Создание качественного интерфейса: принципы и шаблоны

Мы говорили о том, в какой последовательности следует принимать решения, чтобы можно было задумать и спроектировать желанный и эффективный продукт.

Рассмотрим принципы и шаблоны проектирования взаимодействия, которые позволяют сделать проектирование качественным.

Принципы проектирования – это рекомендации по проектированию полезных и желанных продуктов, систем и услуг, а также рекомендации по успешному и этичному проектированию.

Шаблоны проектирования – это типовые обобщенные решения конкретных классов проблем проектирования.

Принципы проектирования взаимодействия

Принципы проектирования взаимодействия — это обобщенные рекомендации, ориентированные на особенности поведения, формы и содержания продукта. Они поддерживают проектирование такого поведения продуктов, которое служит потребностям и целям пользователей и вызывает у них положительные эмоции при использовании этих продуктов. По сути, эти принципы представляют собой набор правил, основанных на ценностях, носителями которых мы являемся как проектировщики, и на нашем опыте, связанном с претворением этих ценностей в жизнь. В основе наших ценностей – мысль о том, что технология должна служить разуму и творчеству человека (а не наоборот) и что опыт общения человека с технологией должен структурироваться сообразно возможностям человеческого восприятия, познания и движения.

Принципы применяются на всем протяжении процесса проектирования, помогая нам преобразовывать задачи и требования, возникающие в ходе разработке сценариев, в формализованные структуры и поведенческие реакции интерфейса.

Принципы проектирования действуют на нескольких уровнях детализации – от общей практики проектирования взаимодействия до конкретики интерфейса. Границы между уровнями, мягко говоря, размыты, однако принципы проектирования взаимодействия можно в целом разбить на следующие категории.

Категории принципов проектирования взаимодействия:

1. **Ценности проектирования** – императивы эффективной и этичной практики проектирования.

2. **Концептуальные принципы** помогают определять *сущность продукта* и его место в более широком контексте использования, который требуется пользователям.

3. **Поведенческие принципы** описывают, как *продукт должен себя вести* – в целом и в конкретных ситуациях

4. **Интерфейсные принципы** описывают эффективные стратегии визуальной коммуникации поведенческих и информационных аспектов интерфейса.

Большинство принципов проектирования взаимодействия и визуального дизайна не привязаны к конкретной платформе, хотя некоторые платформы, например мобильные устройства и

системы реального времени, требуют особых соображений, связанных с такими ограничения ми, как размер экрана, способы ввода или контекст применения.

Одно из главных назначений принципов – оптимизировать опыт пользователя, взаимодействующего с системой. В случае офисных инструментов и других продуктов, не ориентированных на развлечения, такая оптимизация опыта означает **минимизацию трудозатрат**. Сокращать следует такие виды работ:

1. **Когнитивная работа** — понимание поведения продукта, а также текста и организующих структур.

2. **Мнемоническая работа** — запоминание поведения продукта, векторов команд, паролей, названий и расположения объектов данных и элементов управления, а также других связей между объектами.

3. **Работа зрения** — поиск стартовой точки на экране, поиск одного объекта среди многих, расшифровка визуальной планировки, вы явление различий между элементами интерфейса, имеющими цветовую кодировку.

4. **Физическая работа** — число нажатий на клавиши, перемещения мыши, использование жестов (щелчок, перетаскивание, двойной щелчок), переключение между режимами ввода, количество щелчков для осуществления навигации.

Большинство принципов направлены на минимизацию работы и в то же время на обеспечение более качественной обратной связи и лучшего информирования пользователя в рамках определенной ситуации.

Следует также заметить, что некоторые развлекательные продукты (в частности, игры) способны завоевывать внимание пользователей, требуя от них выполнения разумного объема определенной работы и вознаграждая их за эту работу. Проектирование взаимодействий подобного рода требуется тонкого подхода.

Ценности проектирования

Принципы – это правила, ведущие к действиям и обычно опирающиеся на ряд ценностей и убеждений. Он применим к любой дисциплине проектирования, которая служит потребностям человека.
Задача проектировщиков взаимодействия – создавать такие проектные решения, которые:

Этичны [*тактичны, заботливы*]:

не причиняют вреда;

улучшают положение человека.

Целенаправленны [*полезны, применимы*]:

помогают пользователям решать их задачи и достигать целей;

учитывают контексты и возможности пользователей.

Прагматичны [*жизнеспособны, осуществимы*]:

помогают организации, внедряющей ваши проектные решения, достигать своих целей;

учитывают требования бизнеса и технические требования.

Элегантны [*эффективны, искусны, вызывают эмоции*]:

представляют собой простые, но полноценные решения;

обладают внутренней (самоочевидной, понятной) целостностью;

учитывают и пробуждают эмоции и познавательные процессы.

Проектирование этичного взаимодействия

Интерактивные продукты *обладают возможностью изменять мир*, и проектировщики должны стремиться к тому, чтобы это были изменения *к лучшему*. Относительно легко спроектировать систему, которая хорошо относится к пользователям, однако вычислить, какое косвенное влияние система оказывает на других людей, гораздо сложнее.

Не навреди

Продукты не должны никому причинять вреда – или, принимая во внимание сложность реального мира, должны *минимизировать ущерб*. Вот некоторые разновидности вреда, в причинении которого может принимать участие интерактивная система:

1. межличностный вред (потеря достоинства, оскорбления, унижение);
2. психологический вред (замешательство, дискомфорт, раздражение, давление, скука);
3. физический вред (боль, травмы, лишения, смерть, угрозы безопасности);

4. экологический вред (загрязнение, сокращение числа биологических видов);

5. социальный и общественный вред (эксплуатация, создание или поддержание несправедливости).

Предотвращение двух первых разновидностей вреда требует глубокого понимания аудитории пользователей, а также убежденности заинтересованных лиц в том, что такие вопросы могут решаться в рамках проекта. Предотвращение физического вреда требует глубокого понимания эргономических принципов и соответствующего использования элементов интерфейса с целью минимизации работы. Очевидно, что последние два пункта к большинству продуктов не относятся, однако можно придумать такие примеры, когда они будут иметь значение, например система управления нефтяной платформой или система электронного голосования.

Улучшай положение человека

Чтобы результатом проектирования стало действительно этичное взаимодействие, мало просто не вредить – требуется способствовать. Вот некоторые улучшения, которым в широком смысле могут содействовать интерактивные системы:

1. улучшение понимания (индивидуального, социального, культурного);

2. повышение эффективности/действенности отдельных личностей и групп;

3. совершенствование коммуникаций – как между отдельными личностями, так и между группами людей;

4. снижение социокультурной напряженности между личностями и группами;

5. умножение справедливости (финансовой, социальной, правовой);

6. сглаживание культурных противоречий путем стимулирования общественного согласия.

Проектировщикам, начинающим работу над новым проектом, необходимо в глубине души осознавать эти глобальные вопросы. Следует всегда предоставлять людям возможность творить добро, даже если это немного выходит за рамки проекта.

Проектирование целенаправленного взаимодействия

Целенаправленность – это не только понимание целей пользователей, но и осознание их ограничений. В этом смысле персонажи служат качественной меркой, поскольку шаблоны поведения, которые вы сможете наблюдать в ходе исследований и при создании персонажей, дадут вам хорошее представление о сильных *и* слабых сторонах пользователей. Целеориентированное проектирование помогает проектировщикам создавать продукты, которые поддерживают пользователей в том, в чем они слабы, и делают их более производительными в том, в чем они сильны.

