

С. В. Ветров

Учебное пособие

Введение в проектирование пользовательских интерфейсов

УДК 004.5
ББК 32.973

Рекомендовано к изданию учебно-методическим советом Забайкальского государственного университета.

Рецензенты

[REDACTED], [REDACTED]

[REDACTED], [REDACTED]

Ветров, Сергей Владимирович

Человеко-машина взаимодействие / С. В. Ветров; Забайкальский государственный университет. – Чита : ЗабГУ, 2022. – 181 с. Дополненное издание.

В пособии освещены все основные темы рабочей программы, оно призвано кратко осветить основные вопросы создания пользовательских интерфейсов. Пособие предназначено для учебно-методической поддержки дисциплины «Человеко-машина взаимодействие».

Издание адресовано студентам бакалавриата, обучающимся по направлениям 09.03.01 Информатика и вычислительная техника.

Последняя версия пособия:

github.com/ivtipm/HCI/releases/download/StudBook/StudBook.pdf

Оглавление

| | |
|--|-----------|
| Введение | 4 |
| 1 Психика с точки зрения дизайна пользовательских интерфейсов | 8 |
| 1.1 Внимание | 9 |
| 1.2 Автоматичные задачи | 13 |
| 1.3 Память | 18 |
| 1.4 Ментальные модели | 20 |
| 2 Пользовательский интерфейс | 28 |
| 2.1 Понятие интерфейса | 28 |
| 2.2 Классификация видов ПИ | 29 |
| 2.2.1 Реализация интерфейса командной строки | 35 |
| 2.3 Парадигмы интерфейса | 37 |
| 2.4 Модальность | 41 |
| 3 Понятие дизайна и визуальное восприятие | 44 |
| 3.1 Общие представления о дизайне | 44 |
| 3.2 Зрительное восприятие | 48 |
| 3.3 Цветовые модели | 58 |
| 3.4 Цвет в интерфейсе | 60 |
| 4 Проектирование UX | 64 |
| 4.1 Проектирование пользовательского интерфейса | 64 |
| 4.2 Проектирование UX | 66 |
| 4.2.1 Уровни UX | 66 |
| 4.2.2 Уровень стратегии | 68 |
| 4.2.3 Метод персонажей | 70 |
| 4.2.4 Диаграмма прецедентов | 74 |
| 4.2.5 Уровень структуры | 77 |

| | |
|--|------------|
| 4.2.6 Дизайн навигации и диаграммы потоков | 79 |
| 4.2.7 Уровень компоновки | 84 |
| 4.2.8 Уровень поверхности | 88 |
| 4.2.9 Основы работы в Figma | 91 |
| 4.3 Методология проектирования | 108 |
| 5 Юзабилити и тестирование | 112 |
| 5.1 Юзабилити | 112 |
| 5.2 Юзабилити-тестирование | 120 |
| 5.2.1 Понятие юзабилити-тестирования | 120 |
| 5.2.2 А/В-тестирование | 121 |
| 5.2.3 Тестирование с наблюдением | 122 |
| 5.2.4 Метод карточной сортировки | 125 |
| 6 Анализ интерфейса | 127 |
| 6.1 Количественная оценка пользовательского интерфейса . | 127 |
| 6.2 Закон Фиттса | 127 |
| 6.3 Закон Хика и проблемы выбора | 136 |
| 6.4 GOMS | 138 |
| 6.5 Информационная эффективность интерфейса | 143 |
| 6.5.1 Информативность интерфейса | 143 |
| 6.5.2 Оценка информативности | 144 |
| 7 Типографика текст | 150 |
| 7.1 Понятие типографики | 150 |
| 7.2 Классификация шрифтов | 152 |
| 7.3 Основные понятия из типографики | 156 |
| 7.3.1 Выбор шрифта | 160 |
| 7.3.2 Шрифтовая иерархия | 163 |
| 7.3.3 Выравнивание текста и длина строки. | 164 |
| 7.4 Текст и синтаксис интерфейса | 167 |
| Заключение | 175 |
| Библиографический список | 176 |
| Предметный указатель | 181 |

Введение

Рассмотрим проблемы, с которыми сталкивается пользователь и задачи, которые решает дизайнер интерфейсов. В 1969 году в США случилась авария на атомной станции Три-Майл-Айленд. Течение аварии усугубили несколько факторов. Одна единственная сигнализация срабатывала при входе за пределы нормы любого из примерно 100 параметров. Одни из параметров имели критически важное значение, другие – нет. Органы управления и индикаторы на пульте станции не были логически сгруппированы. Принтер, печатающий диагностические данные, работал медленно. Во время аварии он отставал на 2 часа.

