Создавая новое поколение - часть 4
Удобство использования
Николас Блэхфорд (Nicholas Blachford), Воскресенье, 14 Ноябрь 2004, 19:39

За годы я использовал множество операционных систем, и очевидно, что в некоторых системах удобство использования^{*} понято лучше, чем в других. Также ясно, что если вы хотите понять его лучше, то в вашей команде должны быть люди, которые будут заниматься только этим. Мне еще не приходилось видеть, чтобы ОС (как платная, так и бесплатная), ведомая программистами, показывала такой же уровень удобства в использовании, как в компаниях, в которых есть люди, занимающиеся этим.

Парадоксально, но практичность - непростой вопрос. В ней нет математическогообоснования или расчетов, так что не существует единого правильного или неправильного пути в ее достижении. Мне она видится как нечто между искусством и наукой, что-то вроде расставления правильных вещей по нужным местам, не переусложняя их, упрощая там, где это необходимо, и перемещая более сложные или менее используемые опции подальше от взгляда простых пользователей.

Требования и предпочтения разработчиков системы зачастую значительно отличаются от требований и предпочтений тех, кто будет ее использовать. Сами разработчики - отличный от обычных пользователей сорт людей, им нравится технические штуковины, они любят все контролировать и иметь столько параметров конфигурировани сколько возможно, чтобы иметь доступ к разным оптимизациям; также, чаще всего, у них хорошая память к деталям.

Пользователи же имеют тенденцию быть полной противоположносю такого поведения, но это не означает, что они глупее. Такой "" может однажды провести вам операцию на мозге. Однако, из-за этих различий разработчики ПО, как правило, не представляют собой хороших разработчиков удобных программ, они могут сделать их удобными для себя, но это может означать, что они будут совершенно непригодны для простых пользователей.

"Программисты делают свою работу, но единожды, в то время как пользователи вынуждены впоследствии с ней управляться." - Джеф Раскин.

Не бывает "" удобства, так как никто не работает одинаково. Попытка навязать пользователям метод работы натолкнется на сопротивление. Вспомните крики некоторых пользователей Gnome, когда он перешел на пространственны Nautilus. Пространственны браузер не очень хорошо подходит для для файловой системы со значительной вложенностью, однако, многие пользователи сталкивались именно с этим и постоянно имели проблемы с системой.

Все пользователи, даже разработчики, привыкают к каким-либо способам управления. Даже если есть что-то новое и лучшее, тот факт, что они привыкли к старым методам означает, что всегда будет сопротивление переменам. Именно по этой причине и только благодаря ней последние изменения в Gnome вызовут шквал жалоб.

Конечно, преимущество создания новой платформы в том, что у нас нет существующих

пользователей, которые могли бы жаловаться на изменения. Конечно, это означает, что миграция пользователей с других платформ будет проходить с некоторыми трудностями, но это случится в том случае, если именно это вы и хотите сделать, в чем цель создания новой платформы, неужели сделать то же самое, что и другие?

Больше или меньше настроек

Один из подходов в улучшении практичности - упрощение всего путем удаления параметров. Такая стратегия подразумевает, что пользователи идиоты и, по моему мнению, скорее расстроит пользователей отсутствием настроек, которые помогли бы в удобстве использования. Только то, что кто-то не знает о том, как работают внутренности компьютера не означает того, что он идиот. Практичность - это не "".

Удаление настроек или гибкости в пользу этого может делать что-то проще в использовании, но это также ведет к неполноценностипродукта. С другой стороны, добавление слишком многих настроек только смущает пользователей.

Gnome и KDE, соответственно, следуют этим принципам и в результате Gnome раздражает некоторых опытных пользователей (в частности, это послужило причиной создания проекта GoneME), в то время как KDE пугает простых пользователей (по крайней мере "Центр управления" точно).

OS X имеет множество настроек, но их не выкидывают вам все сразу, многие прячутся за кнопками "advanced options" или подобными им, она одновременно умеет быть и мощной, и простой в использовании. Различные компании и проекты копировали Mac годами просто потому, что он хорошо смотрится, но похоже никто не смог даже достичь того же уровня удобства использования.

Иногда добавление сложностей в действительност помогает практичности. Попробуйте поработать со звуком на синтезаторе начала 90-х, а потом попробуйте сделать то же на синтезаторе 70-х. Синтезаторы 90-х использовали минималистичныйподход, все лишь несколько кнопок для управления широким спектром различных настроек, их было просто ужасно неудобно использовать. Машины 70-х выносили все параметры на лицевую панель и позволяли иметь прямой доступ ко всему. На первый взгляд они были отпугивающе сложны, но, потратив время на изучение, управляться с ними было очень легко. Сегодняшние "" синтезаторы вернулись назад к методам управления из 70-х.