Проектирование прагматичного взаимодействия

Чтобы иметь ценность, проектирование должно воплощаться в продукте. Будучи созданным, продукт должен увидеть свет. А увидев свет, он должен приносить прибыль создателям.

Проектирование элегантного взаимодействия

Элегантность в словаре определяется как «грациозность и сдержанная красота стиля» или «научная точность, аккуратность и простота». Элегантность в проектировании взаимодействия – включает оба этих идеала.

Создавай простые, но полноценные решения

Интерфейс должен содержать лишь те экраны и элементы, которые требуются для решения поставленной задачи. Это касается и поведения: **следует дать пользователю простой набор инструментов, позволяющий ему добиваться великолепных результатов.** В визуальном дизайне подобная экономия означает использование минимального числа визуальных отличий для четкой передачи нужного смысла. В хорошем проектировании **«меньше» означает «больше»**, и проектировщики должны стремиться разрешать проблемы проектирования с минимальными добавлениями к форме и поведению в соответствии с ментальными моделями персонажей. Такой подход хорошо знаком программистам, которые соглашаются, что лучшие алгоритмы коротки и понятны.

Добивайся внутренней целостности

Качественно спроектированный продукт оставляет **ощущение единого целого**, в котором все составные части сбалансированы и гармоничны. Некачественно спроектированный продукт или продукт, вообще не проектировавшийся, обычно выглядит и создает впечатление, будто его наспех сколотили из выбранных случайным образом разрозненных кусков. Часто такой продукт – результат конструирования на основе модели реализации, когда различные команды разработчиков трудятся над различными модулями интерфейса, не общаясь между собой, или когда программная и аппаратная части проектируются раздельно. Это полная противоположность тому, к чему мы стремимся. Целеориентированный процесс проектирования, в котором концепция продукта зарождается сразу и полностью на высшем уровне с по следующей поэтапной детализацией, дает идеальную среду для создания внутренне целостных проектов. Применение сценариев как точки отсчета проектирования и тестирования решений гарантирует, что решения будут пронизаны единой нитью повествования.

Учитывай и пробуждай эмоции и познавательные процессы

Желание – ограниченная эмоция, если речь идет о продукте, у которого есть определенное назначение – особенно если этот продукт используется в корпоративной среде или же его назначение сугубо техническое либо узкоспециальное. Вряд ли стоит пытаться сделать так, чтобы оператор, работающий с прибором радиотерапии, начал желать эту систему. Мы хотим, напротив, чтобы он проявлял осторожность и, возможно, испытывал почтение к опасной энергии, подчиненной системе. Так что мы как проектировщики делаем все возможное, чтобы этот человек сосредоточился на пациенте и лечении. Поэтому вместо того, что можно назвать *желанием*, мы предлагаем элегантность (в смысле грациозности), которая означает, что пользователь получает стимулы и поддержку – как когнитивные, так и эмоциональные – независимо от контекста, в котором он находится.

Шаблоны проектирования взаимодействия

Шаблоны проектирования — решения целых классов проблем проектирования, возникающие путем выявления и обобщения ценных проектных находок. Эта деятельность по формализации знания и фиксации наилучших решений в области проектирования служит многим важным целям:

1. сократить время и усилия, затрачиваемые на проектирование в новых проектах;

2. повысить качество проектных решений;

3. способствовать улучшению коммуникации между проектировщиками и программистами;

4. повысить профессиональный уровень проектировщиков.

Хотя аспекты применения шаблонов, связанные с обучением проектированию и совершенствованием процесса, очень важны, шаблоны проектирования взаимодействия в первую очередь ценны потому, что служат представлением оптимальных или близких к оптимуму взаимодействий пользователя с теми сферами деятельности, на которые эти шаблоны ориентированы.

Сайт с паттернами (шаблонами): http://ui-patterns.com/

Определение шаблонов проектирования взаимодействия и их использование

Шаблоны всегда применяются в рамках некоторого контекста и конструируются так, чтобы быть применимыми в типичных ситуациях, которые имеют схожий контекст использования, схожие ограничения и условия.

Описывая шаблон, важно четко задать ситуацию, в которой применимо решение, дать один или несколько конкретных примеров, перечислить абстрактные признаки, характерные для всех примеров, а также рассуждения, объясняющие, почему решение является хорошим.

Шаблоны – это не готовые к употреблению компоненты; каждая реализация шаблона немного отличается от всех других.

Мир проектирования программного обеспечения до некоторой степени заворожен идеей, что исчерпывающий каталог шаблонов позволит даже начинающим проектировщикам быстро и легко собирать целостные решения проектирования (если есть четкое представление о потребностях пользователя). И хотя мы сталкивались с определенными подтверждениями этой идеи, когда речь шла об опытных проектировщиках взаимодействия, чисто механическое использование шаблонов, без понимания контекста их применения, попросту невозможно. Крайне важно, в какой среде будет использован шаблон, важны другие шаблоны, которые входят в его состав, примыкают к нему или окружают его. То же верно и для шаблонов проектирования взаимодействия. Суть каждого шаблона – в отношениях между представленными объектами, а также между представленными объектами и целями пользователя. Точная реализация шаблона наверняка будет отличаться в каждом конкретном случае, и определяющие её объекты, естественно, будут разными в разных предметных областях, однако отношения между объектами останутся по существу такими же.

Типы шаблонов проектирования взаимодействия

Подобно большинству прочих шаблонов проектирования, шаблоны проектирования взаимодействия можно выстроить в иерархию, простирающуюся от концептуального уровня до уровня отдельных элементов интерфейса. Подобно принципам их можно применять на раз личных уровнях инфраструктуры интерфейса (причем различия между уровнями точно так же могут быть весьма размытыми):

1. **Шаблоны позиционирования** могут применяться на концептуальном уровне и помогают определить тип продукта в отношении к пользователю. Пример такого шаблона – *временный тип*, который означает, что использование продукта скоротечно и требуется только в составе деятельности, направленной на достижение более глобальной цели.

2. **Структурные шаблоны** решают проблемы, связанные с управлением отображением информации и функциональных элементов на экране.

3. **Поведенческие шаблоны** решают широкий спектр проблем, относящихся к конкретным взаимодействиям с теми или иными функциональными элементами или элементами данных. В эту категорию попадает поведение отдельных компонентов интерфейса.

Структурные шаблоны являются, по всей видимости, наименее документированными, однако при этом они распространены повсеместно.

Создание собственного каталога шаблонов – один из важнейших аспектов обучения проектировщика взаимодействия. Изучая лучшие образцы работы других специалистов, мы коллективно совершенствуем идиомы взаимодействия во благо пользователей, а проделанная работа позволяет нам сосредоточить усилия на решении новых проблем, избавляя от необходимости изобретать колесо.

Техническая платформа и тип интерфейса

Платформой можно считать сочетание аппаратных и программных средств, позволяющих продукту функционировать – как в плане взаимодействия с пользователем, так и на уровне внутренних механизмов.