В 1988 году ракетный крейсер США сбил пассажирский самолёт над Персидским заливом, приняв его за военный. Среди прочих факторов, приведших к инциденту, комиссия, расследовавшая случившееся, отметила недостатки в пользовательском интерфейсе. В полной мере не учли, что интерфейс будет использован в сложной боевой обстановке, во время утомительного дежурства. Незаметно переназначенные программой идентификаторы целей привели к катастрофе.

Пользовательские интерфейсы могут быть настолько плохи, что становятся причинами катастрофы. Но гораздо чаще, интерфейсы «всего лишь» снижают продуктивность пользователей, приводят к потере данных, вызывают неудовлетворённость работой, мешают электронной коммерции, делают продукт неконкурентоспособным.

Мы до сих пор сталкиваемся с проблемами в работе интерфейсов в повседневной жизни. Читатели, наверняка могут привести мно-

го примеров, когда им не удалось совладать с программой, решая несложную задачу, удавалось допустить явные ошибки в работе, которые не были заметны до самого конца или приходилось тратить много времени на рутинную работу, которая, казалось бы, должна быть давно поручена компьютерам.

Одна из основных проблем – непонимание того, что пользователям не нужны программы сами по себе. Пользователям нужны инструменты для решения их собственных задач. Программа должна требовать внимания, времени и сил пользователя только в той степени, в которой это помогает решать проблемы пользователя.

Разработчики программы для ресторана, призванной упростить бронирование столов служащими плохо изучили специфику работы главных для себя людей – пользователей их продукта. Работать с программой оказалось проще, если просто делать пометки прямо на дисплее, когда открыто окно программы, а не взаимодействовать с ней привычным образом.

Многие пользовательские интерфейсы создание инженерами или программистами несут в себе родовую травму полученную не без участия заказчика продукта. Все эти люди заинтересованы в том, чтобы их продуктом пользовались. Но достаточно ли они квалифицированы для проектирования интерфейсов? Хватает ли им здравого смысла, чтобы понять – скорее всего они не компетентны в создании пользовательских интерфейсов. Поэтому часто пользовательский интерфейс разрабатывается или в последнюю очередь, когда основной код уже написан, или людьми без достаточной квалификации, не дизайнерами.

Даже если над продуктом работали отличные архитекторы, программисты и тестировщики продукт будет может стать нежизнеспособным, непопулярным и может не оправдать затрат на его создание, если он не подойдёт или не понравится пользователям. Сайт интернет-магазина может *выглядеть* не внушающим доверия, отдельные части сайта могут сбивать некоторых пользователей с

толку препятствуя совершению целевых действий, например регистрациям или покупкам¹.

Интерфейс — первое с чем сталкивается пользователь. Почти всегда пользователь не может непосредственно оценить внутреннее устройство программы, сколь бы прекрасным оно ни было, но регулярно взаимодействуя с программой он рано или поздно заметит многие недочёты интерфейса.

Современный подход ставит во главу разработку *продукта*² для клиента (пользователя) т.е. проектирование всесторонних потребительских качеств продукта.

Этот путь начинается с идеи продукта. Какую задачу он будет решать? Нужно изучить потенциальную целевую аудиторию, чтобы понять их потребности и специфику. Составить набор возможностей продукта, который поможет пользователям решить их задачи. Придумать, как эти возможности будут реализованы в интерфейсе. Реализовать самый удачный вариант интерфейса, протестировать и исправить недочёты.

К сожалению не существует чётких метрик, по которым можно дать абсолютную оценку качества интерфейса, но сравнивая разные варианты можно выбрать лучший. В этом смысле проектирование интерфейсов — это искусство. Поэтому стоит помнить, что любые правила и рекомендации по проектированию могут иметь исключения.

¹ см. пример в параграфе 5.2.2

² Под продуктом будем понимать программу, безразлично для какого устройства, сайт, веб-приложение.

Структура пособия

Для проектирования хороших пользовательских интерфейсов нужно понимать как устроено человеческое внимание, память, способность к формированию привычек. Важно понимать разницу между внутренним устройством программы, идеей интерфейса и тем пониманием устройства программ, которым обладает пользователь. Об этом рассказывает первая глава.

Во второй главе вводится понятие пользовательского интерфейса, даётся общее описание разным видам интерфейсов и принципиальным подходам, на которых их можно строить.

Третья глава описывает основные особенности человеческого восприятия, которые должен знать дизайнер интерфейсов.

Четвёртая глава посвящена подходу к проектированию продукта от общего к частному. С примерами разобраны основные виды работ, которые выполняет проектировщик интерфейсов. В частности, дано краткое описание возможностям сервиса для проектирования интерфейсов Figma.

