

ЧЕЛОВЕКО- МАШИННОЕ ВЗАИМОДЕЙСТВИЕ

Учебное пособие для дистанционного образования

Сибирский государственный университет
телекоммуникаций и информатики

Новосибирск – 2009

СОДЕРЖАНИЕ

1. ЧЕЛОВЕК И КОМПЬЮТЕР	3
Человек	3
Зрение	3
Чтение	4
Слух	4
Осязание	4
Движение	5
Память	5
Умозаключение	7
Решение задач	7
Компьютер	8
Клавиатура	9
Сенсорные экраны	12
Графический планшет	15
Интерактивная доска	16
Жидкокристаллические дисплеи	16
Плазменные панели	19
OLED-дисплеи	21
2. ПРОБЛЕМНО-ЦЕНТРИРОВАННАЯ РАЗРАБОТКА ИНТЕРФЕЙСА	23
Анализ задач и пользователей	23
Выбор репрезентативных задач	24
Заимствование	25
Черновое описание дизайна	26
Обдумывание дизайна	26
Создание макета или прототипа	27
Тестирование дизайна с пользователями	27
Итерирование	28
Построение интерфейса системы	28
Отслеживание дизайна системы	29
Изменение дизайна	29
Управление процессом разработки	29
Проблемно-центрированный подход против метода "водопада"	29
Команда разработчиков	30
Ответственность	30
Требования практичности	30
3. SWT-АНАЛИЗ ИНТЕРФЕЙСА	31
Описание технологии	31
Что даёт SWT-анализ	34
Типичные ошибки при выполнении SWT-анализа и рекомендации	34
4. АНАЛИЗ GOMS	36
5. ЗОЛОТЫЕ ПРАВИЛА ПОСТРОЕНИЯ ИНТЕРФЕЙСОВ	42
Правила Нильсена-Молиха	42
Принципы организации графического интерфейса	45
РЕКОМЕНДУЕМАЯ ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА	49

Глава 1

Человек и компьютер

1.1. Человек

Когда мы говорим о человеке в плане его взаимодействия с компьютером, то, прежде всего, нужно отметить, что люди ограничены в своих способностях обрабатывать информацию. Это влечёт важные следствия для дизайна интерфейсов человеко-машинного взаимодействия. Информация принимается человеком и его реакции проявляются через различные каналы ввода-вывода, среди которых можно выделить визуальный канал, слуховой, осязательный и двигательный (моторный). Информация хранится в памяти, причём различается сенсорная память, краткосрочная (рабочая) память и долговременная память. Информация обрабатывается и формируется посредством умозаключений, решения задач, приобретения навыков и совершения ошибок. На человеческие способности влияют эмоции. Пользователи обладают общими способностями, но индивидуальны и имеют отличия, которые не должны игнорироваться. Теперь рассмотрим наиболее важные моменты более подробно.

Зрение

Зрение имеет две стадии: физическое восприятие стимула (воздействия) и обработка (интерпретация) стимула. Физическое восприятие осуществляется посредством глаза. Глаз может рассматриваться как механизм, принимающий свет и преобразующий его в электрическую энергию. Видимый свет отражается от объектов. Изображения фокусируются в перевернутом виде на сетчатке глаза. Сетчатка содержит палочки для видения при слабом свете и колбочки для цветного зрения. Кроме того, клетки ганглия в мозгу выявляют характерные шаблоны и движение. В интерпретации сигнала можно выделить несколько аспектов.

Во-первых, это размер и глубина. При восприятии человеком размера и глубины важны такие параметры, как визуальный угол (он показывает, как много места занимает объект в поле зрения и зависит от размера и расстояния от объекта до глаза) и острота зрения (способность воспринимать детали). Для человеческого восприятия характерно то, что знакомые объекты воспринимаются как имеющие постоянный размер (независимо от изменений визуального угла при достаточном отдалении), причём иллюзии, такие как наложение, помогают восприятию размера и глубины.

Во-вторых, это яркость, т.е. субъективная реакция на уровень света, излучаемого объектом; острота зрения увеличивается с ростом освещенности, также этому способствует и мерцание.

В-третьих, это цвет. Он создается из тона, интенсивности и насыщенности. Колбочки чувствительны к длине волны и индивидуально реагируют на красный, зелёный и синий цвета (RGB-система цветности). Известно, что наименьшая чувствительность проявляется к синему цвету. Нужно также помнить о том, что примерно 8 % мужчин и 1% женщин не различают цветов (являются дальтониками).

Зрительная система компенсирует реакции при движении и изменении освещенности. Для разрешения двусмысленности используется контекст. Благодаря компенсации иногда возникают оптические иллюзии. Два примера оптических иллюзий приведены на рис. 1.1. Верхний прямоугольник кажется больше нижнего (на самом деле они одинаковы), точно также верхний отрезок кажется длиннее нижнего.



Рис. 1.1. Оптические иллюзии.

Чтение

Чтение содержит несколько стадий: вначале воспринимается визуальный шаблон (модель), затем он декодируется с использованием внутреннего представления в языке, наконец, интерпретируется на основе знания синтаксиса, семантики и прагматики. При чтении задействуются как быстрое скачкообразное движение глаз, так и фиксация. Для распознавания важно очертание слова. Отрицательный контраст (черные буквы на светлом фоне) улучшает чтение с компьютерного экрана.

Слух

Слух даёт информацию о среде: расстояния, направления, объекты и т.д. Физическим аппаратом слуха являются внешнее ухо (защищает внутренне и усиливает звук), среднее ухо (передает звуковые волны с помощью вибраций к внутреннему уху) и внутренне ухо (высвобождает химические передатчики и вызывает импульсы в слуховой нерв). Звук характеризуется высотой (частота звука), громкостью (амплитуда) и тембром (тип или качество). Человек может слышать частоты от 20 Гц до 15 (некоторые – до 20) кГц, причем более высокие частоты различаются человеком с меньшей точностью. Слуховая система фильтрует звуки, что позволяет прислушиваться к звукам в окружающем шуме.

Осязание

Осязание обеспечивает важную обратную связь с окружающей средой. Оно может быть ключевым чувством для людей с плохим зрением. Стимулы принимаются рецепторами на коже: терморецепторы (тепло и холод), ноцицепторы (боль), механорецепторы (давление мгновенное и продолжительное). Некоторые области (например, пальцы) более чувствительны, чем другие. Кинетезис – ощущение положения тела. Действует на ощущение комфорта и производительность.

Движение

Время, требующееся для ответа на стимул, состоит из времени реакции и времени движения. Время движения зависит от возраста, физической формы и др. Время реакции зависит от типа стимула: для визуального стимула оно составляет примерно 200 мс, для слухового – 150 мс, для болевого – 700 мс. Улучшение времени реакции приводит к ухудшению точности для нетренированного оператора, но не для специально обученного. Закон Фиттса описывает время T , требуемое для попадания в цель на экране:

$$T = a + b \log_2(D/S + 1),$$

где a и b – константы, определяемые опытным путем, D – расстояние, S – размер цели. Из закона Фиттса следует, что для минимизации времени цели должны быть настолько можно большими, а расстояние – как можно меньшим.

Память

Сенсорная память хранит стимулы, полученные с помощью органов чувств. В результате человек может помнить изображения, звуки (музыку), тактильные ощущения и т.д. Эта память постоянно перезаписывается.

Краткосрочная память (КСП) является как бы записной книжкой для временного запоминания. Она характеризуется быстрым доступом (~ 70 мс), быстрым затуханием (~ 200 мс) и ограниченной ёмкостью (5–9 элементов). Например, попытайтесь запомнить следующие цепочки символов:

212348278493202

0121 414 2626

НЕС АТР АНУ РТН ЕТР ЕЕТ

Первое длинное число запомнить трудно, короткие числа и сочетания букв запоминаются легче. По мере запоминания новых элементов старые обычно забываются.

Долговременная память (ДВП) является хранилищем всех наших знаний. Она характеризуется медленным доступом (~ 120 мс), медленным или несуществующим угасанием, огромной или даже неограниченной ёмкостью. Различаются два типа ДВП: эпизодическая (последовательная память событий) и семантическая (структурированная память о фактах, концепциях, навыках и т.д.). Семантическая ДВП берёт свое начало в эпизодической ДВП. Структура семантической памяти обеспечивает доступ к информации, представляет взаимосвязи между единицами информации, поддерживает способность суждения (вывода). Одна из моделей этой памяти – семантическая сеть. Эта модель отражает наследование (дочерние узлы наследуют свойства родительских), явные отношения между частями информации, поддерживает способность производства вывода через наследование. Пример семантической сети приведён на рис. 1.2.

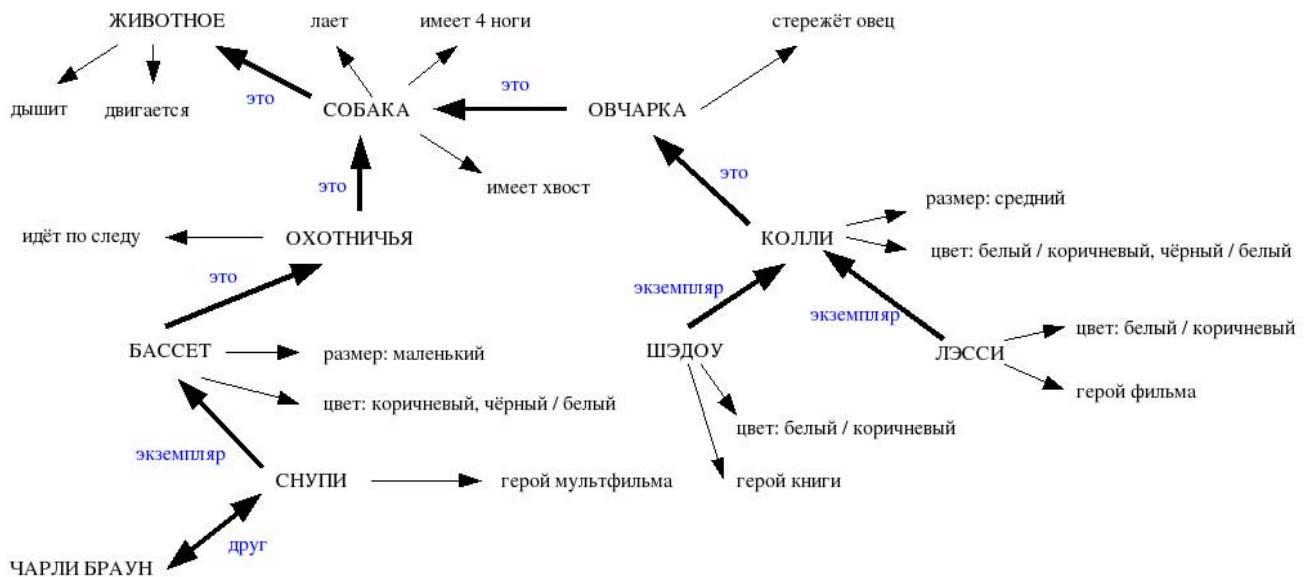


Рис. 1.2. Семантическая сеть.

Пример вывода: "Лэсси имеет 4 ноги и лает" следует из того, что Лэсси является собакой (овчаркой породы Колли), а собака лает и имеет 4 ноги.

Другая модель – фреймовая. В этой модели информация организуется в структуры (записи). Слоты в структурах заполняются значениями экземпляров данных. Структуры отражают отношения тип–подтип.

Struct СОБАКА		Struct КОЛЛИ	
Fixed	Ноги: 4	Fixed	Наследует от: СОБАКА Тип: овчарка
Default	Диета: плотоядная Звук: лай	Default	Размер: 65 см
Variable	Размер: Цвет:	Variable	Цвет:

В качестве важнейшего способа сохранения информации в ДВП выступает заучивание. Во время заучивания информация переносится из КСП в ДВП. Существует гипотеза "полного времени": количество информации, остающейся в ДВП, пропорционально времени (продолжительности) заучивания. Сохранению информации в ДВП также помогает эффект "распределения практики": сохранение оптимизируется путем распределения обучения во времени. Кроме того, имеет значение структура, значение и "знакомость" информации: эту информацию легче запомнить.

Важно иметь в виду особенности забывания информации в ДВП. Для ДВП (как и КСП) характерно угасание: информация теряется постепенно и очень медленно. Действует также эффект вмешательства (интерференция): новая информация замещает старую (ретроактивная интерференция) или старая информация перемешивается с новой (проактивная интерференция).

интерференция). Человек не может забыть информацию волевым усилием. Но волевое усилие иногда помогает вспомнить информацию.

Можно выделить два способа получения информации из ДВП: воспоминание (воспроизведение информации из памяти может упрощаться с помощью аллюзии (намёка), например, категории, образа) и узнавание (информация даёт знание того, что было запомнено ранее). Узнавание менее сложно, чем воспоминание – здесь сама информация является ключом.

Умозаключение

Различаются три способа выработки умозаключения: дедукция, индукция и абдукция.

Дедукция выводит логически необходимое заключение из заданных допущений. Например, (1) по пятницам она работает; (2) сегодня пятница; значит, сегодня она работает. Дедуктивный метод мышления необязательно приводит к правильным заключениям: всё зависит от истинности начальных допущений. Например, (1) когда идёт дождь, дорога сухая; (2) сейчас идёт дождь; значит, сейчас дорога сухая.

Индукция обобщает случаи известные на случаи неизвестные. Например, все слоны, которых мы видели, имеют хоботы; следовательно, вообще все слоны имеют хоботы. Этот способ мышления очень полезен для человека, но он ненадёжен. В действительности, индуктивно можно строго доказать лишь негативные утверждения: мы видели слона с хоботом, следовательно, утверждение о том, что слоны не имеют хоботов, неверно.

Абдукция означает суждение о причине по следствию. Например, апельсины оранжевые; я вижу оранжевый плод; наверное, это апельсин. Такие суждения также свойственны человеку, но ясно, что они могут приводить к ложным заключениям.

Решение задач

Решение задач – это процесс нахождения решения ранее неизвестной задачи с использованием знаний. Мы не будем рассматривать модели и теории, описывающие решение человеком сложных задач. Известно, что здесь играет роль такая трудно формализуемая и не до конца понятная науке вещь, как интуиция или озарение. Для построения человеко-машинного взаимодействия чаще важны следующие методы решения задач человеком

- *Решение по аналогии.* При возникновении новой задачи человек использует знания похожей задачи в той же или близкой области. Решение по аналогии затрудняется, если новая и известная задачи принадлежат семантически различным областям.
- *Решение за счёт приобретения навыков.* Если человек долго тренируется в решении задач, он начинает решать многие новые задачи более успешно и быстро. Считается, что ключевым моментом здесь служит приобретение способности разбивать задачу на элементы, что оптимизирует использование КСП.

В своих действиях человек нередко совершает ошибки. Важно различать два типа ошибок. Первый тип скорее можно назвать упущениями: эти ошибки возникают, когда имеется правильное намерение, но результат не верен в силу, например, плохих физических навыков или невнимательности. Второй тип ошибок возникает, если само намерение было не верным. Это обычно связано с неправильным пониманием.

