволяет добраться до любой страницы. Большинство средних и больших сайтов должны, вероятно, иметь хотя бы карту. Текстовые карты сайтов требуют минимальных затрат в создании и обслуживании, и сослужат хорошую службу вашим пользователям. Если только ваш сайт не подобен каталогу Yahoo!, карта сайта - единственное место, где пользователь может увидеть на одной странице все главные разделы и подразделы вашего ресурса. Это особенно важно, если в навигации вашего сайта используется динамическое меню, пункты которого не видны, пока на них не наведён курсор мыши. Кроме того, использование карт может повысить эффективность коммерческих и информационных сайтов.

Для большинства средних и больших сайтов индексы какого-либо типа тоже могут оказаться ко двору. Для чрезвычайно больших сайтов представляется невозможным включить в индекс абсолютно всё. Такие индексы могут оказаться слишком огромными, чтобы быть эффективными. В них может быть включена лишь самая важная и часто используемая информация. Информационные сайты могут извлечь из индексов больше пользы, чем, например, коммерческие, в виду традиционно большего объёма размещённой на них информации. Однако коммерческие сайты, имеющие обширный

комплекс продаваемых товаров, должны всерьёз рассматривать возможность использования индексов с целью улучшения обслуживания своей аудитории.

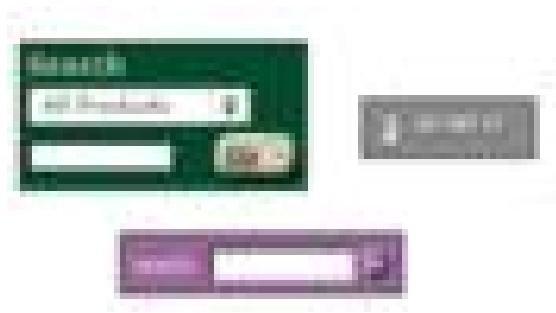
Решение об использовании индекса также зависит от того, какого рода информация размещена на сайте. Имеет ли смысл сортировать содержимое вашего сайта по алфавиту? Для сайта L.L. Bean, например, индекс ничем не поможет пользователям, так как они привыкли искать одежду по категориям (например, "Men's", "Camping", "Active wear"), а не отыскивая её в алфавитном списке. Для такого сайта лучше всего подходит именно карта, а не индекс.

Что бы вы не выбрали для своего ресурса: карту или индекс, или и то и другое, пользователь должен иметь к ним доступ из любого закутка вашего сайта. Располагайте ссылку всегда в одном и том же месте, например в подвале "страницы" или среди других элементов вторичной навигации. Если на вашем сайте имеется раздел помощи, включите ссылку и туда тоже - так как потерявшийся пользователь может не заметить ссылку в её основном местонахождении. Неплохо поместить ссылки на карту или индекс на странице с результатами поиска по сайту, особенно они пригодятся, если при поиске ничего не найдено.

Карты сайтов достаточно легко создавать и поддерживать, так как весь материал в них - вторичен. Все термины и пункты берут свое начало из главной и вторичной навигации сайта. Индексы создавать немного сложнее, так как к ним необходимо более осторожно подбирать ключевые слова. Для создания индекса нужен специальный человек, имеющий опыт составления указателей, и способный проанализировать содержимое сайта, выделив ключевые слова и определив взаимосвязи между разделами. С другой стороны, индекс товаров можно составить и автоматически средствами баз данных, XML, или другими подобными технологиями.

## **5.8 Некоторые сервисы (на примере сайтов интернет-магазина)**

### **Поиск (Search)**



Функция поиска позволяет пользователю ввести ключевые слова и операторы ("и", "или", "не") в поле ввода и затем отыскать информацию, удовлетворяющую заданным критериям поиска.

Положение блока поиска отмечено на диаграмме-сетке размером 800 на 600 пикселей.



Блок поиска был обнаружен преимущественно в левой или верхней частях страницы. Это лишь отчасти соответствует [исследованию](#), проведённому Майклом Бернардом (Michael Bernard). Исследование выясняло, где пользователи ожидают увидеть тот или иной элемент интерфейса коммерческого сайта. Согласно ему, участники исследования предполагали увидеть "Поиск" прежде всего в центре верхней части страницы.

### **Пиктограмма**

Только два сайта (3%) использовали пиктограмму для улучшения заметности блока поиска. В обоих случаях на пиктограмме было изображено увеличительное стекло.

### **Корзина (View Cart)**



Глобальная функция "Корзина" даёт пользователю возможность посмотреть информацию о товарах, помещённых им в его корзину для покупок.

Ниже представлены собранные в мае 2002 года статистические данные о 75 сайтах-лидерах, занимающихся электронной коммерцией. ([список сайтов](#))

### **Частота применения сайтами**

100 % сайтов в выборке предоставляли некоторые разновидности функции "Корзина"

### **Положение на странице**

Положение функции "Корзина" отмечено на диаграмме-сетке размером 800 на 600 пикселей.



Результаты близки к тем, которые [получил](#) Майкл Бернارد (Michael Bernard), исследуя, где пользователи ожидают видеть наиболее распространённые элементы коммерческих сайтов.