Несомненно, вы знакомы с некоторыми распространенными платформами для интерактивных продуктов, включая приложения для настольных компьютеров, вебсайты и вебприложения, киоски, автомобильные системы, портативные устройства (фотоаппараты, телефоны, КПК), домашние развлекательные комплексы (игровые консоли, телевизионные тюнеры, музыкальные центры) и профессиональные устройства (медицинские и научные приборы). Глядя на этот список, вы можете отметить, что понятие «платформа» не имеет четкого определения. Это скорее сокращение для описания ряда важных особенностей продукта, таких как физическая форма, размер и разрешение дисплея, способы ввода и подключения к сети, операционная система и возможности для работы с данными.

Все эти факторы существенным образом влияют на способ проектирования, конструирования и использования продукта. **Выбор платформы – это поиск баланса между наилучшей поддержкой потребностей и контекста персонажей с одной стороны и ограничениями и задачами бизнеса, а также технологическими возможностями – с другой.**

Тип интерфейса – это способ описать то, как много внимания пользователь будет уделять взаимодействию с продуктом и каким образом продукт будет реагировать на это внимание. Как и все прочие проектные решения, выбор типа интерфейса должен опираться на понимание вероятных контекстов и среды применения продукта.

Большинство людей демонстрируют поведение, типичное для их профессии. Солдат насторожен и бдителен, сборщик налогов скучен и безразличен ко всему, актер ярок и заметен в любом обществе, человек из обслуживающего персонала бодр и услужлив. Подобно людям продукты предъявляют себя пользователю в какой-то определенной доминирующей манере.

То, как программа предъявляет себя, формирует отношение пользователя к ней, а это, в свою очередь, сильно влияет на удобство использования программного продукта. **Программа, внешнее представление и поведение которой конфликтуют с её назначением, раздражает и выглядит неуместно.** Внешнее представление и поведение продукта должны соответствовать способу его использования, а не личным вкусам проектировщиков.

Техническая платформа и тип интерфейса тесно связаны: различные аппаратные платформы благоприятствуют приложениям различных поведенческих типов. Приложение, работающее на мобильном телефоне, очевидно, должно разрабатываться с учетом иной разновидности пользовательского внимания, чем образовательная программа для игровой приставки.

Типы интерфейса настольных приложений

Термин «настольные приложения» мы используем в качестве обобщающего для программ, работающих на современном персональном компьютере.

Интерфейс настольных приложений можно отнести к одному из трех типов: **монопольный**, **временный** и **фоновый**. Программа, являющаяся монопольной, не будет удобной, если не ведет себя соответственно этому статусу. Рассмотрим типы подробнее.

Монопольный тип

К приложениям **монопольного типа** относятся программы, полностью завладевающие вниманием пользователей на длительные периоды времени.

Монопольное приложение предлагает пользователям большой набор тесно связанных функций и возможностей, а пользователи обычно разворачивают такое приложение на весь экран и работают с ним непрерывно. Вот характерные примеры приложений такого типа: текстовые процессоры, электронные таблицы, программы для работы с электронной почтой. Пользователи монопольных программ часто оказываются в состоянии потока.

Для продуктов с монопольным интерфейсом характерна непрерывная работа в течение длительных отрезков времени. В процессе работы пользователя монопольный продукт является его основным инструментом и преобладает над остальными.

Принципы проектирования интерфейсов с монопольным типом

1. Не жалейте места на экране

Поскольку взаимодействие пользователя с монопольным приложением занимает почти весь сеанс работы за компьютером, программа может смело затребовать все доступное пространство экрана. Никакая другая программа не станет с ней конкурировать, так что не растрачивайте экранное пространство понапрасну, но и не стесняйтесь – берите столько, сколько нужно. Если требуется предоставить пользователю четыре панели инструментов, создавайте четыре. Для программы, позиционируемой в другой категории, это было бы чересчур, но приложение-монополист – в своем праве.

В большинстве случаев окно монопольного приложения развернуто на весь экран. В отсутствие явных инструкций от пользователя ваше самостоятельное приложение должно по умолчанию занимать весь экран. Программа должна позволять пользователю изменять размер окна и оставаться работоспособной при любом его размере, но по умолчанию **интерфейс должен быть ориентирован на полный экран**, а не на другие, более редкие варианты.

2. Используйте строгий визуальный стиль

Поскольку пользователь смотрит на монопольное приложение в течение продолжительного времени, позаботьтесь о том, чтобы приглушить цвета и текстуру визуальной части. **Придерживайтесь консервативной цветовой палитры.** Крупные и яркие элементы управления способны привлечь новичков, но через пару недель ежедневной работы они станут казаться кричащими. Крохотные точки или легкие цветовые акценты в конечном счете будут эффективнее крупных клякс и к тому же позволят вам плотнее упаковать управляющие элементы.

Пользователь будет подолгу смотреть на одни и те же палитры, меню и панели инструментов и вследствие значительного стажа работы привыкнет к расположению элементов интерфейса. Это дает проектировщику возможность достичь высокой информативности при минимальном «расходе» пикселов. Панели инструментов и соответствующие элементы управления могут иметь уменьшенные размеры. Служебные элементы, например разделители экрана, линейки и полосы прокрутки, можно располагать плотнее и делать более узкими.

3. Обогащенная обратная связь

Монопольные приложения – отличная платформа создания сред с обогащенной обратной связью для пользователей. **Вы можете повышать продуктивность, расширяя интерфейс дополнительными информационными блоками.** Строка состояния у нижнего края экрана, концы полос прокрутки, обычно занятые ползунками, строка заголовка и другие «пыльные» углы видимой области программы могут быть заполнены индикаторами её состояния, состояния обрабатываемых данных, со стояния системы и другими подсказками, повышающими производительность труда пользователя. Однако будьте осмотрительны: **обогащая визуальную обратную связь, следите за тем, чтобы не получить в результате безнадежно замусоренный интерфейс**.

Новичок не заметит эти артефакты и тем более не поймет их, потому что они не будут бросаться в глаза. Но через некоторое время постоянной работы он станет обращать на них внимание, интересоваться их смыслом и изучать их методом «тыка». В этот момент пользователь за хочет приложить определенные усилия, чтобы узнать больше. Если вы предоставите ему простой способ выяснить назначение этих элементов, он не просто станет более опытным пользователем – он станет более довольным пользователем, а его власть над программой будет расти вместе с его уровнем знаний о ней.

4. Обогащенные средства ввода

Аналогичным образом монопольные приложения выигрывают от обогащенных средств ввода. Для каждого часто используемого аспекта приложения необходимо обеспечить несколько способов управления. Непосредственное манипулирование, диалоговые окна, клавиатурные сокращения – все в этом случае будет уместным. При использовании идиом непосредственного манипулирования вы можете предъявлять более жесткие требования к точности моторики пользователя. Чувствительные области на экране могут иметь размер 2×2 пиксела, потому что вы вправе исходить из предположения, что пользователь удобно устроился в рабочем кресле, его рука устойчиво расположена на столе.

Можно также разместить элементы интерфейса в углах и на границах окна программы.

Временный тип

Продукт **временного** типа приходит и уходит, предлагая одну функцию и ограниченный набор связанных с этой функцией элементов управления. Приложение вызывается при необходимости, делает свою работу и быстро исчезает, позволяя пользователю продолжить прерванную деятельность (как правило, в окне монопольного приложения).