Другой вещью, воздействующей на способность человека решать задачи, являются эмоции. Мы можем разделить их на положительные и отрицательные. Известно, что положительные эмоции способствуют творческому мышлению, а отрицательные губят его, делают мышление более узким. Вот несколько следствий, учитывающихся при разработке интерфейсов:

- Стресс увеличивает трудность решения задачи.
- Спокойные пользователи более благосклонно относятся к огрехам дизайна интерфейса.
- Эстетически приятный и полезный интерфейс вызывает положительные эмоции.

При разработке интерфейсов следует учитывать то, что люди индивидуально различны. Можно выделить долговременные различия (пол, физические и интеллектуальные способности), кратковременные (эффект стресса или усталость) и меняющиеся (возраст). Разработчик интерфейса может спросить себя: будет ли его дизайн исключать часть популяции пользователей?

Из всего сказанного о человеке следует сделать вывод о том, что физиологические и психологические его особенности обязательно должны учитываться при разработке интерфейсов компьютерных систем. Некоторые факты приводят к прямым приложениям. Например, низкая острота зрения в области синего цвета приводит к следующему требованию: синий цвет не должен использоваться для важных деталей интерфейса. Но вообще же, корректное приложение требует понимания психологического контекста и конкретных условий работы.

1.2. Компьютер

В этом разделе мы не будем изучать полное устройство компьютера, а рассмотрим только те его части или устройства, которые непосредственно используются для взаимодействия с человеком. Выделим следующие группы устройств:

Устройства ввода (ввод текста, рисование, выбор объектов на экране)

- Ввод текста: компьютерная клавиатура, телефонная клавиатура, речевой ввод, рукописный ввод.
- Указание: мышь, сенсорная панель, перо.
- 3D устройства.

Дисплеи:

- Компьютерные дисплеи
- Большие дисплеи
- Маленькие (специализированные) дисплеи
- Сенсорные экраны

Системы ввода-вывода в виртуальной реальности и играх: джойстики, рули, шлемы и т.д.

Бумажный ввод-вывод

- Принтеры
- Сканеры
- Распознавание текста

Устройства ввода-вывода физических параметров

- Сенсоры

- Элементы управления
- Звук, запах и устройства тактильного воздействия

Некоторые из перечисленных устройств, такие как клавиатура и мышь, настолько хорошо знакомы всем пользователям и программистам, что нет нужды описывать их подробно. С другой стороны, такие устройства, как сенсоры и различные органы управления настолько экзотичны и специализированы в различных технологических системах, что их изучение также выходит за рамки нашего курса. Ввод и вывод запахов пока остаётся нерешённой технической проблемой.

Ниже мы дадим описание некоторых не столь широко известных устройств или их разновидностей, представляющих, однако, большой интерес с точки зрения построения интерфейсов человеко-машинного взаимодействия. Мы кратко опишем принципы действия этих устройств, т.к. их знание необходимо для понимания особенностей их применения и возможностей реализуемых функций интерфейса.

Клавиатура

Сегодня основная клавиатура в большинстве компьютеров – это кнопочная QWERTY-клавиатура. Такие клавиатуры занимают доминирующее положение на рынке. Не нужно долго рассказывать, как они работают и как ими пользоваться. Однако, отметим, что размещение символов по клавишам (keyboard layout) может быть различным. Так для английского языка более эффективно размещение, которое разработал американский психолог Дворак (рис. 2.1). Утверждается, что при работе с клавиатурой Дворака скорость ввода увеличивается на 10–15 % и уменьшается усталость. По оценкам американских исследователей, в течение 8-часового рабочего дня руки наборщика проходят 16 миль на клавиатуре QWERTY, и только 1 милю на клавиатуре Дворака.

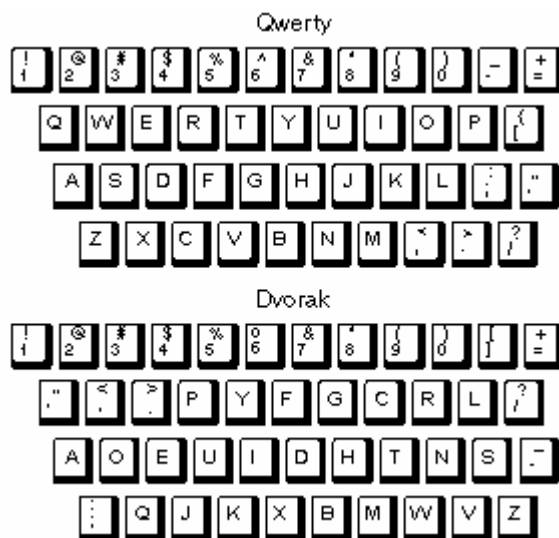


Рис 1.3. Разные клавиатурные раскладки.

Виртуальная клавиатура (рис. 1.4) – это клавиатура, создаваемая путем проецирования полноразмерного изображения клавиатуры на какую-либо поверхность. Устройство, кроме проецирующего лазера, также снабжено видеокамерой. Видеокамера считывает движения пальцев пользователя, по этим данным компьютер определяет необходимые действия и генерируемые символы. С точки зрения пользователя прикосновение к изображению клавиши генерирует уникальный код, соответствующий изображению этой клавиши.

Виртуальные клавиатуры оказываются незаменимыми для смартфонов и устройств PDA (коммуникаторов). Имеются также реализации музыкальных виртуальных клавиатур.



Рис. 1.4. Виртуальная клавиатура для устройства PDA.

Предлагается множество специальных клавиатур, предназначенных для управления различными техническими системами, работы в составе технологического оборудования или для людей с ограниченными возможностями. На рис. 1.5 приведен пример клавиатуры для набора только левой рукой.



Рис. 1.5. Клавиатура для левой руки.

Другим типом клавиатуры, с которым сегодня многие люди имеют дело, является телефонная клавиатура (рис. 1.6). Здесь за каждой клавишей закреплено несколько символов. Нужный символ выбирается путём многократного нажатия. Достаточно длинная пауза между нажатиями означает завершение ввода символа. Обычно отдельно задаётся текущий режим: цифры, буквы, специальные символы, язык ввода (детали меняются в зависимости от модели телефона). Например, чтобы набрать слово *computer* (в режиме ввода букв), необходима следующая последовательность нажатий:

222666 [пауза] 6788 [пауза] 833777.

Такой тип клавиатуры привел к появлению совершенно новой технологии ввода, называемой Predictive Text (предсказуемый текст). Рассмотрим основные идеи этой технологии на примере наиболее распространённого алгоритма T9.



Рис. 1.6. Клавиатура мобильного телефона.

Назначение алгоритма T9 – упростить печатание текстовых сообщений. Он позволяет вводить слова путём однократного нажатия для каждой буквы. Так, например, слово computer может быть введено последовательностью нажатий 26678837. В данном случае количество нажатий равно числу букв в слове, причем при наборе нет необходимости делать паузы.

Алгоритм объединяет группы букв, соответствующих каждой отдельной клавише, с быстродействующим словарём, хранимым в памяти телефона. В словаре отыскиваются все слова, соответствующие последовательности нажатий. Найденные слова упорядочиваются по частоте использования, и на экран выводится слово, стоящее первым в списке. Например, по мере нажатий на клавиши мы можем видеть на экране такую последовательность слов:

Нажатая клавиша	Слово на экране
2	a
6	an
6	ann
7	amos
8	boost
8	comput
3	compute
7	computer

Пользователь соглашается с предложенным словом путём нажатия клавиши ►. Иногда набранная последовательность соответствует нескольким возможным словам. Например, 4663 может означать good, gone, home, goof и т.д. (такие слова называются тайнонимами (T9onyms) и здесь они показаны по убыванию частоты встречаемости). С помощью предопределённой клавиши (например, *) пользователь может пролистать список и выбрать нужное слово. Алгоритм отслеживает частоту использования каждого слова. Если,

например, мы часто используем слово home, то оно постепенно перейдёт в начало списка, и сразу будет предлагаться при наборе комбинации 4663. Словарь может расширяться за счёт добавления в него новых слов.

Другие возможности алгоритма T9 включают в себя «умную» пунктуацию, автозавершение слов, выбор первичного и вторичного языков. Некоторые реализации накапливают статистику по часто используемым парам слов и обеспечивают предсказание слов (например, если вы часто пишете «всего доброго», то после ввода слова «всего» алгоритм автоматически предложит слово «доброго» и вы можете принять этот вариант путём нажатия ►).

Сенсорные экраны

Сенсорные экраны – это по сути дела дисплеи (жидкокристаллические или иные) с наложенными на них прозрачными устройствами координатного ввода.

Базовый вариант сенсорного экрана (рис. 1.7) состоит из трех основных компонент: сенсорной панели, контроллера и программного обеспечения (драйвера). Сенсорный экран – это устройство ввода, поэтому он должен быть объединён с дисплеем и компьютером или другим устройством для получения завершённой системы сенсорного ввода.



Рис. 1.7 Сенсорная панель (1), контроллер (2) и дискеты с драйверами (3).

Сенсорная панель – это прозрачная стеклянная панель с поверхностью, чувствительной к прикосновению или нажатию. Она помещается поверх экрана дисплея так, что чувствительная зона панели покрывает область изображения экрана. Сегодня существует несколько технологий изготовления сенсорных панелей, различающихся методами выявления прикосновений. Как правило, через панель пропускается электрический ток или другой сигнал, и прикосновение приводит к изменению напряжения или сигнала. Это изменение используется, чтобы определить место (координаты) прикосновения.

Контроллер – это небольшое устройство, подключаемое к компьютеру, преобразующее данные от сенсорной панели в информацию, понятную компьютеру. Если сенсорная панель интегрирована с монитором, то контроллер обычно устанавливается внутри корпуса монитора. Если же панель является добавочным устройством, то контроллер размещается в небольшом отдельном боксе. Большая часть контроллеров подключаются к компьютеру через последовательный интерфейс (COM-порт или USB).

Драйвер встраивается в операционную систему компьютера и позволяет компьютеру и сенсорной панели работать вместе. Драйвер говорит системе, как интерпретировать информацию о прикосновениях к панели. Большинство драйверов сегодня эмулируют поведение мыши. Это делает прикосновение к экрану подобным нажатию кнопки мыши в соответствующем месте экрана. Такой подход позволяет использовать существующее программное обеспечение, и новые приложения разрабатываются без какого-либо специфического программирования.

Резистивные сенсорные панели состоят из стеклянной или акриловой подложки, сверху которой находятся проводящий и резистивный слои, разделённые невидимыми разделительными точками (рис. 1.8). Когда к панели прикладывается давление, слои соединяются в точке, обеспечивая электрический ток в резистивном слое. Сначала напряжение к резистивному слою прикладывается вдоль оси X, напряжение, измеряемое в точке касания, пропорционально координате точки на оси. Затем напряжение прикладывается вдоль оси Y и аналогичным путём получается координата Y. Контроллер преобразует измеренные напряжения в численные значения координат и передаёт их в компьютер.

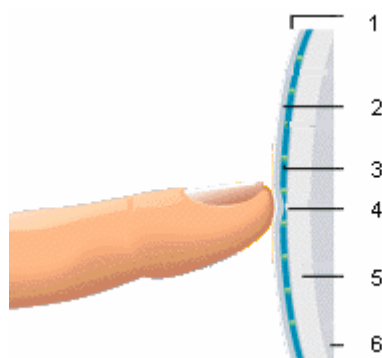


Рис. 1.8. Устройство резистивной сенсорной панели. Давление пальца вызывает электрический контакт между проводящим и резистивным слоем.

- 1 – Покрытие, защищающее от царапин
- 2 – Проводящий слой
- 3 – Разделители
- 4 – Резистивный слой
- 5 – Стекло
- 6 – Монитор

Преимущества резистивных панелей состоят в высокой разрешающей способности, возможности работать с любым типом пера, нечувствительности к пыли, грязи, воде и свету. Основной недостаток – прозрачность составляет только 75 %, поэтому необходима повышенная яркость дисплея. Кроме того, резистивный слой может быть испорчен острым предметом.

Резистивные и проводящие слои в сенсорных панелях делаются из вещества, называемого ИТО (Indium Tin Oxide), обладающего одновременно прозрачностью для видимого света и электрической проводимостью. ИТО представляет из себя твёрдый раствор оксида олова SnO_2 (10 %) в оксиде индия In_2O_3 .

Емкостные сенсорные панели используют тот факт, что кожа человека проводит электричество, а тело обладает электрической ёмкостью. Рассмотрим принцип действия емкостной панели на примере экрана смартфона iPhone (рис 1.9).

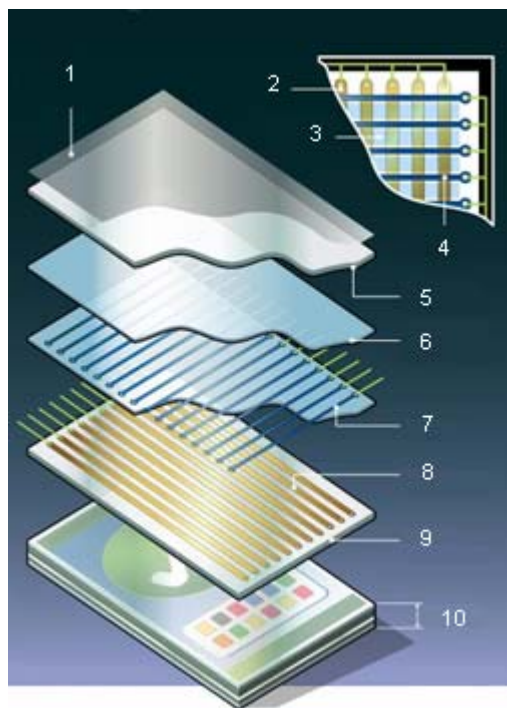


Рис 1.9. Устройство емкостного экрана в iPhone.

- 1 – Защитное антибликовое покрытие
- 2 – Считывающие линии
- 3 – Изолирующий материал
- 4 – Линии возбуждения
- 5 – Защитная крышка
- 6 – Соединительный слой
- 7 – Линии возбуждения
- 8 – Считывающие линии
- 9 – Стеклоподложка
- 10 – Жидкокристаллический дисплей

Когда пользователь прикладывает или лишь подносит палец близко к поверхности экрана, он изменяет электрическую ёмкость в точке касания. Контроллер возбуждает по очереди по одной линии возбуждения слабым переменным током и считывает сигналы одновременно со всех считывающих линий, измеряя тем самым ёмкости в соответствующих точках пересечения. За один цикл сканирования удаётся обнаружить место прикосновения. Путём программной обработки сигналов устраняются шумы и определяются координаты точки.

Эта технология легко обнаруживает несколько одновременных касаний в разных точках, что позволяет расширить возможности ввода. Например, если два пальца движутся по направлению друг к другу, это означает команду сжать что-то (уменьшить размер изображения); если два пальца расходятся, это означает команду расширить что-то (увеличить изображение).

Обратим внимание, что ёмкостной сенсорный экран требует прикосновения незащищённым пальцем. Если вы прикасаетесь к нему в перчатках или пытаетесь использовать (непроводящее) перо, то ничего не получится. Это основной недостаток. Преимуществом же является высокая прозрачность и отсутствие необходимости делать осязаемое нажатие (достаточно только прикоснуться).