## **5.9 Верстка**

### **Модульная сетка**

Модульная сетка представляет собой набор невидимых направляющих,

вдоль которых располагаются элементы веб-страницы. Это облегчает размещение данных в документе, обеспечивает визуальную связь между отдельными блоками и сохраняет преемственность дизайна при переходе от одной страницы к другой.

Очень популярная сетка на информационных сайтах - трехколоночный вариант по пятиколоночному делению. Позволяет получать огромное количество внешних вариантов без каких-то влияний на общий стиль. Визуальные особенности сайта достигаются за счет цвета, шрифта заголовков, цвета панелей.

Еще один пример на тот же вариант, только в четырехколоночный макет. В общем-то чисто полиграфическое устройство сетки.

В мировой практике отработано несколько общих схем построения структуры, и практически все сайты используют один из подвидов со всевозможными модуляциями. Многие редакторы содержат готовые шаблоны с разными вариантами структуры страницы, но обычно такие шаблоны содержат стандартные структуры, а дизайн оставляет желать...

Мы можем внести некоторые принципы пропорциональности и баланса в наши страницы с помощью другой техники хорошо известной в печатном деле *базовой сетки*.

В печати это легко, просто включите базовую сетку в Quark или InDesign и установите необходимую высоту линий, в веб совсем другая история, достаточно сложно выровнять элементы по вертикали, поскольку мы не знаем где заканчивается каждый элемент, еще хуже то, что мы не можем задать размер шрифта, так как нам нужно. И все-таки применив немного математики и сообразительности, можно заставить этот метод работать на веб странице.

Страницы сделанные на сетке ей красивы, но очень сложно применять ее, если ваша разметка усложняется. Как и в печати, базовая сетка, не всегда правильный выбор для любой разметки, иногда вам нужно добавить блоки не соответствующие сетке, чтобы разметка заработала.

Тем не менее, использование базовой сетки в веб вполне возможно, она может быть особенно эффективна в случае использования нескольких колонок. Хорошо сбалансированная сетка, даже в пределах только главной колонки текста, может придать сайту блеск и улучшить читабельность.

Размеры отдельных блоков сильно зависят от общей таблицы и своего предназначения. Часто в качестве модульного размера для ширины колон употребляются стандартные размеры баннеров (не значит, что ширина колонки будет именно 88 или 468 пикселей, но какой-то принцип привязки к этим

модулям будет присутствовать). Наиболее часто употребляемые стандарты баннеров - 468x60, 88x31, 125x125 (западный стандарт, при котором высота может изменяться), 100x100 (стандарт, используемый в российской баннерной системе BB2). Есть и другие, менее распространенные стандарты.

## **Современные направления верстки**

### ***Центральное выравнивание***

Большинство современных сайтов используют центральное позиционирование относительно окна браузера. Значительно уменьшилась доля резиновых и левосторонних сайтов, по сравнению с прошлыми годами.

### **Почему центральное выравнивание?**

Стиль «2.0» простой, смелый и четкий. Сайты, которые размещаются по центру окна браузера, создают именно такое впечатление.

Поскольку мы стали более экономичны в использовании графики (и текста), больше нет необходимости втискивать в страницу информацию по самую ватерлинию и можно разбавить контент значительным количеством пространства.

### **Когда и как использовать центральное выравнивание**

Я же сказал, используйте центральное выравнивание всегда, если, конечно, у вас нет очень веской причины поступить иначе.

### **Небольшое количество колонок**

Несколько лет назад трехколоночные сайты были нормой, не редкость были и сайты использующие четыре колонки. Сегодня наиболее распространены сайты использующие 2 колонки, а три колонки стали практически максимумом.

### **Почему чем меньше колонок, тем лучше**

Чем меньше колонок, тем проще дизайн, мы передаем меньше информации, но более надежным способом.

Малое число колонок это еще и побочный продукт, преобладания сайтов с центральным выравниванием, если мы не пытаемся заполнить экран максимально возможным количеством информации, нам просто не нужно много колонок.

## **Как определить необходимое количество колонок**

Обычно рекомендуется не использовать больше 3 колонок, потому что следует использовать не больше колонок, чем действительно нужно.

Но всегда есть исключения, вот несколько примеров, где более трех колонок используются очень удачно.

### ***Отдельная шапка***

Конечно, в этом нет ничего нового, эта хорошая идея использовалась всегда, но сейчас она используется больше чем когда-либо, и разделение стало сильнее.

Обратите внимание, как четко выделяются шапки на этих 6 примерах.

### **Почему отдельная шапка это хорошо**

Верхняя часть страницы как бы говорит «Это верхняя часть страницы», это и так очевидно, но всегда приятно точно знать, где начинается страница.

### **Когда и как использовать отдельную шапку**

**На каждом сайте, брендинг и навигация должны быть очевидными, четкими и ясными.**

Всегда размещайте логотип вверху страницы, а основную навигацию сразу после него. Добавьте отступ вверху страницы, который позволит логотипу и навигации выглядеть более значимыми.

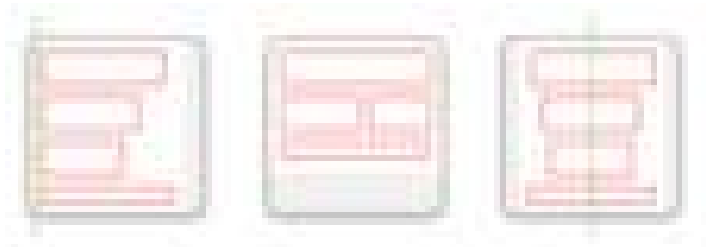
Хорошей идеей будет отделить шапку от остальной части страницы, например с помощью цвета, но возможны и другие варианты.