Определяющей характеристикой временного приложения является его преходящая сущность. Поскольку оно не находится на экране в течение больших периодов времени, у пользователя **нет возможности привыкнуть к нему**. Следовательно, интерфейс программы должен быть недвусмысленным и представлять элементы управления четко и ясно, исключая ошибки или путаницу. Интерфейс должен сообщать о своих функциях. Здесь нет места красивым, но неоднозначным пиктограммам или изображениям. Как раз *здесь* кнопки должны быть большими, а надписи на них – ясными, набранными крупным и хорошо читаемым шрифтом.

Хотя временная программа, несомненно, может быть единственной запущенной программой на рабочем столе, она, как правило, играет роль **вспомогательной при монопольном приложении.** Типичным примером сценария работы с временным приложением является регулировка громкости динамиков компьютера.

Если вся компьютерная система в целом играет роль временного приложения в физическом мире атомов, не обязательно минимизировать число потребляемых пикселов и объем привлекаемого внимания. Таким свойством обладают, к примеру, мониторы слежения в производственных условиях и цифровые видеосистемы в операционной. Здесь весь экран компьютера используется лишь временами, тогда как пользователь монопольно занят механической деятельностью. В подобных случаях крайне важно, чтобы информация была внятной и легко воспринималась с расстояния в несколько метров, а это, очевидно, требует более смелого применения цвета и более щедрого распределения экранного пространства.

Принципы проектирования интерфейсов со временным типом

1. Интерфейс должен быть ярким и понятным

Хотя временная программа должна экономно использовать экранное пространство, её **элементы управления могут быть пропорционально больше, чем элементы управления монопольного приложения**. Перегруженный графикой интерфейс монопольного приложения приестся пользователю через пару недель, но временная программа находится на экране не так долго, чтобы надоесть пользователю. Наоборот, **броская графика поможет пользователю быстрее сориентироваться, когда программа появится на экране**.

Инструкции по работе с временными программами должны быть встроены в их интерфейс. Весьма вероятно, что пользователь видит окно такой программы лишь раз в месяц, так что вполне способен забыть смысл её элементов управления. Так, вместо надписи «Настройка» на кнопке была бы уместна надпись «Настройка пользовательских предпочтений», а сама кнопка, соответственно, должна быть крупнее. Конструкция «действие – объект» делает интерфейс более понятным, а результаты использования кнопки – более предсказуемыми. По аналогичным причинам во временной программе недопустимы сокращения.

Обратная связь должна быть конкретной и ясной, чтобы избежать путаницы. Скажем, пользователь должен без труда понимать, что принтер занят, а длительность только что записанного аудиоролика составляет пять секунд.

2. Интерфейс должен быть простым

После того как пользователь вызвал временную программу, **вся информация и все инструменты должны быть предоставлены ему непосредственно в единственном окне этой программы**. Удерживайте внимание пользователя на этом окне и не распыляйте основную функцию программы по служебным и диалоговым окнам. Если вы ловите себя на том, что создаете диалоговое окно или второе представление для временной программы, это верный признак того, что ваше решение требует пересмотра.

Во временных программах неуместны узкие полосы прокрутки и возня с мышью. **Требования к точности моторики пользователя должны быть минимальными.** Лучшее решение – простые кнопки для простых функций. Непосредственное манипулирование тоже может быть эффективным решением, но объекты манипулирования должны быть достаточно крупными, чтобы было легко находить их и взаимодействовать с ними. Клавиатурные сокращения также желательны, но они должны быть простыми, а все важные функции должны быть напрямую доступны в интерфейсе.

Конечно, для временных приложений возможны исключения из правила «одна функция», но они редки. **Если приложение выполняет более одной функции, интерфейс должен отражать это визуально и недвусмысленно, предоставляя мгновенный доступ ко всем функциям**, но не обрастая при этом дополнительными окнами или диалогами.

Помните, что временная программа, скорее всего, будет вызвана для поддержки некоторых аспектов монопольного приложения. Это означает, что временное приложение, расположившись на экране поверх монопольного приложения, может заслонить ту самую информацию, ради которой было вызвано. Отсюда следует, что **окно временной программы должно быть перемещаемым, то есть иметь заголовок или иной очевидный способ перетаскивания.**

Очень важно, чтобы управление временным приложением требовало минимальных накладных расходов. Пользователь хочет лишь выполнить конкретную функцию и двигаться дальше. Было бы крайне неразумно заставлять его совершать при этом какието непродуктивные действия, связанные с управлением окном приложения.

3. Временное приложение должно восстанавливать предыдущее положение и предыдущую конфигурацию.

Наиболее уместный способ помочь пользователю как с временными, так и с монопольными приложениями – наделить приложение собственной памятью. Если временная программа запомнит свое состояние при последнем использовании, велика вероятность, что и в следующий раз положение и размеры её окна окажутся подходящими. Практически всегда такое решение лучше любых значений по умолчанию. Какие бы форму и положение ни придал пользователь программе, она должна появляться именно в таком виде при следующем вызове. Конечно, то же самое справедливо и в отношении её логических настроек.

Фоновый тип

Программы, которые в нормальном состоянии не взаимодействуют с пользователем, позиционируются как **фоновые**. Они работают в фоновом режиме, невидимые и неслышные, и выполняют задачи, которые, возможно, важны, но не требуют вмешательства пользователя. Драйвер принтера или подключение к сети – вот два отличных примера.

Как можно догадаться, любое обсуждение интерфейса фоновой программы будет по естественным причинам кратким. Тогда как временное приложение управляет выполнением функции, **фоновые приложения обычно управляют процессами**. Сердцебиение – это не функция, которая требует сознательного контроля, но процесс, автономно происходящий в фоновом режиме. Подобно процессам, регулирующим сердцебиение, фоновые приложения остаются обычно совершенно незаметными, добросовестно выполняя свое предназначение, пока включен компьютер. Однако, в отличие от сердца, их требуется время от времени устанавливать и удалять, а также настраивать в связи с изменениями обстоятельств. Именно в такие моменты возникает необходимость в общении фоновых приложений с пользователем. Взаимодействие между пользователем и фоновой программой является по природе своей исключительно временным, так что здесь **действуют все правила для временных приложений.**

Следование принципам проектирования временных приложений, а именно **обеспечение информирования пользователей о назначении приложения и его возможностях**, а также **информирование о смысле выбранных значений**, в ситуации с фоновыми приложениями становится еще более критичным. Во многих случаях пользователь и не подозревает о существовании фоновой программы. С учетом этого факта становится очевидно, что сообщения от таких программ могут сбить пользователя с толку, не будучи предъявленными в соответствующем контексте. Поскольку многие из этих программ выполняют таинственные функции (как, например, драйвер принтера или концентратор соединений), поступающие от них сообщения должны быть такими, чтобы пользователи не испытывали растерянности или недоумения.

Вопрос, ответ на который считается очевидным, когда речь идет о приложениях других типов, становится принципиальным для фоновых программ: если программа невидима, как вызвать на экран её окно в тех редких случаях, когда в нем возникает необходимость? Один из наиболее распространенных подходов в системе Windows – представить фоновую программу пиктограммой в системной области уведомлений.