<hr>
* - Одной из причин задержки с переводом этой статьи стал такой замечательный термин, как "usability". Адекватного перевода ему на русский, именно как **термину**, я так и не нашел. Поэтому перевод осуществляется по обстоятельствам, согласно информации с usability.ru - примечание переводчика.
Вернуться .
[newpage]
Руководства по человеческим интерфейсам (Human Interface Guidelines)
За годы было опубликовано множество руководств по человеческим интерфейсам, и существуют несколько основных правил и руководств (например, сохранение единообразия). Однако, как я уже здесь говорил, не существует единственного "правильного ", и если забыть

это, то даже эксперты в области пользовательски интерфейсов не соглашаются друг с другом - вам не придется много читать, чтобы увидеть сочетание "эксперты не ".

Здесь я вкратце обсуждаю некоторые специфичные идеи человеческих интерфейсов, однако, еще больше написано не здесь, и я включил напоследок несколько ссылок для дальнейшего изучения. Там же есть ссылка на "Зал позора пользовательски ", в котором представлено множество примеров того как не надо делать интерфейсы. Есть и "Зал ", но он значительно меньше.

Руководства ROS
ROS (Ruby Operating System) [ROS] использует ультраминималисичный подход к руководствам по удобству интерфейсов - их всего четыре. Соответственно им, пользовательски интерфейс должен быть:

Простым в

Простым для запоминания
Удовлетворяющим и
Несмотря на обидный недостаток подробностей, весьма легко найти приложения или системы, которые нарушают одно или более из этих правил.

Проксимальный интерфейс
Одной из самых функциональных, хоть и явно мало используемых, концепций в проектировании практичных интерфейсов является проксимальный интерфейс (Proximal Interface) [Proximal]. Идея его в том, чтобы построить интерфейс, который в результате становится невидимым, пользователь просто решает свои задачи, а интерфейс этому способствует.

Используется идея прямого управления, так что вместо выбора объекта и дальнейшего выбора опции из меню вы можете сделать некоторые операции прямо захватом или перемещением объекта, количество параметров может быть увеличено с помощью клавиш-модификаторов. Принцип "тащи-" (drag and drop) - прекрасный пример прямого управления.

Проксимальный интерфейс, конечно же, включает в себя некоторый набор принципов проектирования пользовательски интерфейсов, среди них:

Принцип ступенчатости усилий - самые основные или самые важные параметры должны быть доступны легко и напрямую.
Различение объекта над которым находится мышь и точки куда конкретно она нажимает для обеспечения различных прямых операций.
Принцип левой руки - использование левой руки для уточнения операций (например, левый Shift и нажатие мыши)
Считать действие мышью процессом, например, во время этого процесса могут происходить другие
Принцип визуальных указателей - мы запоминаем, где располагаем элементы, а система не должна их реорганизовыват без серьезных на то причин.
Принцип необратимости - уничтожение данных (например, удаление) должно требовать

больших усилий, нежели содание.

Я думаю, что мы нескоро увидим массовые системы, построенные на этих принципах, хотя некоторые приложения уже следуют похожим техникам (например, компонент векторной графики в Gobe Productive для BeOS). Сама BeOS насквозь пропитана принципом тащи-кидай.

Организация интерфейса - закон Фитта

Практичность - это не только возможности и организация программ, она также применима к расположению элементов на экране. Часто упоминают закон Фитта как закон проектировки внешнего вида интерфейса [Fitts], однако в оригинале он не имел ничего общего с компьютерами.

Разработанный в 1954 году, закон Фитта применим к случайному выбору целей в одном измерении при использовании человеческой руки. Он определяет связь между размером цели, расстоянием до цели и скоростью достижения цели.

Взаимодействие с компьютером значительно отличается от этого, оно происходит в двух измерениях, основные движения не случайны и присутствует ускорение мыши. Удивительно, каким образом кто-то пришел к применению закона Фитта к компьютерам.

Некоторые используют закон Фитта и располагают меню вверху экрана, в противоположнось расположению прямо на окнах. Разумное обоснование находится в том, что, несмотря на то, что меню становится значительно дальше, оно становится практически "" размеров, так как оно находится вверху экрана. Оно может быть неограниченно высоко, но пункты меню не бесконечно широки, реальная цель в которую надо попасть не становится больше от того, что она расположена на краю экрана, а не в окне. Закон Фитта предположил бы, что меню должны быть все-таки на окне, так как ваша мышь скорее всего так будет ближе к ним (здесь предполагаем применение закона Фитта в двух измерениях).