Другими активно применяемыми технологиями построения сенсорных панелей являются ультразвуковая (на поверхностных акустических волнах) и инфракрасная. Обе технологии основаны на создании на поверхности стекла некоторого акустического или светового поля. Палец пользователя или любой другой предмет поглощает соответственно звук или свет, и расположенные по краям панели приёмники улавливают изменение поля и определяют координаты воздействия. Эти панели не содержат никаких внутренних слоёв и поэтому чрезвычайно прочны и долговечны. Инфракрасные панели используются для больших дисплеев, таких как 42-дюймовые плазменные экраны.

Графический планшет

Графический планшет (graphics tablet) предназначен для ввода рукописной информации (текста и рисунков). Он представляет собой поверхность (рис. 1.10), на которой пользователь с помощью специальной ручки может писать и рисовать так же, как на бумаге, с той разницей, что результат передаётся в компьютер. Графический планшет подобен сенсорной панели, но отличие состоит в том, что он обеспечивает значительно более высокую точность, необходимую, например, для профессионального рисования. Чтобы получить более высокую точность, для построения планшетов используют иные технологии по сравнению с сенсорными панелями.



Рис. 1.10. Графический планшет с ручкой и мышью.

Одна из ведущих технологий (компания Wacom) использует явление электромагнитной индукции и может быть описана следующим образом. Под поверхностью планшета находится сетка из пересекающихся (но не замкнутых) горизонтальных и вертикальных проводников, которые работают как передающие и принимающие витки трансформатора. В корпусе ручки смонтирована пассивная LC-цепь (параллельно соединённые ёмкость и индуктивность), способная накапливать, а затем отдавать электрическую энергию. Планшет вначале генерирует электромагнитный сигнал, который принимается LC-цепью в ручке. Затем проводники планшета переключаются на приём и считывают сигнал, генерируемый ручкой. Максимальный сигнал возникает в проводниках, находящихся точно под пером ручки, что и определяет координаты ввода. Если ручку наклонить, то сигнал будет передаваться также и на соседние проводники, что позволяет получать дополнительно информацию о наклоне пера в целях имитации определённой техники рисования. Специальные приспособления дают дополнительно возможность измерения уровня давления на перо. Заметим, что, в отличие от сенсорных панелей, графический планшет не должен быть прозрачным, что даёт большую свободу в выборе технологий и материалов при его изготовлении.

Графические планшеты незаменимы в компьютерной графике, особенно двумерной. Огромное значение графические планшеты имеют в Китае, Японии и Корее, т.к. используются для ввода в компьютер иероглифов.

Интерактивная доска

Интерактивная доска (interactive whiteboard) представляет дальнейшее развитие идей графических планшетов и сенсорных экранов, но характеризуется значительно большими размерами, т.к. предназначена для демонстрации в больших аудиториях (рис. 1.11).



Рис. 1.11. Интерактивная доска.

Для ввода информации интерактивные доски используют примерно те же технологии, что и сенсорные панели и графические планшеты. Для досок обычно не требуется слишком высокое разрешение. Что касается изображения на доске, то оно (в виду больших размеров доски) создаётся с помощью проектора, который может располагаться перед или за доской. Наиболее дешёвый вариант получается при переднем расположении проектора. В этом случае доска может быть плотно прижата к стене (т.е. не требует дополнительного места) и может быть непрозрачной, что упрощает построение сенсорной системы. Недостаток здесь в том, что пользователь может заслонять часть видеопотока от проектора, создавая тень. Использование так называемых UST-проекторов, способных работать с поверхностью экспозиции под углом 45° , позволяет улучшить ситуацию, т.к. проектор в этом случае располагается достаточно близко к доске и пользователь не попадает в его поток. Доски с задним расположением проектора более дороги: они требуют много места, кроме того, должны быть прозрачными.

Жидкокристаллические дисплеи

Жидкокристаллический дисплей (Liquid Crystal Display, LCD) – это сегодня основной тип компьютерного монитора. В настоящее время промышленно освоены три основные технологии производства LCD. Существует три главных характеристики, которые определяются выбором той или иной технологии – глубина цвета, угол обзора и время отклика. Для осуществления осознанного выбора того или иного типа дисплея полезно понимать принципы этих технологий.

Работа жидкокристаллических дисплеев основана на явлениях поляризации света и изменения угла поляризации молекулами жидкого кристалла под воздействием электрического поля. Поляризация света определяется тем, в какой плоскости происходят колебания напряжённости электрического или магнитного поля, составляющих электромагнитную волну. Если колебания происходят в горизонтальной плоскости по

отношению к направлению движения волны, то говорят о горизонтальной поляризации; если в вертикальной плоскости – о вертикальной поляризации. Неполаризованный свет состоит из волн, каждая из которых ориентирована произвольно. Свет становится поляризованным, когда проходит через специальный фильтр, характеризующийся определённым углом поляризации. Ничего не меняется для человеческого глаза кроме того, что яркость света уменьшается в два раза. Но если после первого поляризующего фильтра поставить второй, то интенсивность света будет зависеть от соотношения их углов поляризации. Если угол поляризации второго фильтра совпадает с углом поляризации первого, то свет через второй фильтр проходит беспрепятственно. Если же угол второго фильтра перпендикулярен первому, то свет полностью поглощается. Промежуточное значение угла будет приводить к уменьшению яркости света.

Общий принцип работы всех жидкокристаллических дисплеев показан на рис. 1.12. Свет от флуоресцентной лампы через сложную систему отражателей попадает на первый поляризующий фильтр, поляризуется и затем проходит через слой молекул жидкого кристалла. Электрическое поле, прилагаемое с помощью прозрачных электродов, изменяет пространственную ориентацию молекул жидкого кристалла, что приводит к изменению угла поляризации проходящего через них света. Далее свет проходит через цветные фильтры (R, G или B) и попадает на второй поляризующий фильтр. В зависимости от угла поляризации, приобретённого после прохождения через жидкие кристаллы, свет полностью или только частично поглощается, создавая различные оттенки цвета на выходе.

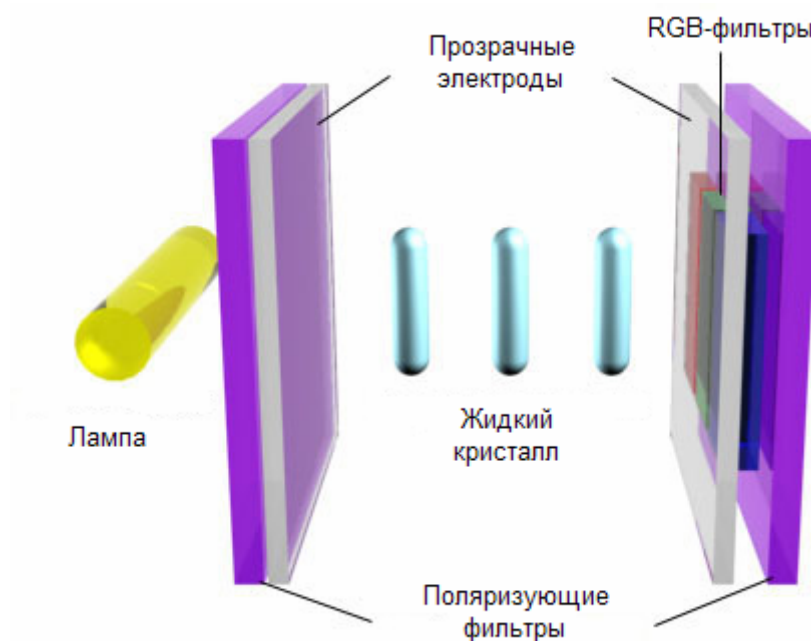


Рис. 1.12. Ячейка жидкокристаллического дисплея.

Поляризация света в жидкокристаллических дисплеях является главной причиной уменьшения угла обзора. Различные технологии построения LCD пытаются улучшить этот показатель. *Угол обзора* – это суммарный угол (влево и вправо или вверх и вниз) между нормалью к плоскости отображения и мнимой линией между глазом человека и жидкокристаллической ячейкой, при котором сохраняется определенный уровень контрастности и цветовой гаммы (измеряется разными производителями по-разному). Угол обзора иногда рассматривается как самый важный параметр LCD, но это далеко не так. Как правило, за монитором работает один человек, который смотрит прямо на монитор, и его угол обзора практически нулевой. Однако следует учесть тот факт, что при увеличении угла

обзора контрастность и цветность начинают постепенно меняться (вплоть до достижения своих крайних значений). Если угол обзора монитора маленький (скажем, 90°), то даже при небольших отклонениях головы будет заметно изменение цветовой палитры. Кроме того, при больших размерах экрана, когда человек смотрит прямо в центр изображения, края экрана наблюдаются под углом и на них могут наблюдаться отклонения в контрастности и цветности. Поэтому экраны с большим углом обзора предпочтительны.

Время отклика определяется инерционностью жидкого кристалла. Для того чтобы ячейка изменила цвет, молекулы кристалла должны изменить свою пространственную ориентацию. На это требуется время. Большое время отклика приводит к тому, что вы видите шлейф при быстром перемещении указателя мыши, а динамичное видео размывается. Приемлемым считается время отклика 25 мс. Некоторые современные модели заявляют о достижении времени отклика 2 мс, что уже очень хорошо.

Третий важный параметр – *глубина цвета*. Для полноцветного изображения (true-color) каждая цветовая составляющая должна иметь 256 различных градаций яркости. Это не всегда возможно в LCD по двум причинам. Во-первых, ячейка, которая в идеале должна быть чёрной, всё же пропускает часть света от лампы, т.е. абсолютно чёрной не является, что снижает контрастность изображения. Во-вторых, чувствительность кристалла к изменению электрического напряжения непостоянна и яркость свечения ячейки зависит от величины напряжения нелинейно. Это приводит к неразличимости некоторых близких уровней яркости. Предлагаются различные схемы компенсации этих недостатков.

Следующим важным понятием является активное управление матрицей жидких кристаллов. В небольших индикаторах каждая жидкокристаллическая ячейка имеет два вывода для подачи управляющего напряжения. В больших панелях такой подход невозможен и все ячейки имеют общие горизонтальные и вертикальные шины управления. В чистом виде такая матрица называется пассивной. Активная матрица в каждой точке пересечения шин управления имеет тонкоплёночный транзистор (thin-film transistor, TFT). При подаче управляющего сигнала на горизонтальную шину все подключенные к ней транзисторы открываются и передают напряжения (фактически, значения яркости соответствующих цветовых составляющих) с вертикальных шин на внутреннюю ёмкость, которую образует сам кристалл. Когда управляющий сигнал снимается, транзистор закрывается и величина напряжения "запоминается" на ёмкости, а молекулы кристалла начинают её "отрабатывать", т.е. соответствующим образом изменяют угол поляризации света. Управляющая схема в это время переходит к следующей строке и т.д. Таким образом все строки дисплея периодически регенерируются. Может показаться, что это то же самое, что и регенерация изображения в электронно-лучевых трубках. Но это не так. Наличие TFT приводит к тому, что каждый пиксель светится постоянно без мерцания.

Иногда жидкокристаллические дисплеи с активной матрицей называют просто TFT-дисплеями. Но это не верно. Как мы увидим, управление с помощью TFT используется и в других типах дисплеев, отличных от LCD.

Охарактеризуем кратко три основные технологии изготовления жидкокристаллических матриц.

Наиболее старая и дешёвая технология называется TN (от английских слов Twisted Nematic – скрученный нитьевидный кристалл). Изначально скрученные кристаллы изменяют угол поляризации поступающего света на 90° , в результате чего свет беспрепятственно проходит через внешний поляризующий фильтр. Мы видим ярко светящуюся точку. Электрическое поле прикладывается перпендикулярно плоскости экрана и заставляет кристалл

распрямляться. Угол поляризации уменьшается и на выходе наблюдается соответствующее уменьшение яркости вплоть до почти чёрного цвета. Такие матрицы наиболее быстродействующие, т.е. имеют наименьшее время отклика, но обладают малым углом обзора. Разновидность технологии TN+Film добавляет на выходе матрицы плёнку с рисками. Риски как бы рассеивают свет и, таким образом, увеличивают угол обзора по горизонтали (но не по вертикали). Во многих источниках считается, что TN-матрицы "честно" передают только 6 бит цветовых составляющих. Для доведения качества цветопередачи до 8 бит могут использоваться различные дополнительные приёмы, такие как FRC (Frame Rate Control). FRC эмулирует оттенки цвета путём быстрого включения и выключения пикселя. Например, если матрица может передавать уровни яркости 60 и 64, а нам нужно получить 63, то можно выдавать 64 в течение трёх последовательных циклов регенерации, затем 60 в течение одного цикла и так постоянно. Человеческий глаз интегрирует яркость по времени и получает значение 63. Заметим, что при выходе TF-транзистора из строя в TN-матрице появляется ярко светящаяся точка.

Другая технология носит название IPS (In-Plane Switching – переключение в плоскости кристалла). Это наиболее дорогая и качественная технология. Её разновидности – S-IPS, H-IPS, FSS и др. Здесь электрическое поле подаётся параллельно плоскости экрана, т.е. вдоль поверхности кристалла. Для этого делаются два электрода по бокам кристалла и для управления ими используются два TF-транзистора. Молекулы кристалла ориентированы так, что их оптическая ось совпадает с направлением поля. Это позволяет резко увеличить угол обзора, но снижает быстродействие. При отсутствии поля угол поляризации не меняется, поэтому внешний поляризующий фильтр полностью поглощает свет, и мы видим чёрную точку. При наличии поля молекулы кристалла поворачиваются, угол поляризации изменяется, и мы наблюдаем увеличение яркости. IPS-матрицы передают оттенки цвета с качеством 8 бит без применения какой-либо эмуляции. При выходе из строя TF-транзистора на экране появляется чёрная точка.

Третья технология – VA (Vertical Alignment – вертикальное выравнивание). Её разновидности – MVA и PVA. Здесь за счёт особой геометрии электродов электрическое поле прикладывается и молекулы жидкого кристалла располагаются под разными углами по отношению к плоскости экрана. При прохождении света молекулы рассеивают его в разные стороны и, таким образом, увеличивается угол обзора. По своим характеристикам данная технология считается средней между TN и IPS. При отсутствии поля ячейка не светится.

Плазменные панели

Плазменная панель (рис. 1.13) представляет собой матрицу газонаполненных ячеек, заключенных между двумя параллельными стеклянными поверхностями. В качестве газовой среды обычно используется смесь гелия, неона и ксенона. Разряд в газе протекает между прозрачным электродом на лицевой стороне экрана и адресными электродами, проходящими по его задней стороне. Газовый разряд вызывает ультрафиолетовое излучение, которое, в свою очередь, инициирует видимое свечение люминофора. Тот же самый принцип используется в флуоресцентных лампах, так что можно рассматривать плазменную панель как множество таких миниатюрных ламп.