В пятой главе приводятся принципы проектирования эффективных интерфейсов с примерами и антипимерами, описываются некоторые подходы к тестированию интерфейсов.

Шестая глава рассказывает об отдельных способах количественной оценки пользовательских интерфейсов и выводах, которые можно сделать на их основе.

В седьмой главе приводятся основные понятия из типографики, принципы оформления и организации текстовой информации вообще и в пользовательских интерфейсах в частности.

После каждой главы приведены избранные источники, которые полностью раскрывают тему главы. Понимание основ проектирования интерфейсов невозможно без изучения основных идей из этих источников.

В конце пособия приведён предметный указатель.

1 Психика с точки зрения дизайна пользовательских интерфейсов

Руководства по разработке продуктов, взаимодействующих с человеком физически, обычно содержат информацию, о свойствах и возможностях опорно-двигательного аппарата и органов чувств человека. Совокупность сведений в этой области составляет науку эргономику. На основе этих знаний можно проектировать стулья, столы, клавиатуры или дисплеи, которые с высокой степенью вероятности будут удобны для своих пользователей.

Невозможно эффективно управлять автомобилем, если его органы управления будут значительно удалены друг от друга, требовать существенных физических усилий или экстраординарной точности движений. Так же программа не может быть эффективно использована, если она требует от пользователя беспрецедентной внимательности, способности удерживать в уме большее количество информации и полного понимания внутреннего устройства самой программы.

Поэтому важно понимать особенности человеческой психики, его сильные и слабые стороны чтобы проектировать интерфейсы, которые помогут эффективно решать задачи пользователя.

1.1 Внимание

Внимание — избирательная направленность восприятия на тот или иной объект.

Даниэль Канеман¹ предложил считать внимание ресурсом. Любая относительно сложная задача или отвлекающий фактор "расходуют" часть внимания человека [14]. Услуга, программа, сайт или устройство редко используются в идеальных условиях, когда им уделено всё внимание пользователя. Они конкурируют за внимание с другими задачами пользователя и с окружающей средой.

Люди часто отвлекаются. Поэтому и интерфейс должен быть таким, чтобы пользователь мог снова как можно быстрее вернуться к работе и допускал как можно меньше ошибок из-за невнимательности.

В том числе поэтому, покупая билет на поезд на сайте железнодорожной компании, мы чаще всего заполняем не все необходимые данные сразу, а поэтапно (обычно на отдельных страницах) уделяя внимание вводу данных одной категории за раз: дата поездки, направления и номер поезда; выбор класса обслуживания и места; заполнение данных пассажира; страницы оплаты с обязательным повторением всех данных о поездке. Возможно, придётся отвлечься, чтобы сверится со своим расписанием или написать сообщение друзьям, найти банковскую карту или паспорт. Отвлёкшись на любом из этапов, пользователь должен быстро понять, что он уже сделал и что от него требуется дальше, иметь возможность понять, не ошибся ли он при вводе данных. Такое поэтапное решение задачи ещё и минимизирует использование памяти пользователя, о чём будет сказано ниже.

В лучшем случае пользователь может уделять всё своё внимание решению *своей задачи* с помощью программы, но не пользовательскому интерфейсу. Внимание не только ограничено, но ещё и изби-

¹Канеман – психолог, специалист по когнитивистике, лауреат Нобелевской премии (2002)

рательно. Например, пользователь пишет код в среде разработки, снабжает его комментариями на русском языке, и часто переключая раскладку клавиатуры, ошибается, не обращая внимания на то, какая именно раскладка задействована.

Плохо, если программа может в любой момент показать сообщение о выходе новой версии, об обновлении плагинов или показать несвоевременную подсказку о новых функциях. Тогда программа перетягивает внимание пользователя с решения *его* задачи, но задачу обслуживания самой себя. Если необходимо, то сообщения лучше показывать когда пользователь ещё не приступил к работе или уже закончил её.

Привлечение внимания

В интерфейсах часто нужно привлечь внимание к отдельным элементам, чаще всего кнопкам. Например в диалоговом окне стоит привлечь внимание к кнопке, которую скорее всего нажмёт пользователь. На сайте привлечь внимание к кнопке целевого действия. Например подписки, регистрации или покупки.

Эффект выскакивания (pop-out effect) помогает сократить время обнаружения² элемента интерфейса [53]. Эффект выскакивания – это независимость скорости поиска целевой стимула от общего количества стимулов. Благодаря этому эффекту человек может быстро находить отличающиеся от остальных объекты на изображении (рис. 1.1). Эффект тем выражение, чем больше отличается искомый объект от остальных. Для статичного изображения большую роль играют цвет, затем размер, форма и наклон объекта.