Также здесь не учитывается ускорение мыши, которое делает прицеливание в отдельную точку на краю экрана практически невозможным с любой значительной дистанции (за исключением углов, где ваш курсор однозначно останавливается). Но, несмотря на это, помещение меню наверх все же выглядит лучше, как же так получается?
[newpage]
Я могу только предположить, что здесь вступают в игру другие факторы:

Имеет влияние мускульная память, так как меню располагаются в одном месте для всех
Ускорение мыши позволяет очень быстро перемещать указатель к вершине экрана.
Как только мышь доходит до вершины, целеуказание становиться только горизонтальным, контролировать мышь в одном измерении проще, чем в

Добраться до верха экрана это всего лишь резкое движение кистью, оно приводит вас к приблизительнойобласти. Как только вы добрались до нее, дойти до необходимого пункта меню уже очень быстро. Это двухступенчатыйпроцесс и закон Фитта на самом деле надо применять к обоим, раздельно.

Если меню находится не вверху, то прицеливание приходится вести в двух измерениях, а это гораздо сложнее. Попробуйте быстро нарисовать точную окружность, это очень сложно.

Попробуйте быстро нарисовать окружность с помощью мыши, это еще сложнее и ускорения здесь только ухудшают ошибки. Прицеливаться мышью в двух измерениях достаточно проблематично, но ускорения означают, что вы скорее промахнетесь, а это делает прицеливание еще сложнее.

Это не означает, что расположение меню или элементов управления на окнах является чем-то непригодным к использованию (вы же можете нажать на веб-ссылку, не так ли?), но если у вас стоит большое ускорение мыши, то добираться до них вы будете дольше, чем до края экрана.

Однако, и здесь все не так просто, вполне возможно, что при низкой скорости/ускорении мыши лучше будет разместить меню на окне - попробуйте поперемещать меню и изменять скорость мыши, на низкой скорости перемещение через весь экран - непростая задача.

Я не верю в то, что закон Фитта есть всё и навсегда, как его многие описывают, но это все же полезное руководство, хотя надо принимать во внимание различия между оригинальным законом и интерфейсами современных компьютеров, наравне с индивидуальныминастройками пользователей.

Практичность везде
Удобство использования касается не только GUI, оно должно применяться ко всему, даже к командной строке. Английские (или других естественных языков) команды использовались со времен MS-DOS, и, возможно, даже до этого. У IBM в течение десятилетий была слегка замудренная, но очень простая в использовании оболочка. Однако, похоже, что концепция удобства использования была полностью проигнорированаосновными интерпретаторам командной строки Unix, которые изобилуют причудливыми сокращениями, понятными их создателям, но непостижимыми для простого пользователя. Что же на самом деле означает "Grep"?

Командная строка - очень полезный инструмент и во многих случаях лучше подходит для задач, которые сложно нормально представить в GUI. Однако, нет причин, по которым она должна быть сложна настолько, чтобы требовать справочника при использовании, она должна быть доступна всем пользователям. Опыт других платформ показал, что это вполне возможно при наличии хорошо спроектированно схемы названий.

Для нашей новой платформы я бы хотел увидеть шелл с мощью bash, но с такими командами, как "list", "remove", "search" и так далее. Также в нем должна быть подробная справочная система по использованию команд. Система должна по умолчанию загружаться с GUI, так что это может использоваться для расширения возможностей консоли, показа параметров, помощи и так далее. Пользователи Unix, возможно, будут смеяться над идеей загрузки системы с GUI по умолчанию, но многие другие настольные системы делали это в течение длительного времени без каких-либо негативных последствий. То, что X может нестабильно работать не означает, что другие графические системы страдают тем же.

Еще одна вещь, которая не понравится чокнутым юниксоидам - избавление от чувствительност к регистру [Case], это еще один пережиток прошлого, который не имеет никакой пользы, но ведет к потенциальной путанице, так зачем он нам? Google и другие поисковики по умолчанию нечувствительныk регистру, и их было бы невозможно

использовать, если бы они были таковыми. Однако, я бы оставил хранение регистра.

В стуктуре файловой системы должна применяться разумная схема наименований, можно начать ее организацию с записей:

/System - файлы операционной
/Applications - приложения и библиотеки сторонних
/Home - все пользовательски файлы сю

Пользовательски и системные файлы разделены для упрощения резервного копирования, чтобы сделать копию или переместить свои файлы необходимо всего лишь копировать "/Users/My Name" в желаемое место (что еще я бы исправил - возможность использования пробелов в именах файлов).

Когда пользователь скачивает приложение, оно само и его библиотеки должны автоматически располагаться в соответствующей области для приложений, пользователь должен это видеть вместе с глобальными настройками приложения (например, тип приложения, кто его может использовать и так далее), так чтобы он мог на это влиять.
[newpage]

Установка программ

Установка программ должна быть быстрой и безболезненной операцией, инсталлятор должен предполагать, что у пользователя нет подключения к Интернету. Если приложению необходимы библиотеки, которые не включены в систему, то они должны поставляться вместе с установочным дистрибутивом, будь то CD, DVD, zip или бинарный файл. Ни при каких обстоятельствах система (и в особенности пользователь) не должна отправляться на поиски дополнительных файлов или "". Это может считаться нормой, если вы поставляете ПО для чокнутых, которые любят все делать своими руками, но это не так, если вы также хотите захватить простых пользователей.