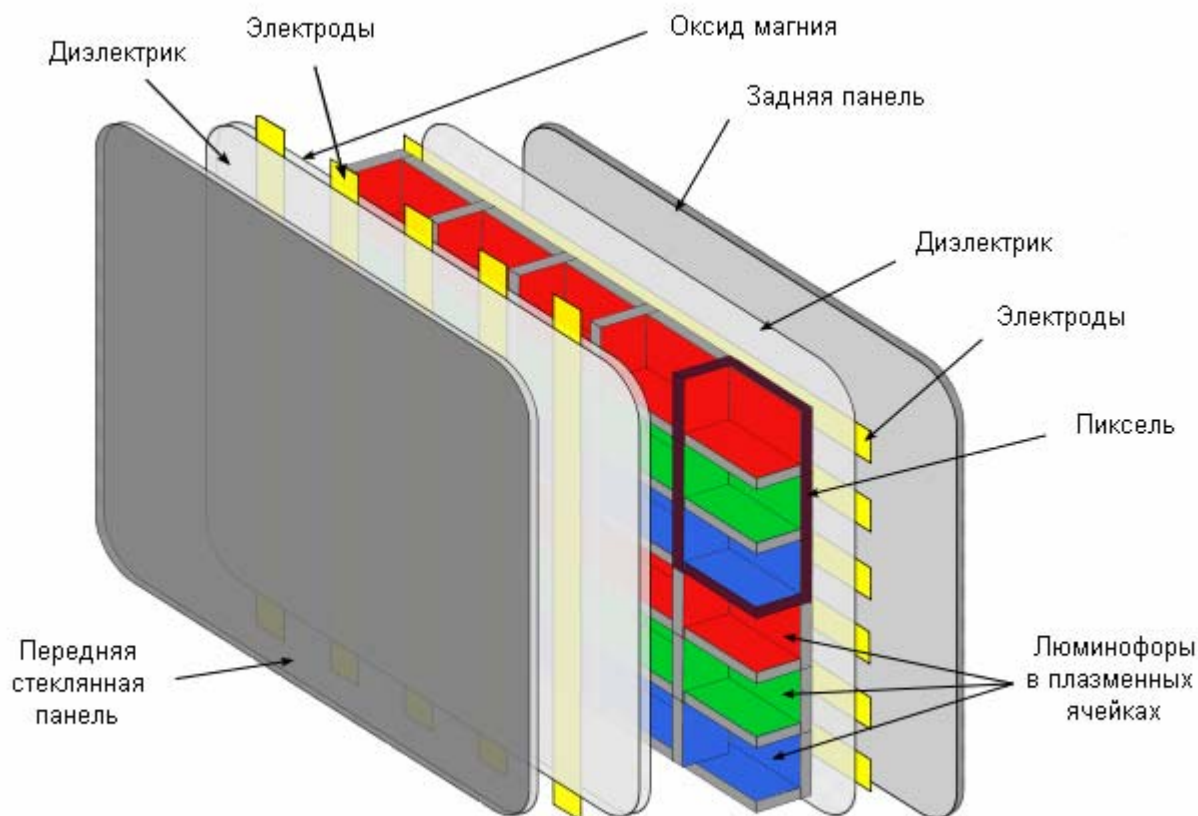


Рис. 1.13. Устройство плазменной панели.

Управляющая электроника плазменной панели зажигает все пиксели по очереди или группами, находящимися на одной линии. Зажигающее напряжение может достигать 300 В. Достаточно быстрая регенерация экрана (типичная частота регенерации – 85 Гц) создаёт статическую картину благодаря инерционности зрения человека. Более того, каждый элемент плазменной панели может светиться только с постоянной яркостью (другими словами, он может гореть или не гореть). Для реализации оттенков цвета применяется широтно-импульсное регулирование времени свечения (это похоже на метод FRC в жидкокристаллических дисплеях).

Главные положительные черты плазменной технологии дисплеев – высокое качество передачи цвета, высокая контрастность (если пиксель не светится, то он абсолютно чёрный), высокое быстродействие (быстрота смены кадров определяется только частотой регенерации экрана, инерционность отдельных ячеек практически отсутствует), большой угол обзора (практически 180° как по горизонтали, так и по вертикали), большие размеры экрана (диагональ до 1.5 м при толщине панели 8 см).

Но у этой технологии есть и недостатки. Обычно среди них называют высокое (по сравнению с жидкокристаллическими дисплеями) энергопотребление, выгорание люминофора (в связи с чем качество воспроизведения цветов снижается с течением времени) и сравнительно большой размер пикселя (его просто технологически трудно сделать маленьким). Но для некоторых людей главный недостаток – мерцание экрана (вызванное его постоянной регенерацией и широтно-импульсным регулированием яркости отдельных пикселей). Даже если оно не различимо явно в силу инерционности зрения, воздействие мерцания на рецепторы глаза приводит к их утомлению, кроме того, мерцание хорошо

ощутимо боковым зрением, в связи с чем плазменные панели очень некомфортно смотреть с близкого расстояния. Заметим, что в жидкокристаллических дисплеях ТF-транзистор "держит" пиксель на постоянном уровне яркости до следующего её изменения, поэтому мерцание отсутствует.

OLED-дисплеи

OLED-дисплеи (Organic Light Emitting Diode – органический светодиод) – наиболее новый и перспективный класс дисплеев. Пиксели в этих дисплеях строятся из разноцветных светодиодов. Устройство одного органического светодиода показано на рис. 1.14.

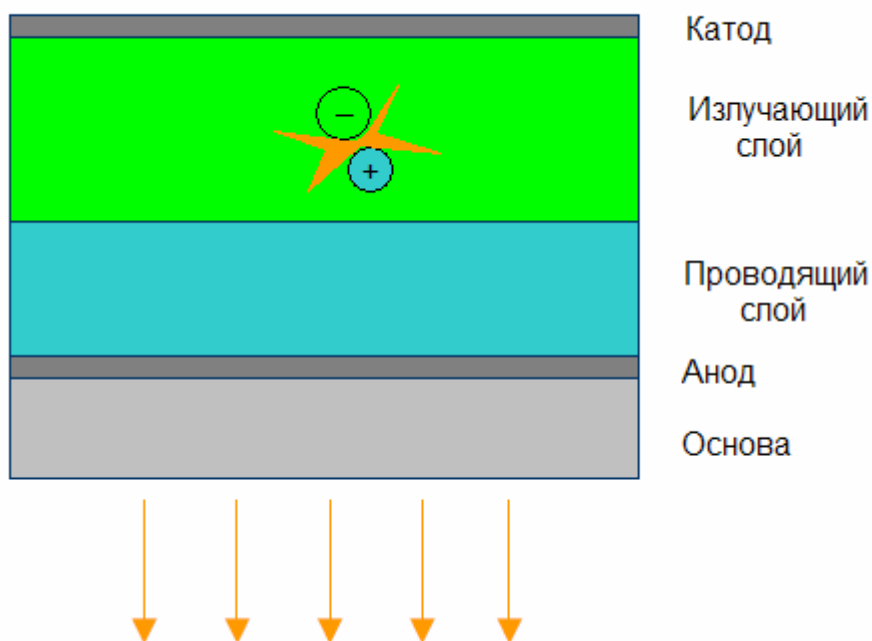


Рис. 1.14. Устройство органического светодиода.

Катод обычно непрозрачный и делается из бария, кальция или алюминия – металлов, легко отдающих электроны в излучающий слой. Излучающий и проводящий слои выполняются из различных полимеров (отсюда и название этого типа диодов – органические) с определенными присадками, обеспечивающими электронную или дырочную проводимость (по аналогии с кристаллическими полупроводниками) и нужный цвет свечения. Анод изготавливается из прозрачного ИТО. Когда к аноду прикладывается положительное напряжение, через прибор начинают течь электроны. Таким образом, катод снабжает электронами излучающий слой, а анод забирает электроны из проводящего слоя, вызывая появление дырок в излучающем слое. При попадании электрона в дырку излучается свет. Интенсивность свечения определяется величиной протекающего тока, который, в свою очередь, зависит от приложенного напряжения. При нулевом и отрицательном напряжении на аноде диод не светится, что позволяет получать абсолютно чёрные пиксели. Органические светодиоды обладают очень малой инерционностью (в сотни раз меньшей, чем у жидких кристаллов) и обеспечивают практически полный угол обзора во всех направлениях.

Отдельные светодиоды собираются в матрицы с перекрещивающимися катодами и анодами абсолютно по тому же принципу, что в жидкокристаллических и плазменных панелях. В

OLED-дисплеях с высоким разрешением, также как в LCD, используется активное управление ячейками с помощью TF-транзисторов. Однако, для каждой ячейки нужно два таких транзистора. Один транзистор открывается при адресации ячейки и передаёт управляющее напряжение на затвор второго. Когда первый транзистор закрывается, переданное напряжение "запоминается" на ёмкости затвора второго транзистора и определяет величину тока, которым этот второй транзистор питает светодиод. В результате светодиод горит с определённой яркостью без мерцания до следующего цикла регенерации, когда яркость может быть изменена.

Один из наиболее технологичных процессов изготовления OLED-панелей состоит в их "печатании" с использованием техники струйной печати: нужные материалы просто последовательно наносятся на основу с помощью струйных принтеров, как показано на рис. 1.15. Это позволяет в перспективе сделать OLED-дисплеи очень дешёвыми. Материалом основания может быть стекло или прозрачная гибкая плёнка. В результате появляется возможность делать гибкие, удароустойчивые устройства. Суммарная толщина слоёв, показанных на рис. OLED, не превышает 1 мкм. Таким образом, OLED-дисплеи могут быть чрезвычайно тонкими.

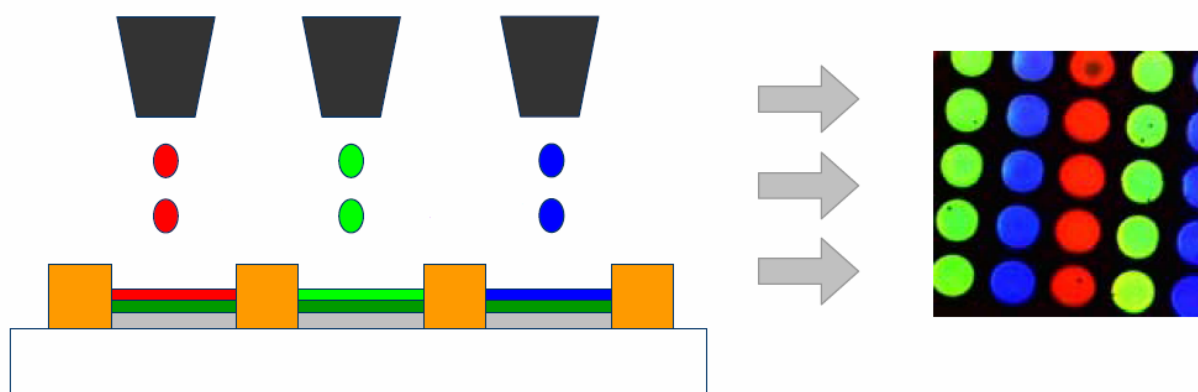


Рис. 1.15. Печатание матрицы OLED.

К достоинствам OLED-дисплеев относятся: высокое качество передачи цвета, большой угол обзора, высокое быстродействие (по этим параметрам они значительно превосходят LCD и приближаются к плазменным панелям), низкое энергопотребление, малая толщина (по этим параметрам они превосходят LCD и плазму). Главной проблемой OLED-дисплеев сегодня является сравнительно невысокий рабочий ресурс органических материалов, из которых они сделаны. Особенно это актуально для синих светодиодов. Время жизни OLED-дисплеев в четыре раза меньше, чем у плазменных панелей и, тем более, LCD. Однако имеются предпосылки решения этой проблемы с помощью новых материалов уже в ближайшее время.

Глава 2

Проблемно-центрированная разработка интерфейса

Одним из наиболее эффективных подходов к разработке интерфейса с пользователем, предлагаемых в литературе по человеко-машинному взаимодействию, является подход, сфокусированный на задачах, которые нужно решать пользователю – проблемно-центрированный подход. Он противопоставляется так называемому "методу водопада", который ещё широко применяется и будет кратко охарактеризован в конце главы. При проблемно-центрированном подходе процесс разработки структурируется исходя из специфических задач, которые пользователь должен будет решать с помощью разрабатываемой системы. Эти задачи выбираются на ранней стадии разработки, затем они используются для выявления требований к дизайну, чтобы облегчить выработку решений и их оценку по мере развития проекта. Вот основные этапы проблемно-центрированного дизайна:

- анализ задач и пользователей;
- выбор репрезентативных задач;
- заимствование;
- черновое описание дизайна;
- обдумывание дизайна;
- создание макета или прототипа;
- тестирование дизайна с пользователями;
- итерирование;
- реализация;
- отслеживание эксплуатации;
- изменение дизайна.

Теперь рассмотрим предлагаемые этапы более подробно.

2.1. Анализ задач и пользователей

На этом этапе предстоит дать ответ на вопрос, кто и зачем собирается использовать разрабатываемую систему. Необходимость в анализе задач очевидна: построение большой системы, которая не делает того, что нужно, вряд ли можно назвать успехом дизайна. Но поверх простого требования "делать то, что нужно" хорошая система должна гладко входить в существующий мир пользователя и его работу. В частности, система должна запрашивать от пользователя информацию в порядке, который кажется ему естественным, она должна давать возможность простой коррекции ошибок при вводе данных, выбранные для организации интерфейса аппаратные средства должны вписываться в рабочую среду пользователя и быть эргономичными для его действий. Рассмотрение этих и множества других деталей интерфейса часто отсутствует при традиционной разработке, но они могут быть раскрыты, если дизайнер уделит время деталям задач, которые в действительности должен решить пользователь.

Важно также понимание пользователей как таковых. Осведомлённость об уровне знаний пользователя позволяет дизайнеру ответить на вопросы о выборе названий пунктов меню, о

том, какие материалы включить в обучающий комплект и контекстную помощь, и даже о том, какие возможности должна обеспечивать система. Если, например, система разрабатывается для пользователя персонального компьютера, привыкшего к приёму "копировать–вставить", то целесообразно, чтобы в новой системе этот приём также работал, даже если он не слишком важен с точки зрения основной задачи. Такие трудно оцениваемые различия среди пользователей, как уровень их квалификации, интерес к изучению новых систем, заинтересованность в успехе разработанной системы, могут обуславливать многие решения дизайнера, например, какой заложить уровень обратной связи с пользователем, где предпочесть команды, подаваемые с помощью клавиатуры, а где – выбираемые из меню и т.д.

Эффективный анализ задач и пользователей требует тесного персонального контакта между командой разработчиков и теми людьми, которые в дальнейшем будут пользоваться системой. Это не всегда просто осуществить. Часто руководители коллективов (как разработчиков, так и пользователей) становятся препятствием. Например, от разработчиков могут требовать представление существенных деталей проекта перед тем, как он будет показан заказчику. Однако несомненно, что ранний и непрерывный контакт между разработчиками и пользователями существует для успеха разработки.

2.2. Выбор репрезентативных задач

После выработки хорошего понимания пользователей и их задач более традиционный процесс разработки может абстрагироваться от этих фактов и привести к общей спецификации системы и её пользовательского интерфейса. Проблемно-центрированный дизайн предлагает более жёсткий подход. Разработчик должен выделить несколько репрезентативных задач, которые будут решаться при использовании системы. Это должны быть задачи, которые пользователи описали разработчикам. Первоначально они могут быть описаны буквально в нескольких словах, но, т.к. речь идёт о реальных задачах, в любой момент в дальнейшем эти описания могут быть расширены до любой степени детальности, чтобы ответить на любые вопросы, касающиеся дизайна интерфейса или анализа предложенных решений. Вот несколько примеров:

- для программы обработки текстов: "извлечь запись мемо и поместить её в список рассылки";
- для электронной таблицы: "рассчитать бюджет заработной платы на следующий год";
- для коммуникационной программы: "войти на сервер организации через модем";
- для промышленной системы управления: "передать управление следующей смене".