### ***Простая разметка***

Кажется, мы стали видеть больше простой 1 и 2 колоночной разметки, чем в предыдущие годы. Общее ощущение такое, что дизайнеры пришли к выводу, что простые страницы лучше работают. Эти страницы читаются просто сверху вниз, вам не нужно пропускать много лишнего и искать глазами, то, что нужно. Это более спокойный и удобный способ просмотра, чем раньше.

## ***Центральное расположение***

Другой момент, который бросается в глаза на всех сайтах представленных выше то, что они расположены в центре окна браузера. Пару лет назад вы могли найти множество примеров сайтов с резиновой версткой или расположенных слева с фиксированной шириной, теперь содержание переместилось к центру экрана.



Разметки с выравниванием по левому краю распространены гораздо меньше чем когда-то.

Также резиновые разметки стали менее популярны.

Всегда считалось полезным показать как можно больше информации «на обложке» (видимой на экране без скроллинга), резиновая разметка хорошо подходит для этой цели.

Тем не менее, сегодня мы видим что наличие скроллинга не критично, и сознательно идем на его использование получая много пространства и больший междустрочный интервал.

## ***Основа дизайна — содержание, а не страница***

Хорошие современные веб дизайнеры вкладывают меньше сил в разработку фоновых элементов, а напротив фокусируются на оформлении содержимого сайта. Это отражает принцип привлечения внимания пользователя к содержимому. Он выражается в следующем:

- Свободная менее сжатая разметка
- Легкая, простая «фурнитура»
- Яркие цвета и 3D эффекты используются, чтобы привлечь внимание к содержанию и бренду.



- Все направлено на то чтобы на сайте с лучшей стороны было показано содержимое, а не дизайнер (что гораздо полезнее для дизайнера в долгосрочной перспективе).

### ***Выделение областей цветом***

Рассматривая примеры раздела о шапке сайта, вы могли заметить, что многие сайты используют цвет, чтобы обозначить свои функциональные зоны, такие как:

- Навигация
- Фон
- Основной контент
- Ссылки
- Прочий материал

Дизайн должен быть разработан так, чтобы эти области четко различались, лучший способ достичь такого эффекта — использовать цвет, но и пространство может быть столь же эффективно.

Опасность использования ярких цветов в том, что они могут отвлечь внимание от других важных элементов сайта, в то же время, размещение контента на белом фоне делает его более читабельным и упрощает просмотр.

Итак, сформулируем основные принципы WEB 2.0.

1. Простота
2. Центральное выравнивание
3. Небольшое количество колонок
4. Отдельная шапка
5. Выделение областей цветом
6. Простая навигация
7. Четкие логотипы
8. Крупный текст
9. Крупный вводный текст
10. Яркие цвета
11. 3D Эффекты
12. Градиенты
13. Отражения

14. Оригинальные иконки
15. Вспышки звездочки

## 5.10 Тестирование WEB-страниц

### *Тестирование шаблонов (МЕТОД ИМИТАЦИИ)*

Удобство работы с отдельными страницами проверить достаточно легко: даете пользователям поработать со страницами и смотрите, понимают ли они все в них или испытывают проблемы. С шаблонами страниц совсем другая история. При традиционном тесте два аспекта накладываются друг на друга и их трудно разделить:

- вы проверяете удобство **компоновки**, которое задается дизайном шаблона
- вы проверяете удобство работы со специфическим **контентом**, который заливается в шаблон на каждой отдельной странице.

Перед тем, как раздавать шаблон большому числу разработчиков контента и давать им приказ использовать его в тысячах страниц сайта, неплохо было бы иметь побольше данных об удобстве работы с самим шаблоном. Под удобством работы с шаблоном (usability of template) я подразумеваю **то качество шаблона, которое делает страницы, построенные на нём, более (или менее) удобными в работе**. Также было бы интересно изучить, насколько дизайнерам страниц проще разрабатывать страницы имея готовый шаблон, но это уже другая тема.

Для того, чтобы проверить, помогает ли пользователям сама компоновка страниц найти различные элементы страницы, **весь текст страницы был заменен имитацией** - а именно, все слова были заменены непонятной чепухой, точнее **абракадаброй**. Когда прочитать текст не представляется возможным, пользователям для выполнения теста приходится полагаться только **на характерные аспекты компоновки страницы**. Если сама компоновка работает даже тогда, когда пользователи не могут понять содержимого

страницы, значит есть надежда, что шаблон сможет пережить "издевательства" редакторов, которые будут наполнять шаблон текстами различного качества.