Если различия целевого стимула и остальных не велики, то эффект выскакивания выражен слабее, время поиска больше зависит от количества остальных визуальных стимулов. Это аргумент в пользу минималистичного дизайна.

²о выборе см. раздел 6.3 описывающий закон Хика

²см. также preattentive attributes

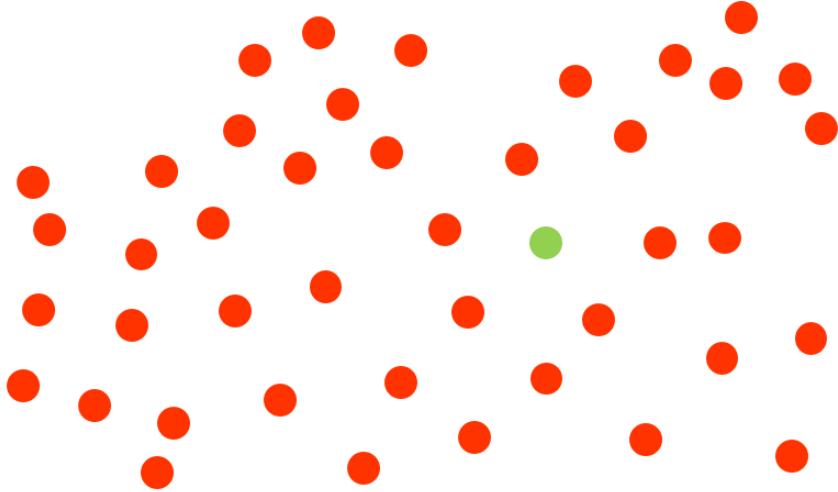


Рис. 1.1. Благодаря *эффекту выскакивания* люди быстро, автоматически находят зелёный круг среди множества других кругов одного цвета.

Человек отлично замечает движение. Поэтому анимация – хороший способ привлечь внимание, пусть ей уместно использовать в относительно узком круге сценариев.

Прерывание опыта – еще один способ привлечь внимание пользователей, особенно когда это делается во время рутинной задачи. Если нужно чтобы пользователи незамедлительно ознакомились с какой-то информацией, то отображение этой информации во всплывающем окне может быть эффективным (рис. 1.2). Но пользователю обычно потом требуется некоторое время чтобы вернуться к решению своих основных задач, возможно вспомнить то, о чём он думал до всплывающего сообщения. Поэтому этим способом стоит пользоваться с осторожностью. Кроме того это может раздражать пользователей.

Локус внимания

Идея или предмет, на котором сосредоточено внимание, называют **локусом³ внимания** (*locus of attention*) [28].

Локус внимания только один. Поэтому человек может не замечать то, что находится вне локуса внимания. Например, не обращать

³лат. locus — место, область

Итак, речь в этой статье пойдет о разоблачении одного мифа, который звучит примерно так (вот вам цитата из первого попавшегося на глаза сайта):

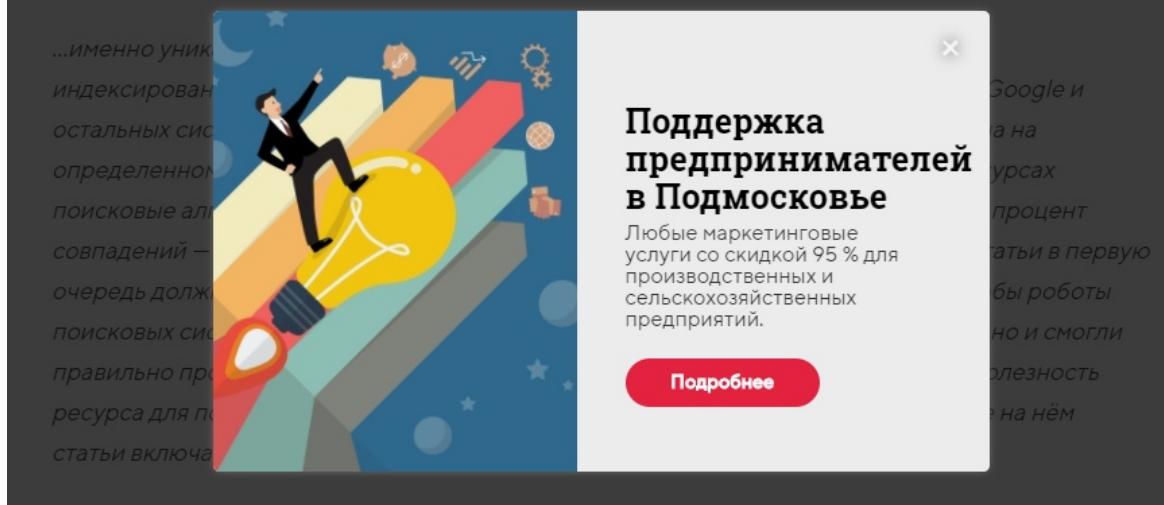


Рис. 1.2. Всплывающее во время просмотра страницы – это *прерывание опыта* – способ привлечь внимание

внимания на раскладку клавиатуры, когда в локусе его внимания находится кодирование алгоритма.