Отметим ещё раз, что это должны быть реальные задачи, с которыми сталкиваются пользователи. Разработчики должны собрать все необходимые материалы для выполнения этих задач: копию файла с мемо-записью, информацию о заработных платах в текущем году и факторы, вызывающие их изменения, и т.д.

Отобранные задачи должны достаточно полно покрывать всю функциональность системы, и дизайнеру полезно сделать проверочный список всех функций и путём сопоставления его с перечнем задач удостовериться, что желаемое покрытие достигнуто. Это также должна быть смесь простых и более сложных задач. Простые задачи, например, "проверить правильность написания слова", полезны на ранних стадиях проектирования, но многие проблемы построения интерфейса будут выявлены только через сложные задачи, отражающие взаимодействия в реальном мире. Способность разработчика произвести эффективный набор задач – это настоящий тест на его понимание пользователей и их работы.

2.3. Заимствование

На данном этапе очень полезно найти существующие интерфейсы, с помощью которых пользователи могут выполнить требуемую работу, и затем строить идеи новой системы на их базе насколько это законно и возможно практически. Этот сорт копирования может быть эффективным как для высокоуровневых парадигм взаимодействия, так и для низкоуровневых решений на основе специальных органов управления и дисплеев.

На высоких уровнях думайте о репрезентативных задачах и пользователях, которые их решают. Какие программы эти пользователи или люди, находящиеся в подобных ситуациях, используют сейчас? Если они используют электронные таблицы, то, может быть, ваш дизайн должен выглядеть как электронная таблица. Если они используют объектно-ориентированные графические пакеты, может быть, ваше приложение должно быть похоже на них. Вы можете быть способны создать новую парадигму взаимодействия, которая лучше подойдёт к разрабатываемому приложению, но риск неудачного дизайна довольно велик. Существующую парадигму реализовать быстрее и проще, т.к. многие проектные решения уже оказываются выполненными. Но что более важно, её легче будет изучить и удобнее применять для пользователей, т.к. они уже заранее будут знать, как работает большая часть интерфейса.

Копирование известных решений также полезно для низкоуровневых деталей интерфейса, таких как размещение кнопок и названия полей меню. Приведём пример. Пусть вы разрабатываете специализированный пакет управления формами, и его спецификация требует, чтобы в какой-то момент могла быть выполнена проверка правописания. Тогда вы должны посмотреть на элементы управления в подсистемах проверки правописания в текстовых процессорах, которые в данный момент используют будущие пользователи вашей системы. Будет чрезвычайно хорошо, если в вашей системе данный интерфейс будет построен таким же образом.

На пути заимствования многие дизайнеры допускают неверные решения по причине недостаточно глубокого видения деталей существующих реализаций поверх задач собственной разработки. Углубимся немного в пример с проверкой правописания. Допустим, ваш анализ показал, что программа проверки правописания обычно находит неправильно написанные слова, в частности, имена и фамилии, и может автоматически скорректировать их написание, используя базу данных пользователя. Вы решаете, что наилучшим вариантом взаимодействия будет высветить на дисплее предлагаемый вариант корректировки и позволить пользователю принять его путём нажатия на клавишу <Enter>. Но текстовый процессор, которые будущие пользователи вашей системы используют наиболее часто, следует другому соглашению: нажатие клавиши <Enter> сохраняет "неправильное" написание слова. Стоит ли следовать этому существующему правилу или реализовать своё, более эффективное? В известной степени решение зависит от того, будут ли пользователи чаще пользоваться вашей системой или текстовым процессором. Но чаще всего наилучшим вариантом будет придерживаться правила старой системы. Это позволяет пользователям применять наработанные навыки, хотя и требует при работе на одно или два нажатия клавиши больше, чем новый "оптимизированный" вариант.

2.4. Черновое описание дизайна

Черновое (грубое) описание разрабатываемой вами системы должно быть положено на бумагу (обязательно). Это позволяет задуматься о многих вещах. Но это описание не следует оформлять в виде компьютерной программы (пока), даже если вы умеете пользоваться какими-либо системами автоматизации разработки. Такие системы вынуждают вас прикрепляться к конкретным решениям, которые ещё слишком рано делать.

На этой стадии команда разработчиков может проводить множество дискуссий по поводу того, какие возможности должна включать в себя система и как они должны представляться пользователям. Дискуссия должна следовать проблемно-центрированному подходу. Если кто-то из команды предлагает введение новой возможности, то другой член команды обязан спросить его, решению какой из репрезентативных задач эта новая возможность будет способствовать. Возможности, которые не способствуют решению любой из задач, как правило, должны отбрасываться. Либо же список репрезентативных задач должен быть дополнен реальной задачей, которая требует этой возможности программы.

Репрезентативные задачи должны использоваться как контрольный список, позволяющий удостовериться, что описание системы полно. Если вы не можете представить решение каждой задачи внутри текущего определения системы, это определение должно быть доработано и улучшено.

2.5. Обдумывание дизайна

Никакая авиационная компания не будет разрабатывать и строить реактивный самолёт без предварительного инженерного анализа, который предсказывает основные технические характеристики. Стоимость строительства и риск неудачи слишком велики. Точно так же стоимость построения законченного пользовательского интерфейса и его тестирование с достаточным количеством пользователей для выявления всех проблем неприемлемо высока. Хотя разработка интерфейсов ещё не достигла такого уровня сложности, как авиационная инженерия, существует несколько структурных подходов, которые можно использовать, чтобы исследовать сильные и слабые стороны интерфейса до его программного воплощения.

Один из методов состоит в подсчёте количества нажатий клавиш, движений мыши и мыслительных операций (решений), необходимых для выполнения задач, предписанных разрабатываемой системе. Это позволяет оценить трудоёмкость выполнения задач по времени и выявить задачи, требующие слишком много шагов. Процедуры для реализации этого подхода, названные анализом GOMS, разработаны достаточно детально. Основные идеи GOMS будут рассмотрены позднее.

Другой метод основан на приёме, названном познавательный сквозной контроль (cognitive walkthrough, CWT), и позволяет находить места в дизайне, где пользователь может делать ошибки. Как и GOMS, CWT анализирует взаимодействия пользователей с интерфейсом при решении отдельных задач. Данный метод также будет представлен позже.

2.6. Создание макета или прототипа

После этапа обдумывания дизайна по его описанию на бумаге, приходит пора построить что-то более конкретное, могущее быть показанным пользователям и представляющее дальнейшую детализацию предстоящей работы. При разработке простых систем этот более конкретный продукт может быть просто серией рисунков на бумаге, показывающих вид интерфейса по мере того, как пользователь проходит шаги решения репрезентативных задач. Удивительно много информации можно получить, просто показав макет на бумаге нескольким пользователям. Макет даже может выявить скрытое непонимание между членами команды разработчиков.

Для более тщательного анализа и для более сложных систем могут использоваться специальные инструменты для создания прототипов. Можно даже построить прототип с помощью так называемых UIMS-систем (от англ. User Interface Management System – система управления интерфейсом пользователя), которые дадут фундамент для окончательного продукта. Этот подход может быть особенно продуктивным не только потому, что он уменьшает объём работы, требующейся для построения финального продукта, но и потому, что интерфейсные приёмы, протестированные на базе самостоятельного макета, могут оказаться трудными в конечной реализации. Рассмотрение профессиональных UIMS-систем, такие как Hypercard или ProcSee, выходит за рамки нашего курса. Для учебных целей в качестве UIMS-системы может рассматриваться визуальная среда программирования, создаваемая такими инструментами, как Borland Delphi, C Builder или MS Visual Studio. Хотя они и не считаются полноценными UIMS-системами, но в какой-то степени решают похожие задачи – отделение графической части интерфейса от логики функционирования системы.

На данном этапе не нужно реализовывать систему целиком. Усилия должны быть сосредоточены на частях её интерфейса, нужных для решения репрезентативных задач. Скрываемая за интерфейсом функциональность системы, которая может находиться ещё в стадии разработки, может эмулироваться методами "Волшебника из страны Оз". То есть сам разработчик или его коллега могут выполнять действия, которые система пока не может делать. Разумеется, не следует вводить в заблуждение пользователей, особенно их руководителей, чтобы они думали, что система уже завершена.

2.7. Тестирование дизайна с пользователями

Опыт показывает, что независимо от того, как тщательно был сделан анализ дизайна на предыдущих этапах, существуют проблемы, которые выявляются только при тестировании дизайна с пользователями. Тестирование должно проводиться с людьми, чей уровень образования и специальной подготовки примерно соответствует тому, что будет у реальных пользователей системы. Следует попросить пользователей выполнить одну или несколько репрезентативных задач, решение которых поддерживает система. Здесь оказывается эффективным приём "думанье вслух". Вы просите пользователя не только делать необходимые действия для решения поставленной задачи, но и говорить, что он делает и о чём размышляет. Можно полностью записывать его комментарии (например, на диктофон) или только делать заметки. Эти комментарии дают неоценимые данные для улучшения дизайна системы.

Информация, собранная при тестировании системы с пользователями, даёт ответ на многие вопросы: сколько времени потребовало выполнение тех или иных действий и задач в целом,

какие допускались ошибки, что вызвало затруднение или удивление у пользователя, даже если это и не привело к ошибке. Записанные комментарии пользователя позволяют понять, почему были допущены ошибки. Без комментариев вы только фиксируете сам факт ошибки, но вынуждены потом догадываться (додумывать за пользователя), почему это произошло. При анализе действий пользователя и его комментариев может выясниться, что он мыслит не так, как дизайнер системы. Это позволит внести корректировки, позволяющие приблизить интерфейс системы к её предполагаемому пользователю.

2.8. Итерирование

Тестирование с пользователями всегда показывает какие-то проблемы с дизайном. Помните, что цель тестирования состоит не в том, чтобы доказать правильность интерфейса, а в том, чтобы улучшить его. Необходимо проанализировать результаты тестирования, соизмеряя стоимость корректировок с серьёзностью возникших проблем, затем доработать интерфейс и протестировать его снова. Серьёзные проблемы могут даже потребовать пересмотра понимания задач и пользователей, т.е. откатить вас на первый этап проблемно-центрированного подхода.

Одна вещь, о которой необходимо помнить на каждой итерации – различные возможности и особенности интерфейса не самостоятельны. Например, переделывание меню для устранения проблемы, возникшей при выполнении одной задачи, может создать проблемы для других задач. Некоторые из таких взаимовлияний могут быть найдены при анализе без пользователей с помощью SWT или аналогичных приёмов. Другие же не выявятся без тестирования с пользователями.

Когда следует прекратить итерации? Если были установлены специфические требования практичности (UO, см. "Управление процессом разработки" далее), то итерации прекращаются, как только эти требования выполнены. В противном случае прекращение итераций – это управленческое решение, принимаемое на основе баланса стоимости и полезности дальнейших улучшений против необходимости выхода на рынок с законченным продуктом или сроков предоставления его пользователям.

2.9. Построение интерфейса системы

Ключевой руководящий принцип при построении (программной реализации) интерфейса состоит в обеспечении возможности его дальнейшего изменения. Постарайтесь предусмотреть некоторую настройку с помощью небольших изменений значений констант и переменных. Например, если вы пишете собственную подпрограмму для реализации специализированного меню, то не закладывайте жёстко в программу такие параметры, как размер, цвет, количество элементов. Постарайтесь также предусмотреть возможность небольших изменений кода, используя ясное разделение на модули. Если доработки в будущем потребуют замены специализированного меню какой-либо более общей функциональной возможностью, доработки кода должны быть тривиальны. Всё это звучит как обычные требования к хорошему стилю программирования и в действительности ими и является. Но это особенно важно для пользовательского интерфейса, который часто занимает более половины кода коммерческого продукта.

2.10. Отслеживание дизайна системы

Фундаментальный принцип рассматриваемого подхода состоит в том, что команда разработчиков не должна быть изолирована от всей остальной деятельности, связанной с функционированием системы. Если этот принцип уважается, то разработчик должен иметь контакт с пользователями не только в процессе разработки, но и после выпуска системы. Это ключевой момент для всех серьезных организаций, заинтересованных в своём постоянном присутствии на рынке.

Один из методов введения разработчиков в контакт с пользователями – организация поочерёдных дежурств на горячей линии связи с потребителями. Другая важная вещь для больших систем – организация собраний групп пользователей и конференций. Такая работа также важна для менеджеров, т.к. позволяет им лучше понять реакцию пользователей на продукт, который они продают.

Кроме помощи в ответе на очевидный вопрос: делает ли система то, для чего была разработана, взаимодействие с пользователями часто рождает новые идеи об использовании продукта, открывая для него новые возможности на рынке. Эта информация может служить обратной связью для процесса разработки, позволяющей улучшить описание задач для новой версии продукта и углубить понимание разработчиком предметной области.

2.11. Изменение дизайна

На существующем сегодня рынке компьютеров и программ вряд ли существуют продукты, которые поддерживают свою жизнеспособность без регулярных усовершенствований (апгрейдов). Независимо от того, насколько удачно система была спроектирована первоначально, с большой вероятностью она будет терять адекватность с течением лет. Меняются и задачи и пользователи. Приёмы работы меняются из-за нового оборудования и программных продуктов. Пользователи приобретают новые навыки и ожидаемые реакции. Разработчики должны стоять вровень с этими изменениями, не только отслеживая состояние той рабочей среды, для которой была предназначена их система, но и развитие всего общества, технологий и методов, потребностей. Следующая версия системы должна не только устранять обнаруженные ошибки и недостатки, но и давать новые возможности пользователям.

2.12. Управление процессом разработки

Проблемно-центрированный подход против метода "водопада"

Традиционный метод "водопада" (waterfall model) начинает разработку с этапа анализа требований, который выполняется системными аналитиками, обычно не являющимися дизайнерами интерфейсов. Затем требования преобразуются в спецификации системы. Далее аппаратура, функциональная часть программного обеспечения и его интерфейсная часть разрабатываются в соответствии с принятыми спецификациями.

На практике оказалось, что метод водопада – плохой подход, если программное обеспечение содержит интерфейс с пользователем как важную компоненту. Как уже говорилось, успешный разработчик интерфейса нуждается в глубоком понимании задач пользователя и

того, как эти задачи вписаны в остальную работу пользователя. Такое понимание не может возникнуть из рассмотрения формальных спецификаций. Более того, опыт показывает, что для создания эффективного интерфейса существенны несколько итераций разработки. Традиционный метод водопада просто не допускает таких итераций.

Команда разработчиков

Так как методология проблемно-центрированного дизайна распространяет процесс разработки интерфейса на весь цикл разработки и жизненный цикл программного обеспечения, интерфейс не может быть сделан и проанализирован группой узких специалистов в одной точке всего процесса. Задача разработки хорошего интерфейса должна решаться командой, которая разрабатывает весь продукт в целом.