Каждому из пользователей раздали по распечатке страниц с имитацией текста (в случайном порядке) и попросили обвести и подписать те части страницы, которые соответствуют девяти стандартным ее элементам. Блоки не могут накладываться друг на друга, и включать друг друга. Если пользователь считает, что какой-то из элементов отсутствует на странице, его следует пометить в списке как "нет". В исследовании использовались пять шаблонов, два из которых показаны ниже (воспроизведено с разрешения):



Шаблон страницы №1



Шаблон страницы №3

Каждый из шаблонов имеет следующие девять стандартных элементов, которые вы сейчас попытаетесь узнать:

- Главный контент данной страницы
- Заголовок страницы
- Лицо, ответственное за данную страницу
- Навигация по интра-сети (напр. главная страница сети, поиск)
- Дата последнего обновления
- Идентификатор или логотип сайта
- Навигация по данному сайту (напр. главные разделы данного сервера)
- Степень открытости информации (напр. открытая, конфиденциальная и т.д.)
- Перечень новостей сайта

Правильные ответы расположены по этой ссылке, но, пожалуйста, **попытайтесь решить эту задачу самостоятельно**, не подсматривая в решение. Если вы будете знать ответ, для вас тест станет бессмысленным.





Как показано в таблице, лучше всего пользователи справились с шаблоном №3, где они смогли правильно определить 67% стандартных элементов страницы. Также пользователям задавался вопрос, насколько по их мнению им нравится тот или иной шаблон по шкале от -3 до +3. Как показано в таблице, пользователям больше всего понравился шаблон №1 даже несмотря на то, что с ним у них возникали значительные проблемы. Этот результат доказывает, что опасно просто показывать людям пустой дизайн и спрашивать их мнение о нем: пользователи могут и сами не знать, что для них хорошо, а что плохо.

Так как по двум шкалам два различных дизайна набрали максимальное количество очков (результат работы и привлекательность), компания Fidelity решила совместить эти обе части дизайна в новом окончательном варианте. Когда тем же методом был проверен окончательный дизайн, оказалось, что он набрал больше очков, чем какой-либо из предыдущих вариантов. Следовательно, на лицо реально достигнутое улучшение.

**Метод имитации позволяет очень быстро собрать данные об удобстве работы с шаблоном страницы до того, как этот шаблон размножен в сотнях страницах. Так как пользователи не могут реально пользоваться текстом-абракадаброй, сами шаблоны даже не нужно верстать: достаточно каждый вариант создать как картинку, распечатать и помечать блоки прямо на бумаге.**

## **6 Проектирование non-WIMP интерфейсов**

Мобильные Интернет-устройства (MID - Mobile Internet Device) представляют собой новейшую разработку в семействе персональных компьютеров. Они потребляют мало энергии и имеют все преимущества ПК. При разработке функций и приложений MID необходимо учитывать множество аспектов, чтобы пользователь мог с удовольствием работать с интерфейсом MID. В данной статье дается обзор методов проектирования пользовательских интерфейсов для MID. Пользовательский интерфейс Hildon является задачей-ориентированным, так что требует особого подхода. В статье также предлагаются практические методы проектирования пользовательского интерфейса Hildon.

Мобильные Интернет-устройства (MID - Mobile Internet Device) представляют собой новую ветку в семействе персональных компьютеров. Они потребляют мало энергии и имеют все преимущества ПК. Мобильные Интернет-устройства – это устройства с малым форм-фактором, что дает возможность носить их с собой повсюду. При этом ультрамобильные устройства могут оставаться подключенными 24 часа в день, 7 дней в неделю.

MID имеют жидкокристаллический, относительно маленький (4.5” – 6”), с высоким разрешением, сенсорный экран и уменьшенную клавиатуру. Чтобы пользовательские ощущения были адекватны используемому устройству, контент для MID должен быть легко читаем, а пользовательский интерфейс легко управляться «одним пальцем». Это означает, что пользовательский интерфейс ОС и приложения должны иметь возможность развернутой настройки пользователем.

### **6.1 Проблемы проектирования пользовательских интерфейсов MID и практические методы работы**

Хотя архитектура ПК и полнофункциональная ОС для ПК упрощают разработчику задачу переноса с настольного ПК на MID, форм-фактор MID обуславливает некоторые уникальные аспекты проектирования пользовательского интерфейса для этих устройств. Кроме того, среда

приложений Hildon – это среда, адаптируемая под устройство, что также добавляет несколько специфических факторов в проектирование графического интерфейса приложений.

### Фактор размера экрана

Поскольку экран MID намного меньше, чем традиционный, к размеру графических элементов надо отнестись с особым вниманием. Разработчики должны избегать слишком малого размера уменьшаемых относительно традиционных компьютеров графических элементов, и слишком большого размера элементов, которые остаются прежними. В частности, текст и значки часто становятся плохо видны, некоторые кнопки или элементы трудно нажимать, а игровые окна плохо подходят к геометрии экрана.

### Размеры текста и значков

**Проблема:** в интерфейсах приложений, которые переносятся на MID, текст и значки могут быть сжаты до такой степени, что они становятся трудночитаемыми. Текст, размер которого выглядит вполне логичным на 15-дюймовом экране, может стать слишком маленьким при его урезании до масштабов экрана пять на семь дюймов, обычного для MID. Кроме реального размера шрифта текста, слишком маленькими могут стать чат и другие текстовые окна. Уменьшение размера шрифта для приспособления к меньшему размеру окна может сделать текст трудночитаемым.