Человек может выполнять несколько задач одновременно, но только одна задача будет выполняться сознательно, она и будет локусом внимания. Выполнение же нескольких дел одновременно – это всегда либо поочерёдная смена локуса внимания либо выполнение других задач не требующих участия сознания (например ходьба). Поэтому невозможно решать несколько сложных интеллектуальных задач одновременно.

Локус может измениться неожиданно. Зазвонил телефон и локусом внимания стало уже само устройство или мысль "Кто это звонит?". Возврат в локус внимания прежнего занятия, от которого отвлекли, всегда требует ментальных усилий. Частые переключения с одной задачи на другую снижают нашу продуктивность. Мысленные усилия, затраченные на такие переключения, создают ложное ощущение, что проделан большой объём интеллектуальной работы.

Состояние потока

Люди, способные всецело сосредоточиться на некоторой деятельности, забывают о посторонних проблемах и отвлекающих факторах. Такое состояние называется **потоком**.

В состоянии потока человек может быть исключительно продуктивным, особенно если он занят созидательной работой, например конструированием, дизайном, разработкой или написание текстов.

Чтобы сделать пользователей более продуктивными стоит проектировать продукты, вызывающие и поддерживающие состояние потока, и прикладывать все возможные усилия, чтобы избежать любого поведения, потенциально способного разрушить поток. Если программа выбывает пользователя из потока (например диалоговым сообщением, рис. 1.6), ему трудно возвращаться в это продуктивное состояние [39].

1.2 Автоматичные задачи

Не все решаемые человеком задачи должны быть локусом внимания. Набор текста на клавиатуре слепым методом, так же как и езда на велосипеде или ходьба пешком по тропинке, лучше всего получается, если человек об этом не задумывается. Но как только задумается, то может можете сбиться. По мере повторения – или с практикой – выполнение того или иного действия становится привычным, и получается выполнять его не задумываясь.

Любая задача, которую человек научился выполнять без участия сознания, становится **автоматичной**. Любая последовательность действий, которую регулярно выполняют, становится, в конце концов, автоматичной. Автоматизм позволяет выполнять сразу несколько действий одновременно. Все одновременно выполняемые задачи, за исключением не более чем одной, являются автоматичными.

Та задача, которая не является автоматичной, является локусом внимания.

Когда человек выполняет одновременно две задачи, ни одна из которых не является автоматичной, эффективность выполнения каждой из них снижается в результате конкуренции за область внимания. Этот феномен психологи называют *интерференцией*.

Чем больший набор задач человек может решать без привлечения сознания – автоматично, тем больше может делать дел одновременно.

Относительно несложные действия, которые человек выполняют регулярно, становятся автоматичными. Это происходит благодаря способности человека формировать привычки. Если пользователь часто вводит один и тот же пароль при входе на сайт или после загрузки ОС, то в определённый момент замечает: вводя пароль легче ошибиться если начать думать о нём во время ввода.

Нужно создавать интерфейсы, которые, во-первых, целенаправленно опираются на человеческую способность к совершению автоматических действий и, во-вторых, развиваются у пользователей такие привычки, которые позволяют упростить ход работы. В случае идеального человекоориентированного интерфейса доля участия самого интерфейса в работе пользователя должна сводиться к формированию автоматических действий. Тогда пользователю легче сконцентрироваться на решении своих задач.

Пользователи запоминают расположение часто используемых кнопок (рис. 1.3), элементов меню и т.д., поэтому могут ими пользоваться практически не читая надписей. Если поменять местами расположение таких часто используемых кнопок (например кнопки "Ок" и "Отмена" в привычном диалоговом окне), то какое-то время человек не сможет эффективно пользоваться программой. Так же как бы не смог бы уверено водить автомобиль, у которого педали газа и тормоза поменяли местами. По той же причине динамиче-