Размещение перед глазами пользователя пиктограммы, которой он, может быть, никогда не воспользуется, является оскорблением не меньшим, чем наклеивание рекламы на ветровое стекло автомобиля. **Пиктограммы фоновых программ постоянно должны быть перед глазами, только если предоставляют полезную информацию о состоянии этих программ.**

Эффективным подходом к настройке фоновых программ, применяемым как в Mac OS, так и в Windows, являются **панели управления**. Это программы временного типа, обеспечивающие единую точку входа для настройки служб. Важно также обеспечить прямой непротиворечивый доступ к фоновым приложениям в любой момент, когда возникает проблема, мешающая пользователю решать свои задачи. Например, если пиктограмма в области уведомлений указывает на проблему с принтером, щелчок по этой пиктограмме должен давать доступ к механизму для исправления ситуации.

Проектирование в среде интернет

Некоторые из современных приложений, работающих в браузере (часто называемых приложениями *Web 2.0*), стирают различия между настольными приложениями и веб-приложениями и даже предлагают возможность создавать новые идиомы взаимодействия, наилучшим образом поддерживающие тех людей, для которых мы проектируем. С появлением так называемых насыщенных интернетприложений (использующих такие технологии, как AJAX, Macromedia Flash, Java и ActiveX) проектирование интернет-приложений стало требовать гораздо большего внимания к тонким аспектам *поведения* продукта, чем в случае с прежними простыми веб-сайтами.

Появившаяся возможность создавать сложное поведение в браузере предъявляет к качеству проектирования взаимодействия такие же требования, как разработка самостоятельных программных приложений. Одного только внимания к внешнему виду сайта со стороны дизайнера и внимания к структуре со стороны информационного архитектора теперь уже недостаточно для того, чтобы на новом витке развития Всемирной паутины создавать эффективный и привлекательный для пользователей опыт взаимодействия.

Рассмотрим виды продуктов и услуг, предлагаемых через веб-браузер:

1. Информационные веб-сайты;
2. Сервисные веб-сайты;
3. Веб-приложения.

Информационные веб-сайты

Изначально веб-браузеры задумывались как средство просмотра опубликованных и связанных документов. Как следствие, изначально веб-среду составляли исключительно коллекции документов (или страниц*)*, известные как **веб-сайты.** Мы по-прежнему используем этот термин для описания информационных служб Интернета, взаимодействие с которыми сводится к поиску информации и переходу по гиперссылкам.

Такие веб-сайты легко придумать: набор страниц или документов, имеющих последовательную или ступенчатую иерархию, модель навигации для перехода с одной страницы на другую, а также функция поиска, обеспечивающая целеориентированный доступ к конкретным страницам. Существует множество простых веб-сайтов – персональных, созданных для нужд маркетинга и технической поддержки, а также информационных для интрасетей.

В случае таких сайтов основные вопросы проектирования связаны с дизайном, визуальной композицией, элементами навигации и структурой (информационной архитектурой). Веб-сайты, как правило, не демонстрируют сложного поведения (то есть такого поведения, где результат взаимодействия с пользователем за висит от состояния приложения), а потому им редко требуется внимание проектировщиков взаимодействия.

Типы информационных веб-сайтов

Чисто информационные сайты, которые не требуют сложных транзакций для реализации постраничной навигации и ограниченного поиска, должны балансировать между необходимостью вывода важной информации с достаточной плотностью и необходимостью предоставления новичку и нечастому гостю возможностей быстро учиться и ориентироваться на сайте. Отсюда проистекает **конфликт между монопольными и временными признаками информационных сайтов**. То, какой тип перевесит, сильно зависит от целевых персонажей сайта и от шаблонов их поведения. Приходят ли эти пользователи изредка или однократно – или же это постоянные посетители, заглядывающие еженедельно или ежедневно, чтобы просмотреть содержимое сайта?

Частота обновления содержимого сайта до некоторой степени предопределяет это поведение. Информационные сайты с постоянно обновляемой информацией, естественно, привлекут больше постоянных посетителей, чем сайты, где содержимое обновляется раз в месяц. Редко обновляемые сайты используются скорее как источники справочного материала (если информация на них не является животрепещущей), чем как ресурсы, к которым пользователь обращается постоянно. Следовательно, они должны демонстрировать поведение, типичное для временного, а не монопольного приложения. Более того, такой сайт может подстраиваться под пользователя в зависимости от частоты его посещений, проявляя признаки монопольного типа.

Признаки монопольного типа

Подробное представление информации легче реализовать в монопольном режиме. Предполагая, что пользователь развернет окно на весь экран, проектировщики могут воспользоваться преимуществами доступного пространства, чтобы ясно представить информацию, а также средства навигации и подсказки, по которым смогут ориентироваться пользователи.

Единственной проблемой при позиционировании веб-сайта как монопольного приложения является выбор оптимального разрешения экрана. Разработчикам вебсайта приходится уже на ранних стадиях проекта принимать решение о том, какое наименьшее разрешение они готовы поддерживать. Технически возможно использовать адаптивную («резиновую») верстку для гибкого отображения содержания страниц при самых разнообразных размерах окон, однако все равно необходима оптимизация под распространенные экранные размеры и наименьшее допустимое для ключевого персонажа разрешение. Качественные исследования помогут с прояснением этого вопроса: какое количество людей, похожих на персонажей сайта, работает с разрешением 800×600?

Признаки временного типа

Чем реже ключевые персонажи посещают сайт, тем более сильным должен быть его уклон в сторону временного типа. Для информационного сайта это означает простоту и прозрачность навигации и ориентирования.

Сайты, используемые от случая к случаю как справочные ресурсы, должны позволять пользователю ставить закладки на любой странице, чтобы он смог быстро вернуться на то же место впоследствии.

Если материал сайта обновляется с периодичностью от одного раза в неделю до одного раза в месяц, пользователи, вероятнее всего, будут посещать его нерегулярно, и, следовательно, система навигации должна быть предельно ясной. Если сайт сохраняет информацию о действиях пользователя с помощью cookie-файлов или серверных методов и предоставляет информацию с учетом интересов пользователя, это окажет серьезную навигационную поддержку тем, кто посещает сайт относительно редко (здесь мы исходим из предположения, что при каждом последующем посещении пользователя интересует похожая информация).

Сервисные веб-сайты

Некоторые веб-сайты выходят за рамки обычных сайтов и предлагают сервисные функции, позволяющие посетителям не только получать информацию. Классические примеры сервисных веб-сайтов – интернет-магазины и сайты финансовых служб. Как правило, они структурируются аналогично информационным веб-сайтам, но помимо информации на страницах размещены еще и функциональные элементы, обладающие более сложным поведением. В интернет-магазине эта группа элементов представлена корзиной, функциями обработки заказа и возможностью сохранять профиль пользователя. На сайтах некоторых магазинов имеются и более сложные интерактивные инструменты, такие как конфигураторы, позволяющие пользователям выбирать и уточнять параметры своих приобретений.

Чтобы реализовать адекватное поведение функционально обогащенных элементов управления, в процессе проектирования сервисных веб-сайтов необходимо уделять пристальное внимание как архитектурной организации страниц, так и проектированию взаимодействия. Разумеется, обе задачи, равно как и эффективная передача атрибутов бренда (а это бывает часто важно, учитывая коммерческую природу большинства сервисных сайтов), нуждаются в поддержке со стороны графического дизайна.