(a)



(b)

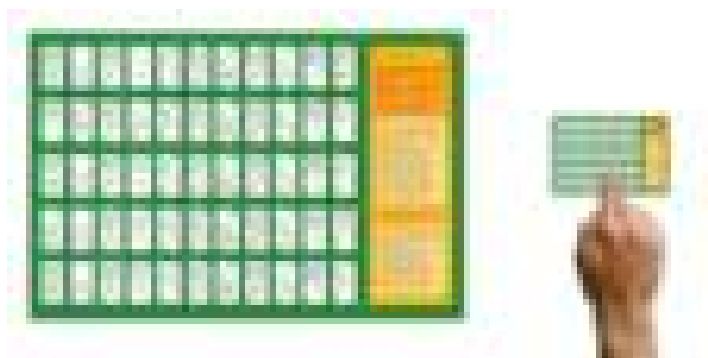
**Рис 3.** Подходящий размер шрифта для читаемого (a) и нечитаемого текста без адекватного масштабирования (b).

**Практический совет:** в идеале приложения, создаваемые с учетом использования на MID,

должны использовать текст ограниченно и учитывать при выборе размера шрифта размер экрана. А в значках не должно быть отличающих их друг от друга мелких деталей, так чтобы их нельзя было спутать на маленьком экране MID. Там, где увеличенный размер шрифта в достаточной мере согласуется с другими элементами приложения, возможность его настройки может быть хорошим выходом из положения. Отдельные пользователи тогда смогут сами решать, какой шрифт им нужен.

## Использование кнопок и других нажимаемых элементов

**Проблема:** аналогично проблеме с маленьким размером шрифта, кнопки и другие нажимаемые элементы добавляют сложности при переносе ПО на MID. Дело в том, что в том время как весь интерфейс программы для MID в целом должен быть меньше, чем версия для ПК, то реальные кнопки для MID должны быть больше, чтобы их удобно было «нажимать» пером или пальцем, которые менее точны, чем мышь в ПК. Даже если при определенном старании нужный элемент все же можно выбрать, эта проблема может заметно ухудшить впечатление пользователя от продукта.

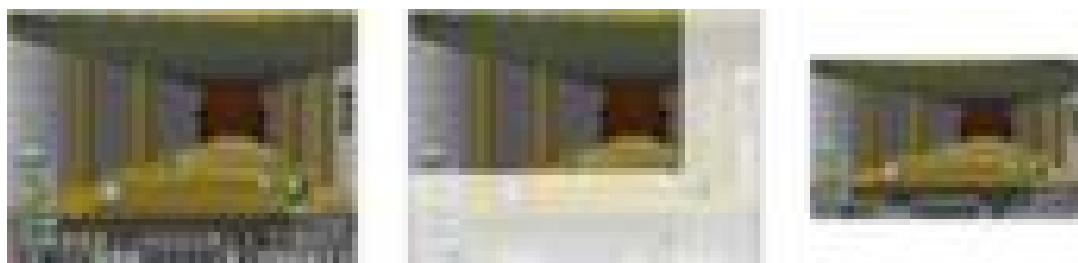


**Рис. 4.** Нажатие кнопок На 20-дюймовом ПК-мониторе (слева) пользователь может легко найти и выбрать кость домино, используя обычный курсор-стрелку. Когда изображения костей масштабируются до 5-дюймового экрана UMPC (справа), пользователю становится очень трудно выбрать каждую из них и даже определить, сколько точек на каждой кости.

**Практический метод:** также как и в случае с размером текста и значков, разработчики должны избегать этой проблемы, используя крупные, четкие кнопки и другие элементы. Между этими элементами должно быть достаточно места, чтобы пользователь не мог попасть в ненужную ему кнопку. А в тех случаях, когда к кнопкам и другим элементам привязаны текстовые ярлыки, эти ярлыки должны быть частью нажимаемой области, связанной с элементом. В этом случае нажимать на элементы будет проще, и не потребуются дополнительное пространство между ними.

## Размер окна

**Проблема:** Если окна приложения имеют фиксированный размер, а не изменяемый до полного экрана MID, некоторые их участки могут вдруг оказаться невидимы и, в конечном счете, недоступны. Эта ситуация осложняется необходимостью использовать свойственные широкому экрану MID разрешения 800x480 и 1024x600, поскольку эти форматные соотношения не совпадают со стандартной пропорцией ПК - 4:3.





**Рис. 5.** Видимость окна. На стандартном ПК (слева) пользователь может видеть все игровое пространство, включая информационные элементы интерфейса игрока. Если окно игры будет просто урезано до более мелкого экрана UMPC и другого форматного соотношения (в центре), то некоторые части игры будут невидны – здесь они показаны прозрачными полями по краям. Если окно игры будет просто урезано до более мелкого экрана UMPC и другого форматного соотношения (в центре), то некоторые части игры не будут видны – здесь они показаны прозрачными полями по краям.

**Практический совет:** всегда помнить о разрешениях 800x480 и 1024x600 при разработке пользовательских интерфейсов для MID, масштабируя исходное окно до размеров экрана или приспособив интерфейс так, чтобы он оптимально использовал широкоэкранный формат. Например, в некоторых случаях эффективным будет применение линейек прокрутки и разрешение изменения размеров окна (вручную или автоматически). Это даст пользователю доступ ко всему экрану и обеспечит ему приемлемые впечатления от использования приложения.