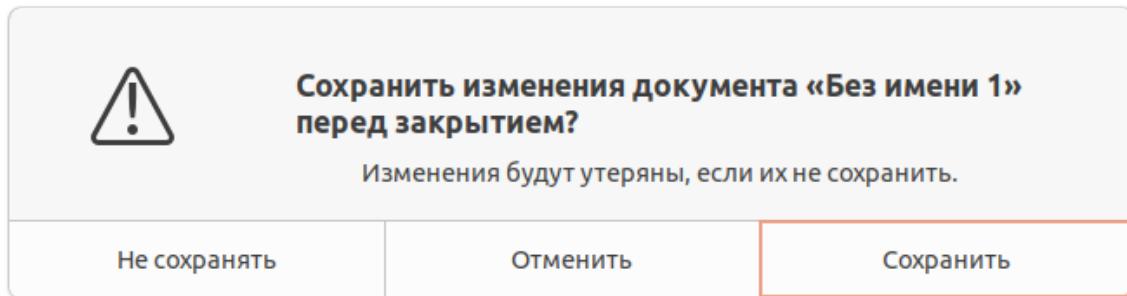


Рис. 1.3. Если дизайн диалоговых окон для типичных запросов содержит одинаковый порядок кнопок, то пользователю будет легко сформировать привычку нажимать на нужную кнопку практически не читая содержимое.

ски упорядочивать элементы меню в программе в зависимости от частоты их использования – плохая идея.

С другой стороны, интерфейс может способствовать формированию вредных пользовательских привычек. Злоупотребление всплывающими окнами с сообщениями (только с кнопкой ОК) или с простым выбором (OK, Отмена) привело к тому, что многие пользователи привыкли эти окна тут же закрывать. Такую привычку легко понять. Чаще всего окно с сообщением не содержит полезной информации (рис. 1.4), например: "ошибка 50" или "неожиданная ошибка". Иногда окно не содержит важной информации и пользователь может просто продолжить работать дальше даже не читая сообщения.

Многие проблемы, которые делают программные продукты сложными и неудобными в использовании, происходят из-за того, что в используемом интерфейсе «человек-машина» не учитываются полезные и вредные свойства человеческой способности формировать привычки. Тенденция предусматривать сразу несколько путей решения одной и той же задачи в одних и тех же условиях как раз мешает формированию привычек. Множество вариантов действия приводит к смешению локуса внимания пользователя с самой задачей на выбор пути её решения.

Привычки трудно менять. Поэтому человеку бывает трудно, переключится на использование новой клавиатуры или новой, сильно изменившей интерфейс, программы. Это может удержать поль-

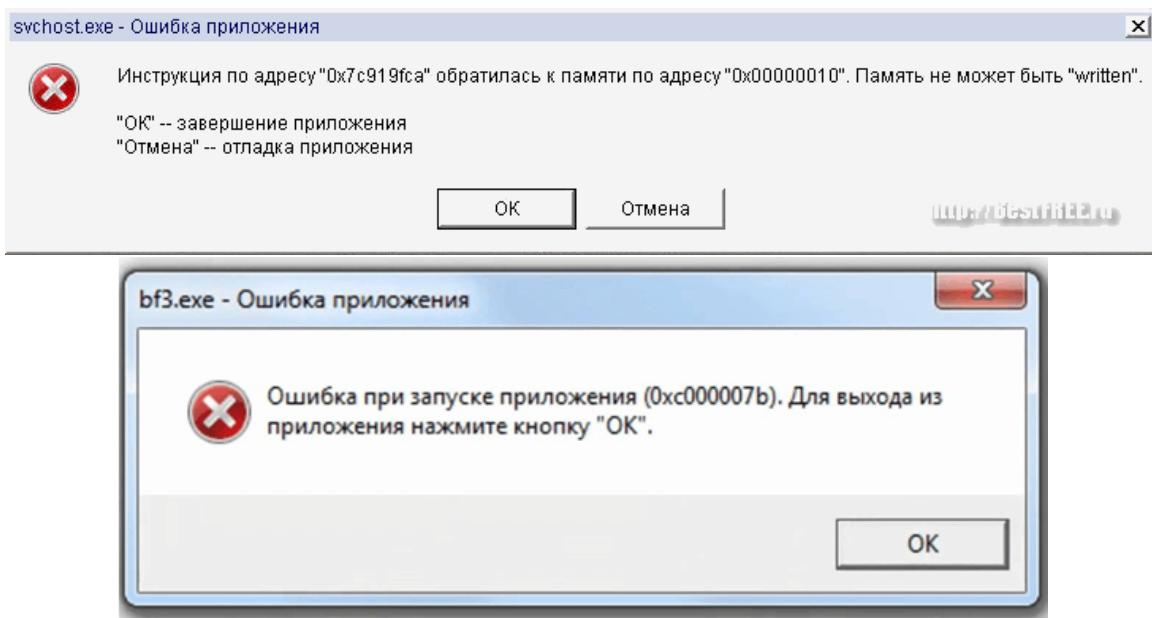


Рис. 1.4. Злоупотребление диалоговыми окнами с бесполезными для пользователя сообщениями часто приводит к формированию привычки закрывать любые диалоговые окна не вникая в содержимое сообщения.