Типы сервисных веб-сайтов

Подобно информационным веб-сайтам интернет-магазины, интернет банки, инвестиционные сайты, порталы и прочие сайты должны удерживать равновесие между монопольной и временной линиями поведения. И действительно, многие сервисные сайты содержат значительные объемы информации – к примеру, онлайн-покупатели любят изучать и сравнивать продукты. В рамках этой деятельности пользователи часто уделяют значительное внимание одному сайту, но в некоторых случаях (как при выборе наилучшего предложения) они просматривают несколько сайтов подряд. Для таких сайтов крайне важна **простота навигации, а равно наличие доступа к сопутствующей информации и эффективность транзакций.**

Поисковые машины и порталы вроде Google и Yahoo! представляют собой особую разновидность сервисного сайта; их назначение – обеспечивать навигацию по другим веб-сайтам, а также доступ к подборкам новостей и информации из многочисленных источников. Очевидно, что поиск и переход на сайты из результатов поиска – деятельность, носящая временный характер, однако для агрегации информации на портале вроде Yahoo! при реализации иногда требуется использовать монопольный подход. Мимолетность общения пользователей с временными функциями сервисных сайтов делает особенно важной минимизацию действий, связанных с навигацией. Возникает соблазн разбить информацию и функции на несколько страниц, чтобы снизить время загрузки и визуальную сложность (похвальное стремление), однако следует избегать путаницы и помнить, что аудитория устает щелкать мышью. В важном юзабилити-исследовании 2001 года, проведенном User Interface Engineering и посвященном тому, как пользователи воспринимают время загрузки страниц сайтов электронной коммерции, таких как Amazon.com и REI.com, выяснилось, что *восприятие пользователем продолжительности загрузки страницы в большей степени зависит от того, достигает ли пользователь своих целей, нежели от реальной продолжительности загрузки*.

Веб-приложения

Веб-приложения насыщены взаимодействием и демонстрируют сложное поведение – они во многом похожи на серьезные настольные приложения. И хотя некоторые веб-приложения сохраняют постраничную модель навигации, их страницы скорее похожи на *представления*, чем на веб-документы. Хотя многие подобные приложения по-прежнему ограничены архаичной моделью запрос/ответ (которая требует, чтобы пользователь вручную «отправлял» каждое изменение состояния), существующая технология уже поддерживает асинхронный обмен с сервером и локальное кэширование данных, что позволяет приложению в браузере вести себя во многом подобно сетевому настольному приложению.

Такие веб-приложения во многом могут предъявляться пользователям **как настольные приложения** – просто работающие внутри браузера. Возникающие при этом шероховатости не будут значительными, если взаимодействия тщательно спроектированы с учетом технологических ограничений. И хотя задача проектирования и создания насыщенного и оперативного взаимодействия, работающего в различных браузерах, определенно может быть сложной, Интернет как платформа замечательно подходит для создания инструментов, способствующих и оказывающих поддержку совместной работе. Помимо прочего, веб-браузер является еще и хорошим средством предоставления нечасто используемой функциональности, поскольку не требует установки исполняемых файлов на компьютер пользователя. И, разумеется, веб приложение дает пользователям **возможность работать со своей информацией и нужными функциями в любом месте**, где есть доступ к сети Интернет. Это не всегда уместно или необходимо, однако, учитывая современную мобильность сотрудников компаний и растущую популярность удаленной работы, такая возможность может представлять собой определенную ценность, позволяя людям получать доступ к одним и тем же инструментам и функциям с различных компьютеров.

Типы веб-приложений

Веб-приложения, как и настольные приложения, могут быть монопольными или временными, однако поскольку веб-приложениями мы называем продукты со сложной и богатой функциональностью, то они по определению **тяготеют к монопольному типу**.

Монопольные веб-приложения стремятся доставлять информацию и функциональность так, чтобы наилучшим образом поддерживать сложную деятельность человека. Часто это требует создания функционально насыщенных высоко-интерактивных пользовательских интерфейсов.

К проектированию монопольных веб-приложений лучше подходить так же, как к проектированию настольных приложений, забыв об информационных и сервисных веб-сайтах с постраничной навигацией. Проектировщикам также требуется четкое понимание технических ограничений среды и пределов возможного в рамках имеющихся у разработчика времени и бюджета. Подобно монопольным настольным приложениям, большинство монопольных веб-приложений должны быть полноэкранными, плотно заполненными информационными и функциональными элементами и должны использовать специализированные панели или другие экранные области для группировки родственных функций и объектов. У пользователей должно возникать ощущение среды, а не постоянных переходов от страницы к странице или с места на место. Следует минимизировать видимую перерисовку экрана (на обычных веб-сайтах практически каждое действие требует полного обновления страницы).

Преимущество позиционирования монопольных веб-приложений как настольных приложений, а не наборов веб-страниц, состоит в том, что проектировщики получают возможность вырваться из страничной модели взаимодействия в браузере и работать со сложными вариантами поведения, столь необходимыми веб-приложениям.

Одно из преимуществ подачи важной для корпораций функциональности с помощью браузера заключается в том, что в случае грамотной реализации такой подход позволяет пользователям в любой момент работать с редко требуемой информацией и функциональностью, **не прибегая к установке на компьютер всех инструментов, которые только могут понадобиться**. Рутинная задача, выполняемая лишь раз в месяц, или внезапно возникшая необходимость сгенерировать отчет – именно такая работа хороша для временных веб-приложений.

Проектируя временные веб-приложения, как и любые другие временные приложения, крайне важно обеспечить прозрачную навигацию и ориентирование. Помните также, что временное приложение одного пользователя может быть монопольным приложением другого. Про анализируйте как следует, насколько совместимы наборы потребностей этих двух пользователей – часто бывает, что корпоративное веб приложение обслуживает широкий спектр персонажей, так что для доступа к одной и той же информации требуется целый ряд пользовательских интерфейсов.

В результате непрерывного развития сети Интернет родились две удивительных особенности: мгновенный доступ к невероятным объемам информации и простота организации совместной работы.

Другой отличный подход – отказаться от браузера полностью и создавать *интернет-приложения*. Разработка приложения на стандартной для персонального компьютера платформе, такой как .NET или Java/ Swing, с использованием стандартных протоколов сети Интернет обеспечивает богатые, четкие, сложные взаимодействия, сохраняя возможность обращаться к данным Всемирной паутины. Развитие протоколов доставки данных, таких как RSS, и интерфейсов прикладного программирования вебприложений позволяет продуктам извлекать из веб-среды всю ту же информацию, которая доступна браузеру, но быть гораздо более приятными для пользователей благодаря возможностям, доступным только «родным» для персонального компьютера приложениям.

Хороший пример такого приложения – Apple iTunes. Программа позволяет покупать и загружать музыку и видео, получать информацию о компакт-дисках, делать музыку доступной через сеть Интернет – и все это посредством пользовательского интерфейса, оптимизированного для подобных действий способом, который вряд ли возможно реализовать в веб-браузере.

Другой пример такого подхода – системы архивации и передачи изображений, используемые радиологами для просмотра снимков пациентов, в частности снимков, полученных магнитнорезонансным сканированием. Такие системы позволяют радиологам быстро работать с сотнями изображений, масштабировать конкретные снимки и выполнять коррекцию снимков для более точной идентификации различных типов тканей. Очевидно, такого рода взаимодействие плохо подходит для реализации в браузере. При этом возможность просматривать изображения из любого места может оказаться очень полезной для радиолога, если он, скажем, находясь в крупном исследовательском медицинском центре, консультирует деревенскую больницу, где нет специалистов, способных диагностировать определенные

заболевания. Поэтому во многих системах архивации изображений применяются интернет-протоколы, обеспечивающие возможности удаленного просмотра и совместной работы.