### **Фактор сенсорного экрана**

Использование на MID сенсорного экрана вместо мыши добавляет переносу приложений еще один уровень сложности. В большинстве задач сенсорный экран ведет себя как мышь, только вместо того чтобы регистрировать траекторию движения курсора, он во время касаний пользователем экрана создает серии событий нажатий кнопок с их координатами. Невозможность передвигать курсор без нажатия и связь между нажатием правых и левых клавиш на элементы интерфейса также могут вызывать проблемы.

### **Точная интерпретация ввода с сенсорного экрана только с нажатиями**

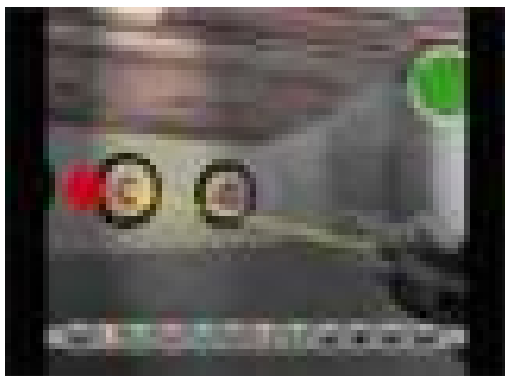
**Проблема:** приложения MID должны быть способны интерпретировать ввод курсора, производимый пользователем, как серию точек (соответствующую сериям нажатий на сенсорный экран), а не требовать линейных схем движений (тех, что дают движения мыши, контролируемые курсор). Эта проблема возникает особенно часто, например, в играх, которые меняют перспективу перед пользователем, так, что курсор всегда остается в центре экрана (обычная практика в «стрелялках» от первого лица). При таком сценарии, когда игрок касается экрана, курсор может бессистемно прыгать по экрану, вместо того чтобы переместиться в указанное пользователем место.

**Практический совет:** если игра отслеживает перемещения курсора из точки А в точку В, она должна быть способна понять нажатие в точке А, за которым следует нажатие в точке В, без того чтобы курсор сначала был перемещен из точки А в точку В. Эту проблему также можно решить требованием подключать к MID внешнюю мышь, хотя с точки зрения пользовательских впечатлений это не лучшее решение.

### **Точная разметка сенсорного экрана**

**Проблема (разрешение окна меньше, чем физический экран):** Когда приложение считает, что оно работает в полноэкранном режиме на разрешении меньшем, нежели физический

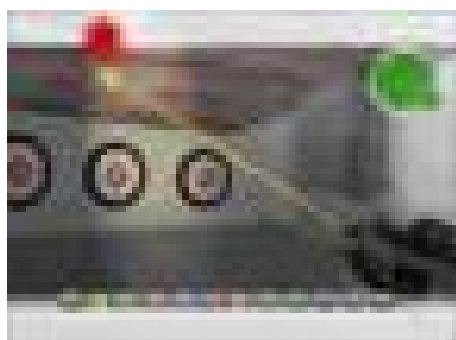
экран, дисплей графического интерфейса может появляться в середине экрана с пустым пространством на его сторонах. Так происходит и когда окно 640x480 размещается в центре экрана 800x480, и когда окно 800x600 находится на экране 1024x600 - по сторонам возникают черные полосы. При этом может возникнуть несоответствие разметки экранов, когда вся сенсорная поверхность будет соответствовать относительно маленькой площади дисплея. Несоответствие приводит к неадекватному отклику интерфейса на пользовательские действия, - например, когда нажимаемая область кнопки не совпадает с видимым представлением кнопки.



**Рис. 6.** Если игровое окно центрировано с черными полосами по двум сторонам, логика игры, которая интерпретирует экранные нажатия, должна избегать привязки всей физической области экрана к уменьшенному окну игры. Такая некорректная привязка может привести к тому, что нажатие экрана в одном месте (показано желтой стрелкой) будет понято как произошедшее в другом месте (показано красной вспышкой).

**Практический совет:** приложения можно центрировать с черными полосами по сторонам, не вызывая несоответствия разметок сенсорного экрана, если логика разметки будет считать черные полосы частью экрана. С этой поправкой прикосновения к экрану будет пониматься с точным соответствием месту на экране.

**Проблема (разрешение окна больше, чем экран):** в тех случаях, когда окно программы занимает больше места, чем полное вертикальное пространство экрана, сенсорный экран может быть неправильно размечен вертикально, а не горизонтально. В этом сценарии сенсорный экран размечается на все окно игры, а не только ее видимую часть. И здесь также нажатия не попадают в ту область окна игры, которую имел в виду пользователь.



**Рис.7.** Когда часть игрового окна (представленная прозрачной областью) выходит за пределы физического экрана, это может привести к тому, что нажатие на экран в одном месте (показано желтой стрелкой) будет понято игрой как нажатие в другом месте (показано красной вспышкой). Это происходит из-за несоответствия между разными формами игрового окна и физического экрана.

**Практический совет:** разработчик должен быть уверен в том, что операционная система размечает сенсорный экран точно в соответствии с видимой областью экрана приложения

(включая участки, не используемые при игре), и не включая участки игрового окна, которые не видны. Другое решение – изначально внедрить пользовательский выбор разрешений экрана (800x480 и 1024x600). В некоторых случаях возможно реализовать автоматическое растягивание окна до всего экрана, если нарушение пропорций элементов является приемлемым.