Команда разработчиков должна состоять из людей с разнообразными способностями и несколькими общими чертами. Они должны знать интересы пользователей, иметь опыт работы как с хорошими, так и с плохими интерфейсами, и, настроенные оптимистически, они должны быть привержены идее создания эффективной системы. Команда должна включать представителей всего диапазона интерфейсно-ориентированных специальностей: программистов, технических писателей, разработчиков учебных пакетов, специалистов по маркетингу. Команда может включать аналитика в области интерфейсов, но это не столь существенно. Разделяемая всеми приверженность качеству интерфейса, дополненная возможностями контактов с реальными пользователями, почти обязательно приведёт к хорошим результатам.

Ответственность

Ответственность за весь интерфейс должна быть централизована. В частности, разработчики, создающие программы, не должны от них отписываться и сдавать их совершенно отдельной группе, которая пишет руководства, которые затем сдаются другой группе, разрабатывающей обучающие пакеты. Вся эта деятельность должна координироваться, что достигается только посредством централизованного управления.

Требования практичности

Практичность (англоязычный термин – usability) – это одна из важнейших характеристик систем, изучению которой посвящено множество исследований. Управленческая деятельность в серьёзной корпорации может потребовать от вас представить различные показатели, которые количественно измеряют практичность. Требования практичности – это целевые значения для таких характеристик, как скорость выполнения репрезентативных задач и допустимое количество ошибок. Эти показатели могут использоваться, чтобы мотивировать разработчиков и обосновывать решения по распределению ресурсов. Целевые значения могут быть выбраны так, чтобы побить конкурентов или обеспечить функциональные нужды для хорошо определённых задач. Например, в целях успешной конкуренции от интерфейса может потребоваться не только полнота и удобство для пользователя, но и достижение результатов типа сокращения среднего времени выполнения задач пользователя на 20 % по сравнению с известными аналогами. Типичным примером специальной разработки в области интерфейса, значительно улучшающим скорость работы, является алгоритм T9 для набора текстов на телефонной клавиатуре.

Глава 3

CWT-анализ интерфейса

3.1. Описание технологии

Название этого метода происходит от английских слов Cognitive Walkthrough (познавательный сквозной контроль). CWT – это формализованный способ представления мыслей и действий людей, когда они пользуются интерфейсом в первый раз. Дадим описание основной идеи метода, затем поясним её на примере и разберем следствия и возможные ошибки.

Всё начинается с того, что у вас есть прототип или детальное описание интерфейса, и вы знаете, кто будет пользователем системы. Вы выбираете одну из задач, решение которых интерфейс должен поддерживать. Вы формируете полный письменный список действий, необходимых для выполнения задачи при помощи интерфейса. Затем вы пытаетесь рассказать правдоподобную историю о каждом действии, которое должен выполнить пользователь. Чтобы история была правдоподобной, необходимо давать мотивацию каждого действия пользователя исходя из его предполагаемых знаний и подсказок и реакций интерфейса. Для каждого действия могут обнаруживаться проблемы, мешающие правильному его выполнению. Эти проблемы записываются, но затем вы предполагаете, что они исправлены, и переходите к следующему действию. В результате у вас остаётся список проблем, что является руководством к действию по исправлению (улучшению) интерфейса.

Если вы попытаетесь применить CWT-анализ к какой-нибудь современной программе, выпускаемой солидной фирмой, то, скорее всего, вам придётся поставить отметку ОК по отношению к каждому действию. Дело в том, что крупные компании разрабатывают интерфейсы своих программ очень тщательно, используя рассматриваемые в нашем курсе методы и многие другие. Неудивительно, что их программы обычно успешно проходят CWT-анализ, и, взяв их в качестве примера, мы не увидим никаких особенностей этого анализа в смысле выявления проблем. Мы рассмотрим пример CWT-анализа на малоизвестной программе CDCору, разработанной Маркусом Бартом в 1997 году.

Программа CDCору предназначена для пользователей, владеющих лишь общими приёмами работы с компьютером в ОС Windows. Она не русифицирована и поэтому неявно предполагает, что пользователь знаком с английским языком. Именно такую модель пользователя мы и будем использовать при анализе. Одна из задач, которую может решать программа – это сохранение треков с аудио диска в формате MP3. Внешний вид интерфейса программы представлен на рис. 3.1.

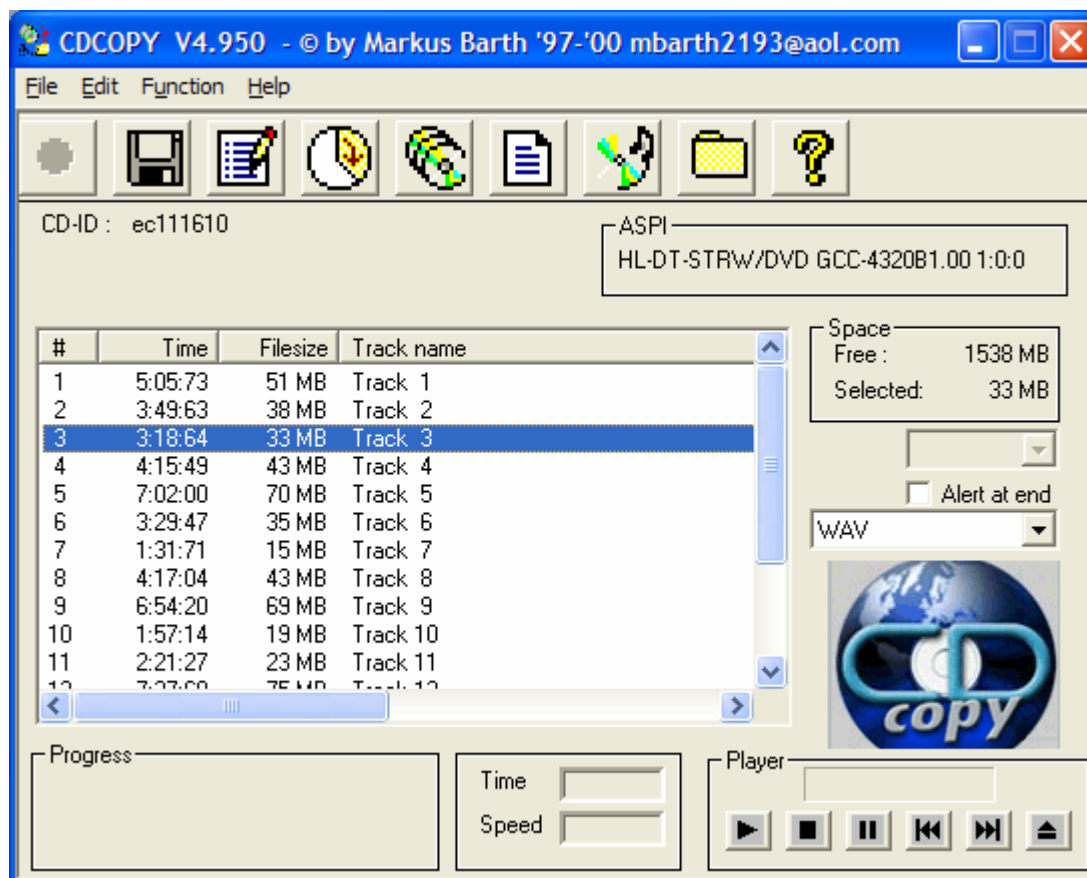


Рис. 3.1. Главное окно программы CDCopy.

Более точно наша репрезентативная задача формулируется так: преобразовать третий трек аудио компакт-диска в формат MP3 и сохранить на жёстком диске компьютера. Последовательность действий для выполнения данной задачи выглядит следующим образом: (1) загрузить аудио диск в устройство чтения компакт-дисков компьютера; (2) запустить программу CDCopy; (3) в появившемся списке треков выбрать трек № 3; (4) в списке форматов файлов выбрать MP3(MPEG 1 Lay. 3); (5) нажать кнопку "Start copying".

Решение задачи выглядит довольно просто: оно требует от пользователя всего пять действий. Однако выполним их подробный анализ. Первые два действия вроде бы не связаны с интерфейсом программы и должны обеспечиваться интерфейсами других систем, не подвергающихся анализу. Но всегда ли для пользователя естественна такая последовательность действий? Многие пользователи предпочитают сначала запускать нужную им программу, а уже затем вставлять диск (в противном случае может сработать автоопределение и запуститься совсем другое приложение). Высказанное замечание ещё не означает, что в интерфейс программы нужно вносить изменения: может быть, программа будет корректно работать и при другой последовательности действий. Но значение метода SWT в том, что любое замечание является поводом дальнейших проверок. И действительно, при дальнейших исследованиях оказывается, что если вначале выполнить действие 2, то программа выдаёт окно с надписью "No media present" и кнопкой "OK"; при закрытии окна программа запускается с пустым списком треков. Это может сбить с толку неопытного пользователя и заставить его закрыть программу и попытаться начать всё с начала в другой последовательности. Анализ первых двух действий сразу даёт подсказку к улучшению интерфейса: если программа запускается до загрузки диска, то не нужно выдавать предупреждающее сообщение, а после запуска программы с пустым списком треков выдать пользователю явный запрос на загрузку диска. Итак, мы выявили некоторые слабости

интерфейса в первых двух действиях, но теперь предположим, что они устранены, и перейдём к третьему действию.

Третье действие заключается в выборе нужного трека в списке. Здесь интерфейс стандартен и однозначен. Никаких возможностей для непонимания пользователем, как правильно выполнить это действие, мы не находим. Замечаний нет.

Четвёртым действием пользователь должен выбрать формат MP3 в списке форматов. Сразу возникает вопрос: как он найдёт этот список? На самом деле имеется в виду выпадающий список, в котором сейчас выводится слово WAV. Но пользователь может не знать, что такое WAV. Ведь его задача состоит в сохранении аудио трека в формате MP3. Он может вообще не знать других слов, кроме MP3. Если он знает, что WAV – это один из форматов, то он может догадаться, что элемент управления с этим словом и есть список форматов. Очевидное решение проблемы – снабдить список форматов меткой с текстом типа "Output format". Считаем, что эта проблема исправлена, и переходим к пятому действию.

Пятое действие. Опять возникает проблема с нахождением кнопки "Start copying". Может возникнуть желание нажать на большую кнопку с похожей надписью "CD copy", но она выполняет совсем другое действие. На самом деле имеется в виду кнопка в верхнем ряду с изображением дискеты. Всплывающая надпись "Start copying" появляется при наведении на неё указателя мыши. Можно улучшить ситуацию, если при описании действий пользователя использовать картинку с изображением кнопки. Но всё равно, многие привыкли к тому, что кнопка с изображением дискеты сохраняет файл без преобразования формата. Лучше было бы изменить вид кнопки. После нажатия на кнопку начинается процесс копирования, его ход отображается в новом окне. По завершении процесса программа без всяких вопросов переходит в исходное состояние. Но у пользователя возникает вопрос: если преобразованный файл был сохранён, то где? Здесь мы сталкиваемся с типичным недостатком интерфейса – отсутствием необходимой обратной связи. Пользователь не видит результата своего действия (в предыдущих действиях результат был ясно виден, т.е. обратная связь была вполне нормальной). Предпринятые изыскания показали, что файл был сохранён по месту установки программы (\Program Files\CDCopy) в папке es111610. Если присмотреться, то видно, что название папки взято из CD-ID, отображаемого над списком треков (см. рис. p375). Отметим, что если бы программа была запущена не администратором системы, то копирование не смогло бы выполниться, т.к. папка Program Files и все её внутренние файлы недоступны для записи другим пользователям. Программа не работает в многопользовательском режиме. Простая доработка интерфейса устраняет проблемы: запросить у пользователя путь для сохранения файла с помощью стандартного диалогового окна (как это делают многие программы); начальный путь брать из переменной среды USERPROFILE (в ней записан путь к каталогу текущего пользователя).

Итак, мы провели CWT-анализ интерфейса программы CDCopy на примере решения одной из репрезентативных задач. Был выявлен ряд недостатков интерфейса и предложено 4 простых доработки, их устраняющих.

3.2. Что даёт CWT-анализ

Из рассмотренного примера можно сделать вывод, что CWT-анализ позволяет обнаружить несколько типов проблем с интерфейсом.

1. Он может поставить под сомнение ваши первоначальные и не вполне обоснованные предположения о том, как мыслит пользователь (вначале поставить диск, а потом запустить программу, или наоборот? кнопка с изображением дискеты означает копирование или сохранение?).
2. Он может выявлять элементы управления, которые очевидны для разработчика, но могут быть скрыты от пользователя (список форматов выходного файла).
3. Он может выявлять затруднения с надписями и подсказками (неудачное предупреждение "No media present").
4. Он может обнаруживать неадекватную обратную связь, что может заставить пользователя сомневаться в результате и повторять всё с начала, хотя всё было сделано правильно (отсутствие индикации пути, по которому сохраняется файл).
5. Он может показывать недостатки в текущем описании интерфейса (слова "Start copying" вместо графического изображения кнопки в руководстве пользователя).

Как уже отмечалось, CWT-анализ фокусируется в основном на проблемах, которые пользователи испытывают при первом взаимодействии, не проходя предварительных тренировок. Такая постановка вопроса чрезвычайно важна для некоторых критических систем, таких как банкоматы или терминалы оплаты. Но та же самая ситуация возникает и в сложных программных системах, когда пользователь выполняет какую-либо задачу впервые. Пользователи часто изучают сложные программы постепенно, углубляясь в детали интерфейса по мере возникновения в том необходимости. Поэтому проблемы "первого знакомства" могут возникнуть даже при взаимодействии с давно используемой программой. Если система построена с уважением принципов CWT-анализа, то она позволяет пользователю плавно подниматься с уровня новичка до уровня эксперта.

3.3. Типичные ошибки при выполнении CWT-анализа и рекомендации

Как показывает практика, обучающиеся CWT-анализу часто проявляют два общих недопонимания.

Во-первых, многие упускают необходимость сформировать полный и точный список действий для выполнения задачи. То есть они сами точно не знают, как выполнить задачу, и блуждают по интерфейсу, пытаясь отыскать правильную последовательность действий, а потом они оценивают сложность этого блуждания. Иногда, действительно, бывает полезно оценить сложность такого блуждания, но это не метод CWT. В CWT вы должны начинать, имея в руках полный список отдельных элементарных действий, необходимых для выполнения задания. Причина такого подхода в том, что в "лучшем из миров" пользователь должен определить и выполнить оптимальную последовательность действий. Поэтому CWT рассматривает именно такую последовательность. Если в ходе анализа выясняется, что пользователь испытывает затруднения в определении и выполнении одного из действий, то вас интересует не то, как пользователь выйдет из положения, а сам факт того, что проблема возникла и интерфейс нуждается в доработке.

Второе момент заключается в недопонимании того факта, что CWT-анализ не тестирует реальных пользователей системы. Конечно, тестирование системы на пользователях – чрезвычайно нужная вещь, и об этом говорилось при описании проблемно-центрированного

подхода. Но сейчас речь идёт не об этом, а о средстве анализа, которое помогает разработчикам максимально отладить интерфейс ещё до его представления реальным пользователям. При проведении CWT необходимо представлять весть класс пользователей. Вследствие этого CWT часто находит больше проблем, чем вы можете найти при тестировании системы с одним реальным пользователем.