вателя от перехода на продукт конкурентов, или наоборот помочь перейти на ваш, если старые привычки пользователей окажутся учтены в новой программе. Например версии приложения [REDACTED]⁴ для iOS и Андроид используют привычные для этих платформ расположение меню (рис. 1.5). Многие программисты имеют привычку нажимать комбинацию клавиш Ctrl+S для сохранения файла исходного кода после каждого логически завершённого изменения. Абсолютное большинство программ интерпретируют эту комбинацию клавиш одинаковым образом.

Другая польза автоматических задач – они не отвлекают пользователя и он может выполнять несложные манипуляции не отвлекаясь на интерфейс программы. После сформированной привычки ему не приходится искать нужный пункт меню или кнопку, вспоминать горячую клавишу.

⁴автор вынужден цензурировать некоторые места учебного пособия, посвящённого исключительно дизайну интерфейсов из-за событий 2022 года

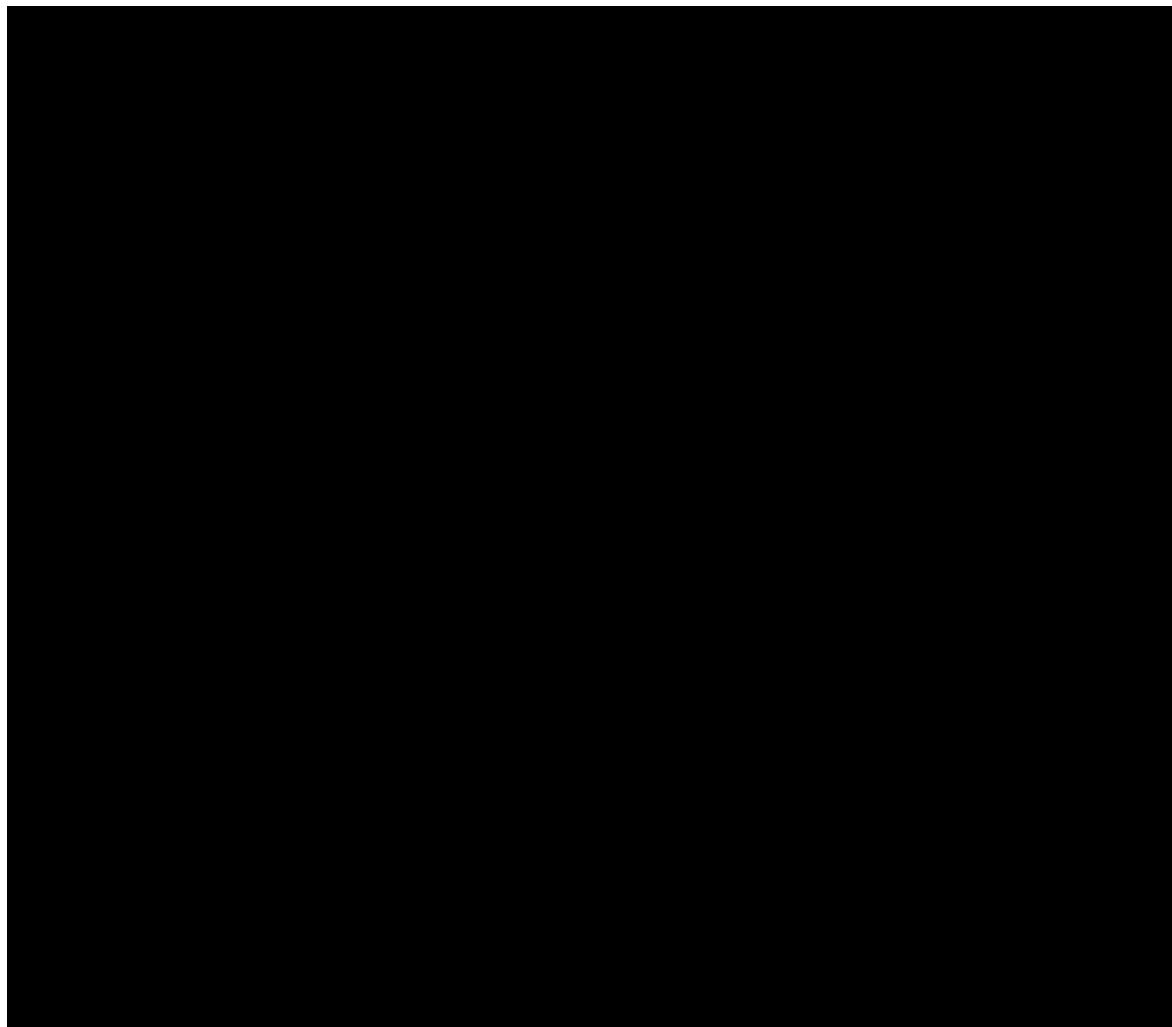


Рис. 1.5. Пример адаптации приложения [REDACTED] под разные ОС и, следовательно, привычки пользователей этих ОС. Меню приложения в iOS расположено внизу (изображение слева), в версии для Андроид меню приложения расположено вверху.