### Альтернативы эффектам всплывания

**Проблема:** Во многих играх используется функция всплывающих подсказок, которая позволяет игроку получать данные об объекте или определенном месте в игровом пространстве, подведя к нему курсор мыши и не нажимая на нее. Эта функция, которая часто удобна для обеспечения инструкциями начинающих игроков, несовместима с сенсорными экранами, поскольку без нажатия такой экран не может передвинуть курсор.



Рис. 8 . Эффекты всплывания.

**Практический совет:** поскольку функция всплывания не совместима с сегодняшним аппаратным обеспечением MID, разработчики должны искать альтернативные способы запуска анимации или подсказок. Например, они могут появляться тогда, когда игрок попадает на соответствующий им этап игры. Другой вариант – разработать интерфейс со средствами временной интерпретации событий нажатий как событий «мышь сверху». Например, это можно сделать с помощью клавиши, которую нужно удерживать, чтобы прикосновения к сенсорному экрану позволяли передвигать курсор без нажатий.

### Внедрение функций правой клавиши мыши

**Проблема:** на обычном сенсорном экране MID пользователи производят нажатия на правую клавишу мыши, удерживая указатель на экране на большее время, чем нужно для левого нажатия. Это может привести к тому, что пользователи нечаянно будут производить левое нажатие, думая, что делают правое. Более того, в этом случае невозможно одновременно производить правое и левое нажатия.

**Практический совет:** разработчики должны обеспечивать альтернативные способы взаимодействия пользователя с приложением, чтобы заменить функциональность правого нажатия, внедряя, например, двойные нажатия или нажимаемые контрольные элементы на экране. Кроме того, можно просто потребовать использования внешней мыши или другого указывающего устройства, хотя это может не лучшим образом сказаться на

пользовательском впечатлении от продукта.

### **Проблемы форм-фактора**

Ряд проблем связан непосредственно с отличиями аппаратного обеспечения MID от стандартных ПК. Например, в отличие от ПК, у MID может быть сенсорный экран, джойстик, программируемые пользователем и специальные клавиши, например, клавиши меню; и при этом у них обычно нет клавиатуры и CD-ROM. Хотя подключение клавиатуры, привода CD-ROM и других периферийных устройств может оказаться возможным, это снижает степень портативности MID. Более того, различные модели MID могут иметь различные элементы, что приводит к ограничениям, связанным с отличиями форм-факторов между этими устройствами. Эта разность обычно требует от разработчика сфокусироваться при создании приложения на наиболее часто встречающихся общих характеристиках оборудования, в смысле базовых требований к нему.

### **Ограничение числа способов управления**

**Проблема:** Форм-фактор MID предусматривает ограниченное число средств ввода команд по сравнению с полной клавиатурой и мышью, доступных на стандартном ПК. Даже те игры, которые разработаны для использования на ПК с джойстиком, могут потребовать использования нескольких кнопок стандартного джойстика, которые не обязательно будут доступны на MID. Устройства MID также плохо работают с играми, в которых в качестве важного игрового элемента присутствуют «горячие клавиши». Однако игры, в которых основным средством управления является мышь, не так сильно страдают от ограничений в этой области, связанных с форм-фактором.

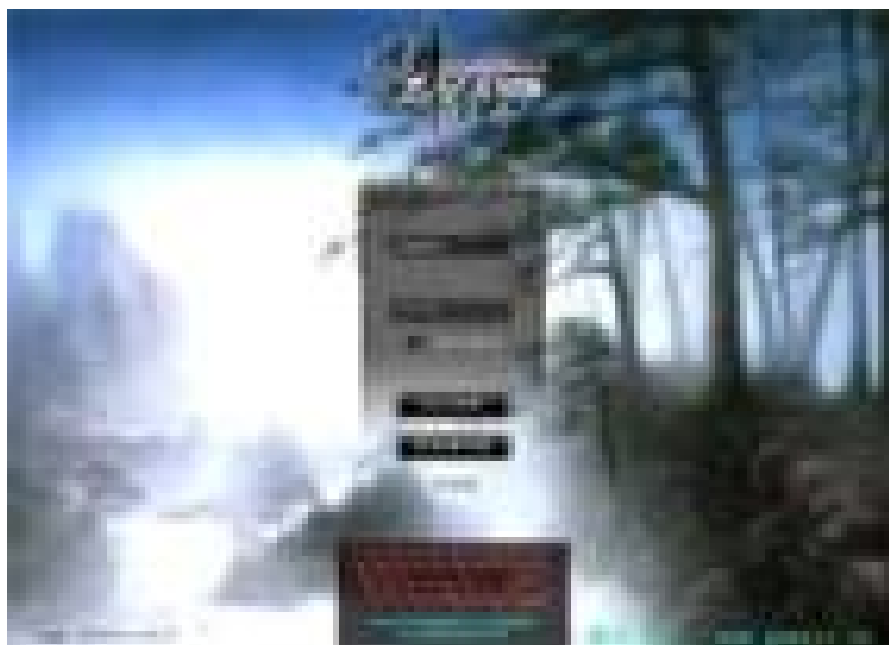
**Практический совет:** разработчики могут снижать количество средств управления в игре для MID, уменьшая число команд, которые в целом требуются для игры, вместо того чтобы внедрять дополнительные удобства, типа прямых команд для функций меню. Есть, однако, и более гибкое решение – создавать кнопки на экране. Это может стать пригодным для дальнейшего усовершенствования решением, особенно если кнопки будут контекстно-зависимы и появляться только там, где они нужны.