Основные рекомендации по проведению CWT-анализа сводятся к следующему. Вы определили задачу, класс пользователей, интерфейс и корректную последовательность действий. Далее рекомендуется собрать вместе группу разработчиков и других заинтересованных лиц (в учебных целях вы можете действовать в одиночку). И после этого начинается процесс анализа. Вы пытаетесь рассказать историю о том, почему пользователь выбрал бы каждое действие в списке корректных действий. Вы критикуете историю, чтобы сделать её правдоподобной. Рекомендуется держать в уме четыре вопроса:

- Будут ли пользователи пытаться произвести тот или иной эффект, который даёт действие?
- Видят ли пользователи элемент управления (кнопку, меню, переключатель и т.д.) для осуществления действия?
- Если пользователи нашли элемент управления, поймут ли они, что он производит тот эффект, который им нужен?
- После того как действие сделано, будет ли понятен пользователям тот отклик, который они получают, чтобы перейти к следующему действию с уверенностью?

Что делать с результатами CWT-анализа? Исправлять интерфейс! Большинство исправлений будут очевидны. Вероятно, самый тяжелый случай возникает тогда, когда у пользователя вообще нет никаких причин думать, что действие должно быть сделано. Поистине красивое решение этой проблемы – исключить это действие; пусть система сама выполнит его. Если так не получается, то необходимо перестроить задачу, чтобы пользователи начали делать то, о чём они знают, что это должно быть сделано, и в результате получали бы побуждение к выполнению "проблемного" действия.

Глава 4

Анализ GOMS

CWT-анализ позволяет выявить много проблем с интерфейсом, но не даёт ответа на вопрос, насколько сложен интерфейс, т.е. сколько времени тратит пользователь на выполнение задачи. Как мы увидим в приводимых ниже примерах, даже если интерфейс успешно проходит CWT-анализ, это не означает, что он оптимален с точки зрения трудоёмкости. Если есть несколько альтернативных вариантов построения интерфейса, то анализ GOMS позволяет выбрать тот из них, который требует меньше времени для решения задачи пользователя. В отличие от CWT, GOMS предполагает, что пользователь уже знает интерфейс, т.е. видит его не первый раз. Иными словами, GOMS оценивает время работы с интерфейсом обученного пользователя.

Аббревиатура GOMS означает Goals, Operations, Methods, Selections (цели, операции, методы, правила выбора). Модель GOMS состоит из описания методов, необходимых для достижения заданных целей. Методы представляются последовательностями шагов, состоящих из операций, которые выполняет пользователь. Метод может быть назван подцелью, т.о. методы образуют иерархическую структуру. Если существует более одного метода для достижения цели, то включаются правила выбора, с помощью которых в зависимости от контекста выбирается один метод.

Замечание. Мы здесь следуем той терминологии, которая принята в анализе GOMS и вытекает из его названия. Но должно быть ясно, что цель – это то, что мы раньше называли (репрезентативной) задачей, метод – это список действий пользователя, операции – элементарные действия пользователя. Некоторые особенности будут ясны из дальнейшего рассмотрения.

Операции в GOMS – это элементарные действия, которые нельзя разложить на более мелкие. Причём учитываются как внешние действия, т.е. те, что приводят к видимым физическим эффектам, так и внутренние, связанные с мышлением пользователя. Например, действие (шаг метода) "нажать кнопку <Start copying>" представляется в виде последовательности следующих операций:

- визуально определить местонахождение кнопки (мыслительная операция);
- навести на кнопку указатель мыши (внешняя операция);
- щелкнуть кнопкой мыши (внешняя операция).

В нашем курсе мы рассмотрим анализ интерфейсов для типичной конфигурации персонального компьютера, где в качестве устройств ввода-вывода выступают монитор, клавиатура и мышь. Практически все интерфейсные взаимодействия в этом случае можно описать следующими операциями:

- K* – нажатие клавиши;
- B* – клик кнопкой мыши;
- P* – наведение указателя мыши;
- R* – ожидание ответной реакции компьютера;
- H* – перенос руки с клавиатуры на мышь или наоборот;

D – проведение с помощью мыши прямой линии (например, выделение или прокрутка текста);

M – мыслительная подготовка (к осуществлению одной из перечисленных операций).

Разные пользователи выполняют указанные операции за разное время. Однако, напомним, что GOMS исследует работу опытного пользователя. Многочисленные исследования выявили средние значения времени операций, выполняемых опытными пользователями. Приведём их значения:

K	0.2 с
B	0.2 с
P	1.1 с
H	0.4 с
M	1.35 с

Оказывается, что время выполнения задачи, посчитанное с использованием этих значений, является хорошей средней оценкой сложности интерфейса и хорошо работает на практике.

Время ожидания R зависит от характера действия, выполняемого компьютером. Обратим внимание, что речь идёт только о задержках, связанных с работой интерфейса (например, ожидание открытия нового интерфейсного объекта). При анализе GOMS, как правило, не учитывается время, расходуемое компьютером на выполнение целевой вычислительной функции (скажем, преобразования аудио трека в MP3-файл). Такие вычислительные операции относятся уже к функциональному программному обеспечению; скорость их выполнения определяется быстродействием аппаратуры, объёмом обрабатываемых данных и эффективностью алгоритмов, но не связана со сложностью интерфейса. С другой стороны, очень быстрые реакции компьютера, кажущиеся для человека практически мгновенными (например, эхо-печать символа после нажатия клавиши), также не учитываются. Следует учитывать только ожидания, которые сбивают ритм выполнения других операций. Время таких ожиданий можно оценить "на глаз" при работе с реальной программой. Если реакция компьютера достаточно быстрая, но всё же ощутима, то "стандартное" время R рекомендуется брать равным 0.25 с.

Время операции D также может быть разным в зависимости от величины перемещения и других сопутствующих деталей. Его рекомендуется приблизительно оценить при работе с реальным приложением. В качестве "стандартного" значения можно взять 2 с.

В общих чертах, оценивание задачи (цели) методом GOMS производится следующим образом. Цель разбивается на подцели, для каждой подцели расписываются методы, методы могут раскрываться далее через внутренние методы и т.д. пока всё не будет сведено к отдельным операциям. Следующим важным шагом является добавление мыслительных подготовок M .

Мы не будем описывать каких-либо формальных правил для расстановки этих операций. Важно понять саму идею, и тогда станет ясно, как её применять. Вся последовательность операций разбивается на семантические группы. Например, набор на клавиатуре слова "слон" выполняется четырьмя операциями нажатия на клавиши $KKKK$ и рассматривается как отдельная семантическая единица. Перед ней нужно поставить операцию M . Действительно, перед тем как напечатать слово, человек должен сформировать его внутренний образ. Это и отражается добавлением операции M . Но после того как образ сформирован, слово печатается без дальнейших раздумий между отдельными буквами. Пример другой семантической единицы – открытие меню. Оно состоит из двух операций PB . Но прежде чем они могут быть выполнены, нужно решить, какой пункт меню вам нужен и найти, где он находится. Поэтому перед PB (но не между P и B) нужно поставить M . Если

затем с помощью ещё одной пары операций *PB* в меню выбирается элемент, то *M* ставить не нужно, т.к. идея о том, что нужно выбрать из меню, уже сформировалась у человека ранее (иначе он не стал бы вообще обращаться к меню). Напомним ещё раз, что мы исключаем блуждание по меню вследствие незнания, как выполнить действие.

Наконец, когда вся последовательность операций выписана, мы получаем общее время выполнения задачи простым суммированием времён отдельных операций. Процедура GOMS завершается. В результате мы имеем оценку среднего времени. Но это не единственный результат. Проведённый анализ часто позволяет понять, как можно улучшить интерфейс для сокращения времени решения задачи.

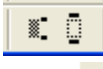

Рассмотрим конкретный пример анализа GOMS. Договоримся вначале о некоторых обозначениях. Вместо длинной последовательности операций *KKKKKK* будем писать *6K*. Время операций *R* и *D*, если оно отличается от "стандартного", будем указывать в секундах в скобках, например *R(1.5)*.

Возьмём в качестве исследуемой программы Microsoft Word 2003. Пусть поставлена цель: напечатать (вставить в текст) уравнение $x^2 + x - 1 = 0$ с помощью редактора формул (перед печатью уравнения набирался текст).

Для выполнения цели сформулируем три подцели:

1. Вызвать редактор формул через меню "Вставка | Объект... | Microsoft Equation 3.0".
2. Напечатать формулу в редакторе.
3. Выйти из редактора и подготовиться к продолжению набора текста.



Теперь опишем методы для каждой подцели:

1. Вызвать редактор формул через меню "Вставка | Объект... | Microsoft Equation 3.0".
 - 1.1. Войти в меню "Вставка"
 - 1.2. Выбрать пункт "Объект..."
 - 1.3. Выбрать объект "Microsoft Equation 3.0" путём скроллинга списка типов объектов.
2. Напечатать формулу в редакторе.
 - 2.1. Напечатать символ x .
 - 2.2. В окне "Формула" выбрать объект .
 - 2.3. В выпадающем меню объектов выбрать .
 - 2.4. Напечатать символ 2.
 - 2.5. Нажать клавишу \rightarrow .
 - 2.6. Напечатать " $+ x - 1 = 0$ ".
3. Выйти из редактора и подготовиться к продолжению набора текста.
 - 3.1. Нажать клавишу Esc.
 - 3.2. Нажать клавишу End (иногда при выходе из редактора формула остаётся выделенной, поэтому необходимо снять выделение).

Теперь распишем каждый метод с точностью до операции (повторим для наглядности название подцелей и методов):

1. Вызвать редактор формул через меню "Вставка | Объект... | Microsoft Equation 3.0".
 - 1.1. Войти в меню "Вставка"

<i>H</i>	(переместить руку на мышь, т.к. мы набирали текст)
<i>P</i>	(переместить указатель мыши)
<i>B</i>	(клик мыши)
 - 1.2. Выбрать пункт "Объект..."
PB

- 1.3. Выбрать объект "Microsoft Equation 3.0" путём скроллинга списка типов объектов.
PB (перемещение указателя и фиксация мыши на элементе управления скроллинга)
D(3.0) (скролинг вниз и поиск нужной строки, экспериментальная оценка времени)
PBB (установка указателя и двойной щелчок)
R(0.8) (ожидание запуска редактора формул)
2. Напечатать формулу в редакторе.
 - 2.1. Напечатать символ x .
H (перемещение руки на клавиатуру)
K (печать x)
 - 2.2. В окне "Формула" выбрать объект .
H (перемещение руки на мышь)
PB
 - 2.3. В выпадающем меню объектов выбрать .
PB
 - 2.4. Напечатать символ 2.
H
K
 - 2.5. Нажать клавишу \rightarrow .
K
 - 2.6. Напечатать " $+ x - 1 = 0$ ".
7K (нажать Shift, печатать + (отпустить Shift), печатать оставшиеся символы)
3. Выйти из редактора и подготовиться к продолжению набора текста.
 - 3.1. Нажать клавишу Esc.
K
R(0.8) (ожидание выхода из редактора формул и переход в текстовый режим)
 - 3.2. Нажать клавишу End.
K

В результате получаем следующую последовательность операций:

HPBPBPBD(3.0)PBBR(0.8)HKHPBPBHK7KKR(0.8)K

Можно возразить, что мы иногда указываем лишнюю операцию *H*: напечатать x и 2 можно левой рукой, тогда как правая остаётся на мыши. Но дело в том, что человек действует в соответствии со своим навыком, который говорит ему о том, что нужно перейти к клавиатуре. Ведь иногда нужно вводить символы, находящиеся в ведении левой руки. Так что оставить операции *H* будет более верным решением. Их всегда нужно предусматривать при переходе от клавиатуры к мыши и наоборот (за исключением, быть может, очень специальных случаев). Заметьте также необходимость учёта переключений регистра (клавиша Shift). Если при наборе требуются частые переключения регистра, это безусловно замедляет процесс.

Теперь добавим мыслительные подготовки:

MHPBPBPBD(3.0)PBBR(0.8)MHK MHPBPB MHK MK M7K MKR(0.8)MK

Просуммируем время выполнения отдельных операций и получим общее время решения задачи. Оно складывается из следующих величин: $8M = 10.8$, $4H = 1.6$, $6P = 6.6$, $7B = 1.4$, $D(3.0) = 3.0$, $2R(0.8) = 1.6$, $12K = 2.4$. Общий итог – 27.4 с.

Одно из назначений GOMS – сравнение по трудоёмкости различных интерфейсов. То же самое уравнение можно было бы набрать в текстовом режиме без использования редактора формул. Результат выглядит следующим образом (хуже с типографической точки зрения):

$$x^2 + x - 1 = 0$$

При наборе мы не забыли о вводе пробелов между элементами формулы (в редакторе формул пробелы не вводятся), о курсивном начертании переменной (курсивный и прямой шрифт переключался с помощью значка в строке форматирования). Кроме того, в математических формулах лучше смотрится длинный минус, который вводится с помощью комбинации клавиш Ctrl+Num-. Мы также считали, что необходимо переключиться в режим ввода латиницы и вернуться из него, и что в строке форматирования присутствует значок для ввода верхних индексов. Мы не будем приводить подробный анализ GOMS такого способа набора, ограничимся только его результатом. Общий итог по операциям получился $8M$, $10H$, $3P$, $6B$, $20K$. Оценка среднего времени решения задачи составляет 23.3 с.

Отметим, что анализ GOMS позволяет не только оценить трудоёмкость интерфейса, но и вскрыть проблемные места. Бросается в глаза довольно большое количество мыслительных подготовок при выполнении задания (они составляют примерно половину общего времени). Это связано с постоянными переключениями между клавиатурным вводом и работой с мышью. Ниже мы приведём пример ещё одной системы ввода формул, которая вообще не требует использования мыши, а работает только с клавиатурой.

Другая возможность, которую открывает GOMS – это сопоставление по трудоёмкости отдельных подцелей или выявление подцелей, требующих иной реализации интерфейса. Например, первая подцель в рассмотренном примере (запуск редактора формул) требует операций

$$MHPBPBPD(3.0)PBRR(0.8)$$

Для неё получаем оценку времени 10.95 с. Можно предложить простую доработку интерфейса: дать возможность пользователю, если он часто вводит формулы, вывести в панель управления значок для запуска редактора формул. В этом случае первая подцель реализовывалась бы следующей последовательностью операций: $MHPBR(0.8)$, а время составило бы всего 3.85 с. С учётом такой доработки общее время решения задачи с помощью редактора формул составило бы 20.3 с, т.е. стало бы меньше, чем при обычном вводе.

В приведённых выше оценках мы считали, что время нажатия на клавишу при вводе составляет 0.2 с. Однако это справедливо только в случае, когда вводимая последовательность символов кажется естественной для человека, как это было в рассмотренном примере. Если необходимо ввести последовательность букв, которая кажется случайной (скажем, текст на незнакомом языке или какие-то условные обозначения или идентификаторы), то время операции K увеличивается до 0.5 с. Если нужно печатать сложные коды (абстрактные числа, смесь букв и цифр), то время ещё сильнее возрастает до 0.75с.