Прочие платформы

Программа, работающая на персональном компьютере, располагает роскошью получать внимание пользователей, когда ей это необходимо. Проектирование взаимодействия для продуктов, работающих в мобильных и публичных контекстах, требует особого внимания, так как окружающая эти продукты обстановка реального мира полна событий и шумов. **Портативные устройства, киоски и прочие встроенные системы** (в телевизорах, микроволновых печах, автомобильных приборных панелях, фотоаппаратах, банкоматах, лабораторном оборудовании) – уникальные платформы, каждая из которых имеет свои возможности и ограничения. Бездумно наделяя устройства и приборы цифровыми мозгами, мы рискуем создать вещи, которые будут вести себя скорее как компьютеры, чем как удобные продукты, которые нравятся пользователям.

Общие принципы проектирования

Хотя встроенные системы (физические устройства с интегрированными программными системами) и могут предлагать типичное для программ взаимодействие с пользователем, они обладают уникальными особенностями, которые отличают их от систем, работающих в настоль ном компьютере. При проектировании любой встроенной системы, будь то интеллектуальный бытовой прибор, киоск или портативное устройство, не забывайте следующие основополагающие принципы:

1. Не думайте о продукте как о компьютере
2. Объединяйте проектирование аппаратной и программной частей
3. Позвольте контексту определять направление проектирования
4. Используя режимы, делайте это взвешенно
5. Ограничивайте функциональность
6. Выдерживайте баланс между навигацией и плотностью отображаемой информации
7. Минимизируйте сложность ввода

Рассмотрим каждый из принципов подробнее.

Не думайте о продукте как о компьютере

Пожалуй, самое важное правило для разработчика встроенных систем заключается в том, что он создает *не* компьютерную систему, хотя в её интерфейсе, возможно, преобладает дисплей, аналогичный компьютерному. У пользователей могут быть вполне конкретные ожидания относительно возможностей продукта (если это бытовой прибор или иное известное им устройство) либо, наоборот, не слишком большие ожидания (если это киоск в общественном месте). В такой ситуации последним делом будет переносить весь профессиональный багаж (идиомы и терминологию) из мира персональных компьютеров на «простенькое» устройство вроде фотоаппарата или микроволновки. Аналогично этому пользователи научного или иного технического оборудования ожидают получить быстрый и непосредственный доступ к информационным и функциональным элементам тем способом,

который им привычен, не продираясь в поисках необходимого сквозь дебри операционной системы или файловой системы.

Программисты (особенно те, кто имеет опыт проектирования для настольных платформ) легко забывают, что, хотя продукт, который они разрабатывают, является программным продуктом, он предназначен не для компьютера, типичными характеристиками которого являются большой цветной экран, высокая производительность, необъятная память, полноценная клавиатура и мышь. У большинства встроенных систем ничего подобного нет. И, что еще важнее, эти продукты используются в совершенно иных контекстах, чем настольные компьютеры.

Идиомы, ставшие привычными на персональных компьютерах, совершенно неуместны для встроенных систем. «Отмена» – неподходящее название кнопки, выключающей микроволновку, а требовать от людей входа в режим «настройки», чтобы изменить температуру нагрева, попросту абсурдно. Чем втискивать компьютерный интерфейс в форм фактор устройства с маленьким экраном, лучше вникнуть в то, для чего создано устройство и как цифровая технология может быть приложена к нему, чтобы облегчить пользователям обращение с этим устройством.

Объединяйте проектирование аппаратной и программной частей

С точки зрения взаимодействия с пользователем отличительной характеристикой встроенных систем является тесное переплетение аппаратных и программных составляющих интерфейса. В отличие от настольных компьютеров, где все внимание пользователя сосредоточено на большом цветном экране с высоким разрешением, большинство встроенных систем предлагают аппаратные элементы управления, на которые направлено внимание пользователя и которые должны точно соответствовать его целям. Стоимость производства, производительность и формфактор накладывают ограничения, в силу которых аппаратные элементы управления и навигации часто используются вместо экранных эквивалентов. Следовательно, они должны быть подстроены под требования программной составляющей интерфейса и в то же время со ответствовать требованиям эргономики и целям пользователя.

Таким образом, разрабатывать программные и аппаратные компоненты интерфейса системы (и взаимодействие между ними) следует одно временно и с учетом эргономики и целей пользователя, а также соображений эстетического плана. Многие из лучших, передовых цифровых устройств, существующих сегодня (например, TiVo, iPod), проектировались именно в соответствии с таким целостным подходом – их аппаратная и программная части сочетаются настолько хорошо, что опыт, получаемый пользователями, оказывается привлекательным и эффективным. Это редко происходит в ходе стандартного процесса разработки, когда команда конструкторов аппаратной части периодически передает команде разработчиков программной части за конченные механические решения и решения промышленного дизайна, готовые к производству, а уж разработчики программ должны подстраиваться под эти решения без учета интересов пользователей.

Позвольте контексту указывать направление проектирования

Еще одним принципиальным отличием встроенных систем от настольных приложений является значение, которое имеет для них контекст среды. Хотя вопросы контекста иногда возникают и при разработке настольных приложений, проектировщики могут предполагать, что большинство настольных приложений будут выполняться на стационарном компьютере, расположенном в относительно тихом и уединенном месте. По мере того как портативные компьютеры догоняют по мощности и коммуникационным возможностям настольные

системы, это предположение все чаще нарушается, но в целом остается справедливым, поскольку даже с ноутбуком человек старается найти для работы спокойное и малолюдное место.

В отношении многих встроенных систем верно ровно обратное. Эти системы разрабатываются либо для работы на ходу (в портативных устройствах), либо для стационарных применений в общественных местах (в информационных киосках). Даже самые стационарные и уединенные встроенные системы (домашние электроприборы) имеют ярко выраженный контекст. Хозяйка, расставляющая тарелки с горячей едой, пока гости ждут в другой комнате, постоянно отвлекается от элементов управления «интеллектуальной» микроволновки. Навигационные системы в приборных панелях автомобилей не должны иметь программируемых кнопок, изменяющих свое назначение в различных контекстах, поскольку водителю придется отрывать взгляд от дороги, чтобы прочитать подпись. От работника в цехе нельзя требовать, чтобы все его внимание было сосредоточено на расшифровке элементов управления оборудования. В некоторых случаях отвлечение внимания от основной деятельности может представлять угрозу для жизни.

Таким образом, проектирование встроенных систем должно очень четко соответствовать контексту их использования. У портативных устройств этот контекст определяется тем, как и где (физически) пользователь работает с устройством. Как он его держит? Одной рукой или двумя? Где оно находится, когда не используется? Чем еще занимается пользователь во время работы с устройством? В каком окружении оно используется? Там шумно? Светло или темно? Что чувствует пользователь этого устройства, когда работает с ним на глазах множества людей? Некоторые из этих вопросов мы более подробно рассмотрим далее.