### **3.3.2 . Решение проблем, связанных с клавиатурой**

**Проблема:** многие игры используют клавиатуру не только для ввода команд, но и для ввода имен персонажей, создания профилей, сохранения игр или поддержки функции чата для онлайн-игр. Игры часто требуют клавиатуру, например, для ввода названия профиля в начале игровой сессии, но не требуют ее в течение всего остального времени сеанса. Некоторые игры также требуют от игроков вводить с клавиатуры имя при сохранении игры. Во многих случаях экранная клавиатура не будет работать поверх игрового интерфейса, так она не сможет разрешить эти проблемы.

**Практический совет:** разработчикам следует либо создавать игры, явно не конфликтующие с экранными клавиатурами (например, без установки по умолчанию полноэкранного режима)

или предусматривать такую клавиатуру в самом интерфейсе игры, чтобы она появлялась по мере надобности. Простое средство минимизировать необходимость во вводе с клавиатуры – предусмотреть значения по умолчанию для текстовых строк, таких как имена профилей, имена сохранения игр и т.д., и разрешить их выбор через нажатия на экран. Дополнительные значения для этих текстовых строк также могут предлагаться с помощью кнопки интерфейса, которая будет случайным образом выбирать новое значение из списка имен персонажей, созданного разработчиков. Сохраненные состояния игр могут идентифицироваться по снимку экрана, где игрок вышел из игры, вместе с датой и временем выхода. Поскольку функция чата является очень редкой частью игрового процесса, то на MID ее можно отключить без особого ущерба для пользовательских впечатлений.



**Рис. 9.** Экранная клавиатура при полноэкранном режиме.

Есть несколько различий между взаимодействием пользователя с ПК, которое ориентировано на действия мышью и клавиатурой, и интерактивной моделью Hildon, которую следует принимать во внимание при разработке пользовательских интерфейсов. При работе «на ходу» трудно точно попадать в нужные места на экране и работать с небольшими элементами управления на нем. Пользователи могут также закрывать области экрана, элементы управления или показываемый контент. Оптимизируйте свой дизайн интерфейса так, чтобы для выполнения задачи были нужны только несколько нажатий, и избегая ненужных обходных путей. Потоки задач в вашем интерфейсе должен следовать стандартно, как слева направо, так и сверху книзу экрана, а не заставлять пользователя возвращаться обратно или вверх.

## **Меню**

Команды меню должны быть структурированы согласно их важности и функциональным взаимоотношениям. Если у вас много команд, попробуйте сгруппировать функционально близкие команды в подменю и использовать на верхнем уровне только заголовки подменю, например 'Файл'->'Создать' и 'Файл'->'Сохранить'.

## Панель инструментов

Панель инструментов должна обеспечивать пользователю легкий доступ к наиболее частым и важным функциям в приложении. Вот несколько правил разработки и использования панелей инструментов:

- Число пунктов в панели инструментов зависит от ширины каждого пункта и от того, показывается или нет Навигатор задач (переключение с обычного экрана на полный).
- Если ваше приложение использует несколько видов экрана, то проанализируйте и оптимизируйте отображение функций в панели инструментов для каждого из них.
- На панели должны быть представлены основные значки с базовой функциональностью.
- Образы значков на панели инструментов должны быть понимаемыми, интуитивными и тесно связанными с действиями, на которые они ссылаются. Лучше всего, если они будут связаны с повседневной жизнедеятельностью, так чтобы пользователь мог ассоциировать задачи на устройстве с теми, которые знакомы ему из других ситуаций.
- Все отображаемые команды должны быть доступны в меню приложения.
- Из-за ограниченного размера экрана должна быть предусмотрена возможность скрывать панель инструментов командой из меню.
- При необходимости для группирования значков следует использовать разделитель.
- Избегайте использования на панели инструментов текста, который требует локализации, так как разная длина слов может нарушить макет панели. Везде, где это возможно, для экономии места на панели инструментов следует использовать значки, а не текст.



**Рис. 11.** Панель инструментов в Midinux. (браузер, плеер, E-mail, GPS, офисные средства, телефонная книга...)

## Диалоги

Диалоги или более длинные диалоговые пути можно применить, чтобы помочь пользователю завершить выполнение определенных задач. Не составляйте цепочки диалогов и делайте задачи простыми. Иногда хорошей

идеей может оказаться использование нового окна для более сложных задач. Если задаче требуются сведения из других приложений и информация не предоставлена в качестве заранее подготовленного списка, то лучше реализовать задачу в виде окна, а не в виде модального диалога. Диалоги должны располагаться в центре области приложения.



**Рис. 12.** Диалог в офисном средстве.

### Уведомления

Уведомления используются для организации обратной связи с пользователем. Есть два типа уведомлений: сообщения и баннеры.

Сообщения располагаются в центре экрана и содержат небольшой объем информации и кнопки для ответа. Приложение останавливает работу, пока пользователь не завершит диалог с сообщением.

Баннеры не требуют реакции от пользователя и не оказываются в центре его внимания. Они показываются в верхнем правом углу области приложения.

На рис. 3 показана разница между сообщениями и баннерами.



**Рис. 13.** Сообщение (слева) и баннер (справа)