Рассмотренных операций может оказаться недостаточно для некоторых интерфейсов. Например, если вы используете сенсорные панели или графические планшеты, то понадобятся специальные, связанные с ними операции. Их временные характеристики можно определить экспериментально или найти в специальной литературе. Скажем здесь только об одной операции, которая возникает, например, в командных интерфейсах. Это операция извлечения простого элемента знания из долговременной памяти, скажем, название команды "dir" и правила её использования. Среднее время такой операции оценивается в 1.2 с. Если этот элемент знания затем используется повторно, то он уже извлекается из краткосрочной памяти, и время оценивается в 0.6 с.

Командные интерфейсы сегодня значительно уступили свои позиции интерфейсам графическим, но их нельзя списывать со счёта. Многие специальные задачи намного более эффективно решаются с помощью командных интерфейсов. Командный интерфейс требует больших усилий и времени для изучения (необходимо запомнить множество команд и правил их использования). Но, будучи освоен, он может быть крайне эффективен. Рассмотрим в заключение пример набора уравнения $x^2 + x - 1 = 0$ в системе LaTeX (это издательская система, разработанная, в первую очередь, для набора математических текстов, обеспечивающая чрезвычайно высокий полиграфический уровень получаемого документа).

Для того чтобы ввести уравнение в системе LaTeX, достаточно набрать $\$x^2+x-1=0\$$. Далее система сама преобразует его в читаемый вид при обработке всего документа. Оценим сложность этого (командного) интерфейса методом GOMS. Для человека, знающего LaTeX, данная последовательность символов кажется вполне естественной: $\$$ означает переход в математический режим, $^$ означает степень, все остальные символы используются по прямому назначению. Поэтому трудоёмкость операции K обычная и составляет 0.2 с. Количество операций определяется числом символов (11); 4 из них вводятся в верхнем регистре, поэтому добавим 4 нажатия клавиши Shift; кроме того, учтём необходимость перевода в латиницу и возврата к кириллице путём нажатия Alt+Shift (4 нажатия). Всего получаем 19 нажатий. Им предшествует операция извлечения конструкции из памяти. Поэтому общее время будет $19 * 0.2 + 1.2 = 5$ с. Сравните это время с тем, что предполагает интерфейс Microsoft Word. Командный интерфейс побеждает с большим отрывом.

Глава 5

Золотые правила построения интерфейсов

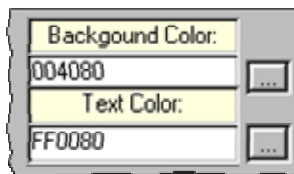
В этой главе мы опишем рекомендации, которые выкристаллизовались за годы изучения проблем человеко-машинного взаимодействия. Многие специалисты считают их наиболее существенными при построении интерфейсов, поэтому мы и назвали их "золотыми правилами". Эти правила могут также использоваться для экспертной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

5.1. Правила Нильсена-Молиха

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

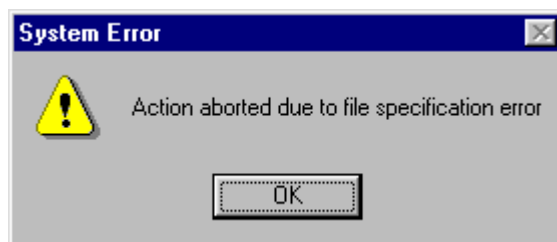
1. **Простой и естественный диалог.** Простота означает, что не должно присутствовать не относящейся к теме или редко используемой информации. Старайтесь добиваться максимального следования задачам пользователя, минимизируя сложность поиска соответствия между семантиками задачи и интерфейсом. Важно представить точно ту информацию, в которой нуждается пользователь, следуя принципу "лучше меньше да лучше". Вся редко используемая информация должна быть спрятана. Естественность означает порядок, соответствующий задаче. Информация, которая выводится на экран, должна появляться в естественном порядке, т.е. в порядке, соответствующем ожиданиям пользователя. Вся взаимосвязанная информация должна группироваться в одном месте. В одной программе для тестирования студентов была сделана следующая вещь. Перед началом теста студент должен был выбрать свою специальность из выпадающего списка. Каждый элемент списка содержал код специальности и её название. Порядка двухсот специальностей в списке были упорядочены по коду – так было проще сделать, потому что код шёл в начале строки. Но для пользователя-студента, который знал примерное название своей специальности, но был абсолютно не в курсе её кода, такой порядок был просто ужасен. Приходилось последовательно просматривать весь список, прокручиваемый в маленьком окошке, и тщательно выискивать в нём знакомые слова. Это пример несоответствия порядка предоставления информации решаемой задаче.
2. **Говорите на языке пользователя.** Используйте слова и понятия из мира пользователя. Не используйте специфических инженерных терминов. Разработчики одной программы для юристов в США, проводя тестирование интерфейса с пользователями, столкнулись с тем, что те вместо слова "параметр" постоянно употребляли слово "периметр". Термин "параметр" казался совершенно обычным для разработчиков и постоянно использовался в инструкциях к программе, но оказалось, что этого слова нет в словаре юристов (американских). Слово "опция" было им лучше знакомо.

В качестве другого примера приведём такой фрагмент интерфейса:



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

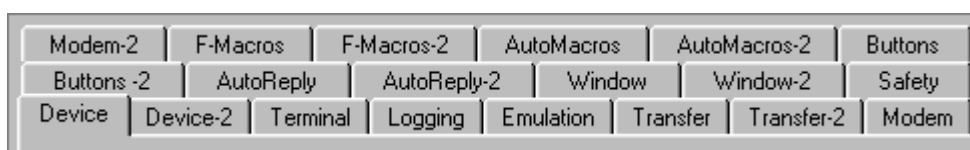
Ещё один пример иллюстрирует использование инженерного языка, непонятного пользователю:



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

3. **Минимизируйте загрузку памяти пользователя.** Не заставляйте пользователя помнить вещи от одного действия к следующему. Оставляйте информацию на экране до тех пор, пока она не перестанет быть нужной.

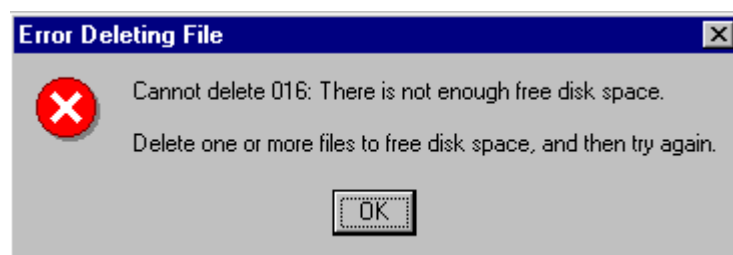
В этом месте хочется упомянуть о чрезмерном использовании закладок, как это сделано в программе Zoc:



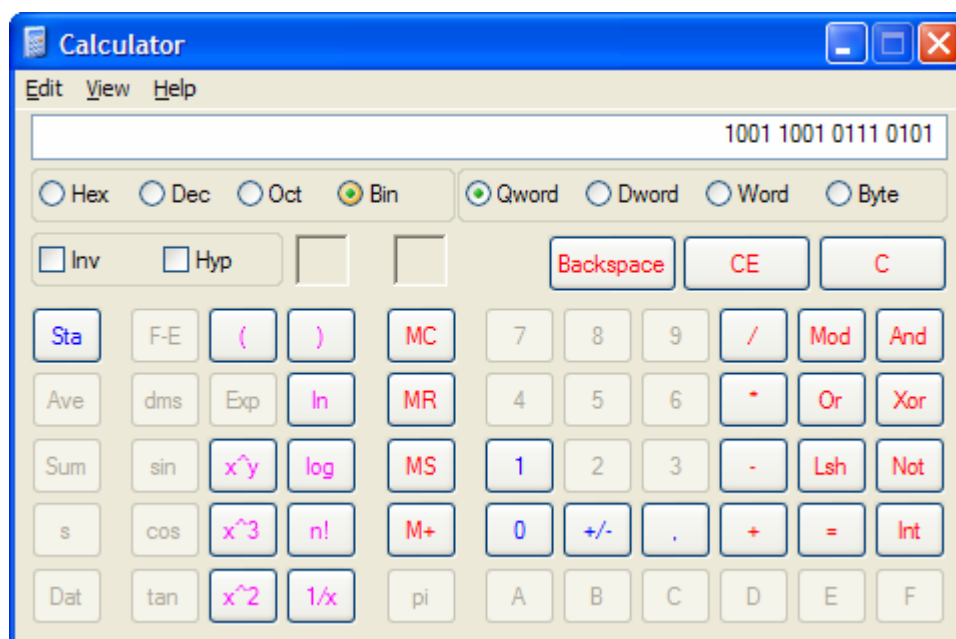
После каждого использования закладки переупорядочиваются и, если вы что-то пытаетесь отыскать, перебирая их, нужно держать в памяти, какие закладки вы уже открывали. Хорошим стилем считается делать только один ряд закладок.

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

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



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



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

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

5.2. Принципы организации графического интерфейса

А теперь рассмотрим несколько принципов организации графического интерфейса.

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

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

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

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

Принцип "цвет как приложение". Не полагайтесь на цвет как носитель информации; используйте его умеренно, чтобы лишь акцентировать информацию, передаваемую другими средствами. Цвет значительно проще использовать неправильно, чем верно. Для разных людей разные цвета означают разные вещи, особенно эти вариации сильны между различными культурами. Так, красный цвет означает опасность в странах европейской

культуры, смерть – в Египте, жизнь – в Индии и счастье – в Китае. Дополнительная проблема в том, что некоторые люди не могут различать цвета: около 7 % взрослых страдают от той или иной формы дальтонизма.

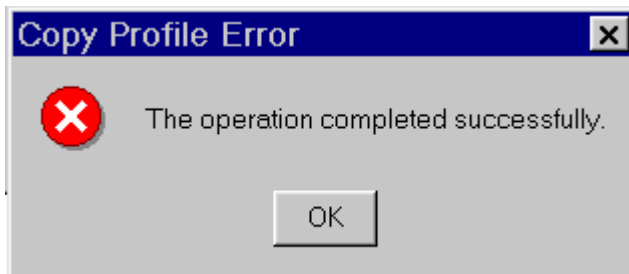
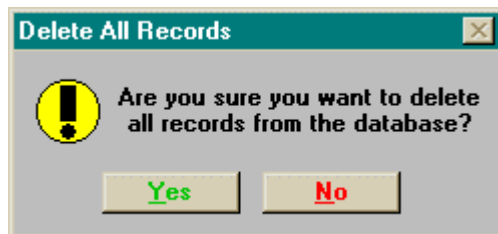
Хорошее правило для большинства интерфейсов – разрабатывать их в чёрно-белом варианте, убедиться, что они работоспособны, а затем добавить минимальный цвет для их окончательного облика. Цвет, безусловно, полезен, когда необходимо выделить предупреждение или информационное сообщение, но обязательно дайте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

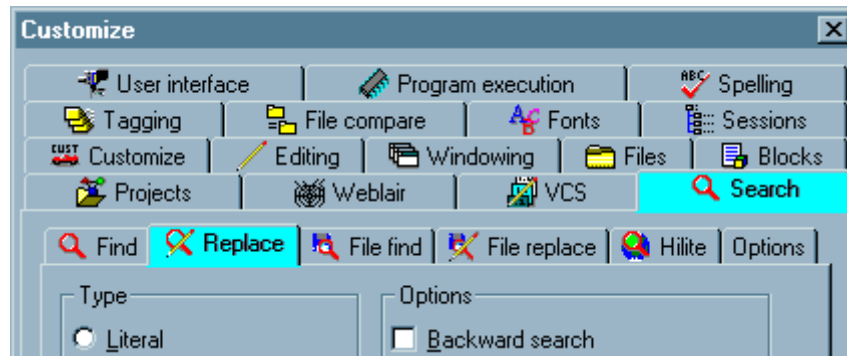
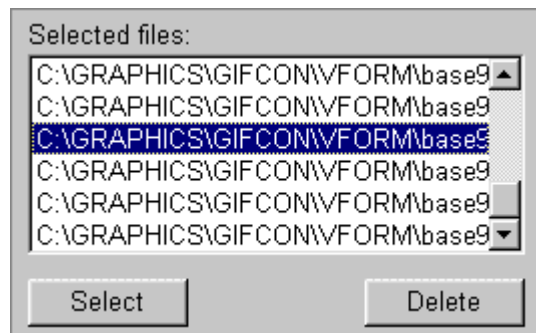
Если только вы не опытный графический дизайнер, придерживайтесь принципа минимума цвета для создания привлекательного интерфейса. Постарайтесь придерживаться серых тонов в большей части системы, дайте немного ярких красок для логотипа или метки для придания индивидуальных отличий вашему продукту. Помните,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

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

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

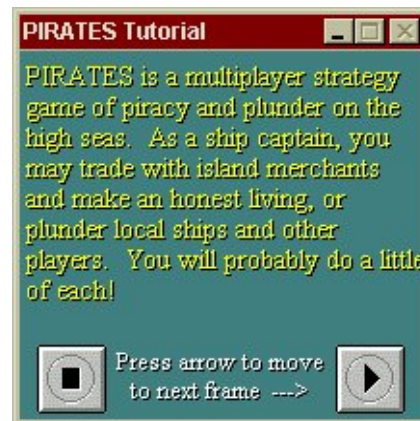
В заключение приведём несколько плохих примеров, нарушающих принципы, изложенные в этой главе. Хорошие примеры у всех перед глазами. ОС Windows, как бы к ней ни относиться, в огромной степени соответствует принципам хорошего дизайна интерфейса.



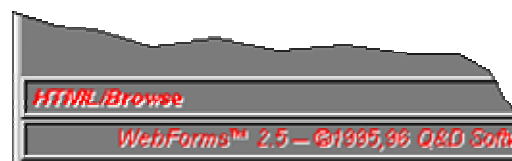


Whenever your local SMS Administrator sends you an actual software Package, the SMS Package Command Manager will appear (usually at network login time) displaying the available Package(s). The following screenshots display scenes similar to what you will see when you receive an actual SMS Package.

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



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



Думается, комментарии излишни.

Рекомендуемая дополнительная литература

На русском языке:

1. Акчурин Э.А. Человеко-машинное взаимодействие. Учебное пособие. Солон, 2008. 96 с.
2. Логунова О.С., Ячиков И.М., Ильина Е.А. Человеко-машинное взаимодействие: теория и практика. Учебное пособие. Феникс, 2006. 285 с.

На английском языке:

1. Human Computer Interaction: Concepts, Methodologies, Tools, and Applications. / P. Zaphiris, C. S. Ang eds. London, Hershey: Information Science Reference, 2009. 4 Vols. 2734 P.
2. Human-Computer Interaction. Fundamentals / A. Sears, J. A. Jacko eds. CRC Press, 2009. 331 P.
3. Andrews K. Human-Computer Interaction. Lecture Notes. Graz University of Technology, 2009. 181 P.
4. Sharp H., Rogers Y., Preece J. Interaction Design: Beyond Human-Computer Interaction. Wiley, 2007. 776 P.
5. Berkshire Encyclopedia of Human-Computer Interaction / W. S. Bainbridge ed. Berkshire Publishing Group LLC, 2004. 2 Vols. 931 P.
6. Dix A., Finlay J., Abowd G., Beale R. Human Computer Interaction, 3rd Edition. Prentice Hall, 2004. 817 P.

Много информации по данному предмету можно найти в Интернете, особенно на английском языке. Ключевые слова для поиска: human-computer interaction, interface design, usability.