Для киосков вопрос контекста сводится к окружению, в котором он находится, и к некоторым социальным моментам. Какую роль играет киоск в данном окружении? Находится ли он на пути людского потока? Является ли он источником справочной информации или же представляет интерес сам по себе? Способна ли архитектура окружения естественным образом направить людей к киоску, когда это требуется? Можно ли оценить, сколько людей будут пользоваться киоском одновременно? Достаточно ли киосков, чтобы удовлетворить запросы пользователей, не создавая длинных очередей? Достаточно ли свободного пространства вокруг киосков, не мешают ли они людскому потоку?

Используя режимы, делайте это взвешенно

Как правило, приложения в настольных компьютерах имеют множество рабочих режимов. Программа может находиться в разных состояниях, в которых элементы управления и ввода данных могут иметь различные линии поведения. Хорошей иллюстрацией сказанного являются палитры инструментов в Photoshop: выберите инструмент – и ваши манипуляции с мышью и клавиатурой будут связаны с набором функций выбранного инструмента. Выберите другой инструмент – и реакция устройств ввода на аналогичные манипуляции будет иной.

К сожалению, пользователей очень легко сбить с толку с помощью не достаточно очевидного режима поведения. Поскольку самостоятельные устройства обладают обычно дисплеями меньшего размера и ограниченными механизмами ввода, очень сложно сообщить пользователю, в каком режиме находится продукт, а для смены режима часто требуются сложные навигационные действия. Возьмем для примера мобильные телефоны. Обычно режимы организованы иерархически, и для доступа к конкретному режиму приходится нажимать огромное количество кнопок. Большинство владельцев сотовых телефонов используют только функцию набора номера и телефонную книгу, и многие из них сразу теряются, когда возникает необходимость в доступе к другим функциям. Даже такая важная

функция, как выключение звонка, часто выходит за рамки умений среднего пользователя телефона.

При проектировании встроенных систем важно ограничивать число режимов, а переключение между ними в идеальном случае делать естественным следствием изменений контекста. Например, в смартфоне переключение в режим телефона должно происходить, когда поступает звонок, а по окончании разговора устройство должно возвращаться в предыдущий режим. (Ответ на звонок при работе с другими данными – предпочтительная альтернатива.) Если работа в нескольких режимах действительно необходима, они должны быть недвусмысленно пред ставлены в интерфейсе, а путь выхода из каждого режима должен быть столь же ясен. Четыре аппаратных кнопки в большинстве наладонных компьютеров под управлением Palm OS – хороший пример четкого обозначения режимов.

Ограничивайте функциональность

Большинство встроенных систем используются в конкретных ситуациях с конкретными целями. Боритесь с искушением превращать их в универсальные компьютеры. Устройства с ограниченным набором функций обслужат пользователей лучше, чем устройства, пытающиеся совместить много несопоставимых функций в одном корпусе. Такие устройства, как КПК под управлением Microsoft Windows Mobile, в последнее время претендующие на эмуляцию полноценных настольных систем, рискуют отпугнуть пользователей громоздким интерфейсом, который под завязку набит функциями, включенными в него только потому, что они существуют в настольных системах. Многие из нас полагаются на свои «умные» телефоны (такие как Treo, BlackBerry), но думаю, большинство согласится, что спектр функций этих устройств до некоторой степени снижает их пригодность как телефонов.

Многие устройства способны обмениваться информацией с компьютерами. Имеет смысл подходить к проектированию таких систем, ориентируясь на персональный компьютер: устройство является *спутником* настольного компьютера, его расширением и предоставляет пользователю важнейшие данные и функции в таких контекстах, где использование настольного компьютера невозможно. Сценарии помогут вам определить, какие именно функции действительно необходимы в системахспутниках.

Выдерживайте баланс между навигацией и плотностью отображаемой информации

Многие самостоятельные устройства имеют ограниченное экранное пространство. Это является следствием стоимости аппаратных составляющих, формфактора, соображений мобильности или потребления электроэнергии, но, каковы бы ни были причины этого, проектировщикам следует оптимально использовать доступную технологию вывода информации для удовлетворения потребностей пользователей. При создании встроенных систем с ограниченным экранным пространством ценен каждый пиксел, каждый сегмент и каждый квадратный миллиметр экрана. Такие ограничения по площади дисплея почти всегда требуют компромисса между ясностью информации и сложностью навигации. Разумно ограничивая набор функций, вы можете до определенной степени смягчить ситуацию, но конфликт между выводом информации и навигацией существует почти всегда.

Вы должны тщательно продумывать вывод на дисплей во встроенных системах, создавая иерархическую информационную структуру. Определите, какая информация наиболее важна для пользователя, и сделайте соответствующую функцию самой заметной в интерфейсе. Затем выясните, какая дополнительная информация еще может поместить ся на экране. Постарайтесь избежать переключений между различными группами данных, при которых экран мигает. Например, микро волновая печь с цифровым управлением может

переключаться между температурой, до которой следует выполнять нагрев, и текущей температурой. В этом случае очень легко перепутать, где какая темпера тура. Более удачным решением будет вывод конечной и текущей температур при помощи столбцовой диаграммы, показывающей, насколько текущая температура близка к желаемой. На дисплее должно еще оставаться место для отображения состояния соответствующих аппаратных элементов управления, а еще лучше будет, если эти элементы сами будут демонстрировать свое состояние. Для этой цели подходят кнопки со встроенными светодиодами, двух и многополюсные выключатели, ползунковые регуляторы, рукоятки.

Минимизируйте сложность ввода

Почти все встроенные системы оборудованы упрощенным механизмом ввода, а отнюдь не клавиатурой и устройством графического ввода. Это означает, что ввод любой (особенно текстовой) информации в сис тему неудобен, выполняется медленно и труден для пользователей. Даже самые изощренные из механизмов ввода – сенсорные экраны, устройства голосового ввода и распознавания рукописного текста, встроенные клавиатуры – неудобны по сравнению с полноразмерной клавиатурой и мышью. Таким образом, ввод должен быть максимально ограничен и упрощен.

Устройство BlackBerry от компании RIM эффективно задействует колесико в качестве основного механизма выбора. Быстрая прокрутка колесика перебирает варианты, а нажатие на него (или на соседнюю кнопку) приводит к выбору текущего пункта. В устройстве используется и встроенная клавиатура для ввода текстовых данных. В Palm Treo, для сравнения, используется сенсорный экран и встроенная клавиатура. Такой подход был бы эффективен, если бы экран Treo позволял активизировать любую функцию прикосновением пальца. Однако большинство элементов управления на экране Palm столь малы, что приходится использовать перо. Это означает, что пользователю приходится переключаться между пером и сенсорным планшетом, и в результате ввод становится неудобным. В поздних устройствах компании Palm проблема решается расположенным между сенсорным экраном и встроенной клавиатурой «джойстиком» с двумя осями и кнопкой, что позволяет выполнять навигацию и активировать элементы управления на экране, не прикасаясь к экрану.

В информационных киосках экраны обычно крупнее, но и там тоже следует по возможности отказываться от текстового ввода. Сенсорные экраны, если позволяет размер, могут отображать виртуальные клавиатуры, но каждая клавиша должна быть достаточно велика, чтобы пользователю было сложно опечататься. В сенсорных экранах, кроме того, следует избегать идиом, связанных с перетаскиванием. Простые идиомы прикосновения к объекту легче поддаются контролю и более очевидны для новичков (при наличии ярко выраженного назначения).