Пользователю и программам должно быть запрещено изменение системных компонент. Изменение компонент ведет к проблемам вроде "DLL hell", когда установленная программой системная библиотека может вызвать сбой в программах или даже самой системе. Злонамеренные программы также могут в это включиться и заменять библиотеки, вызывая проблемы. Предотвратив изменение компонентов ОС пользователем и программами, любая устанавливаемая программа может ожидать однозначное окружение, в котором она будет запускаться и для которого ее можно тестировать. Как я уже говорил в предыдущей части этой серии статей, даже если кто-то сможет изменить компонент системы, система автоматически заменит его.

Конечно, DLL hell может случиться и с привлечением сторонних библиотек. Этого можно избежать, позволяя иметь несколько файлов библиотек с одинаковыми именами, но разными версиями. Если одной программе нужна версия X, она получит версию X, в то же время другая программа, требующая версию Y, сможет получить Y без каких-либо конфликтов (это реализовано в Microsoft .NET). Также, система должна заниматься поиском библиотек, дабы обнаружить ситуацию, если они будут размещены в неверном месте.

Практичная разработка

Практичность должна применяться даже к разработке. Программировани для компьютеров

не должно быть черной магией, практикуемой только посвященными.

Многие из различных платформ, существовавших в 1970-80 имели в своей поставке какую-либо версию BASIC и многие программисты начинали с них, однако, это пример хорошей идеи, которая была заброшена. Я бы сделал то же самое и включил язык для пользователей с примерами и инструкциями, которые помогли бы им начать.

Должен существовать "" язык, который можно будет использовать для всего: от простых поигрываний, скриптов для консоли и приложений до, собственно, полноценных приложений. Конечно, стандартный язык не должен исключать использование других языков, но наличие стандартного языка должно поощрять разработку и, если он будет достаточно простым, то он может даже поспособствоват непрограммистампопробовать его. Я скорее всего использовал бы Python в качестве стандартного языка, так как у него и была особая цель сделать программировани проще, и его можно использовать для всех упомянутых мной целей.

Нет логики в наличии различных языков в различных частях системы, если нет сильной в этом нужды, например, в поставке не должно быть других скриптовых языков для консоли. По моему мнению будет очень полезно иметь возможность программироватьразличные части системы на едином и простом в использовании языке. Опять же, это всего лишь "по умолчанию", все равно будет возможно установить и использовать другие языки.

Конечно же, должен быть и стандартный "производительны" язык, такой как C++ или Objective-C для нормальных разработчиков, на котором, как я думаю, будет реализовываться основное число приложений. Я бы также хотел увидеть хорошую реализацию Java, но я бы хотел, чтобы ее богатая библиотека классов была доступна и другой части системы.

Вывод
Сегодня удобство использования - важная часть любой системы, но она далека от совершенства и до некоторых областей еще просто не дошла. Новая платформа даст нам возможность распространить практичность во всех областях, по-настоящему делая компьютер инструментом для всех, а не инструментом, волю которого можно легко или периодически саботировать.

Легко поверить, что все, что может делать компьютер - это то, на что он способен сегодня, но каждая новая платформа приносит с собой новые возможности и новые приложения. В следующей части мы посмотрим на то, как мы на самом деле будем использовать новую систему и новый гибкий графический интерфейс, который я включу.

Чего-то не хватает?
Эта серия не является подробным описанием последних технологий в любой данной области, я не знаю всего и не претендую на это. Если вы знаете технологии, которые я не упомянул, но которые были бы полезны - расскажите о них в комментариях.

Ссылки / Подробная информация

[ROS] Ruby OS Interface Guidelines

[Proximal] Проксимальный интерфейс

[Fitts] Закон Фитта

[Case] Чувствительност к регистру

[Информация по человеческим интерфейсам]

Gnome HIG

Apple HIG

Cornell University HIG

Кратко о человеческих интерфейсах

Nielsen Norman Group HIG

Зал позора пользовательски интерфейсов

Зал славы пользовательски интерфейсов

Предыдущие части этой серии статей:
Часть первая: Аппаратные средства
Часть вторая: ОС
Часть третья: Безопасность и файлы

Об авторе:
Николас Блэхфорд (Nicholas Blachford) - 33-х летний эмигрант из Британии, ныне живущий в Париже, но не говорящий по-французски (еще). Интересуется различными причудливыми темами (Аппаратное обеспечение, программное обеспечение, фотография) и всякими другими вещами, особенно теми, в которых используются новые технологии. На сегодняшний день безработный.