1.3 Память

Человеческую память можно разделить на два вида: долговременную и кратковременную [55]. Кратковременная память — компонент памяти, в который информация поступает из сенсорной памяти, после обработки процессами восприятия, и из долговременной памяти (воспоминания), позволяющий удерживать на короткое время небольшое количество информации в состоянии, пригодном для непосредственного использования сознанием.

Эксперименты Джорджа Миллера показали, что кратковременная память человека способна запоминать в среднем восемь десятичных цифр, девять двоичных цифр, семь букв алфавита и пять односложных слов [51]. Все запоминаемые сущности не были связаны друг с другом по смыслу. Выявленная закономерность получила название "Магическое число 7 ± 2 ".

Это правило широко освещается в интернет-публикациях по проектированию интерфейсов. Однако опыт показывает прямое применение этого правила, например для скрытия редко используемых пунктов меню, ограничения числа элементов в списках и т.п. не приводит к повышению продуктивности пользователя [1, 57].

Продолжительность хранения информации (при условии, что нет повторения) около 20 сек. После 30 сек. след информации становится настолько хрупким, что даже минимальная *интерференция* разрушает его. Это ещё одна причины учитывать смену локуса внимания пользователя во время использования программы. Чтобы вернуться к работе пользователю нужно заново погрузиться в решаемую задачу, желательно как можно быстрее и с меньшими усилиями.

Программа не должна заставлять пользоваться держать в голове информацию, если она сама может её легко хранить и отображать. Ведь память компьютера быстрее, часто надёжнее и способна вмещать большой объём информации, в отличии от человеческой памяти. Например если человек отвлёкся, а потом увидел сообщения (рис.

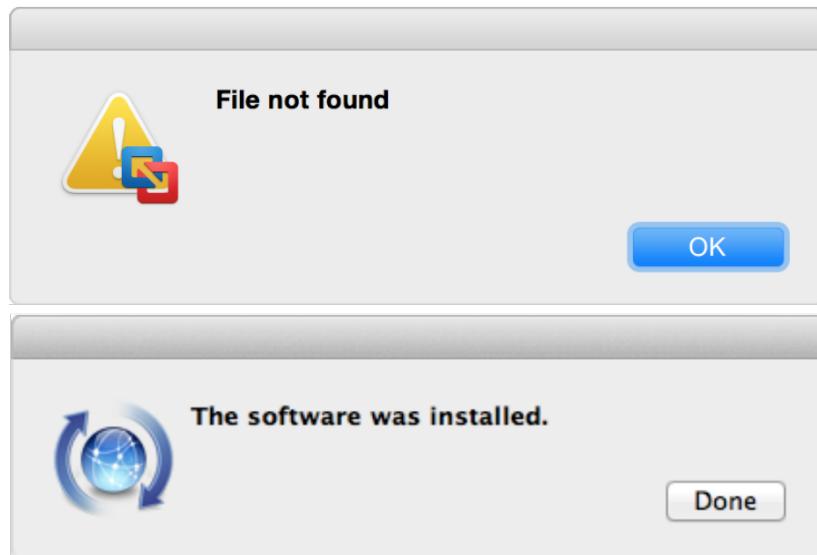


Рис. 1.6. Программа не должна заставлять пользователя держать в голове информацию, если она сама может её легко хранить и отображать. Если человек отвлёкся, то ему придётся вспоминать о каком файле (верхнее окно) и о какой программе (нижнее окно) ему сообщают [1].

1.6), то ему придётся вспоминать о каком файле (верхнее окно) или о какой программе (нижнее окно) ему сообщают [1].

Сохранение не до конца оформленного в интернет-магазине заказа, недописанного сообщения в мессенджере или открытие текстового документа сразу в месте последнего редактирования помогает пользователю быстрее перейти к работе.

Если пользователь хочет конвертировать 35139 рублей в доллары США, то плохая программа заставит пользователя сделать много подготовительных действий и держать всё это время длинную цифру в голове прежде чем записать в поле ввода. Например программа конвертации единиц измерения и валют (рис. 1.7) сначала предлагает выбрать вид конвертируемых единиц (длина, масса, объём, валюта), потом выбрать исходную единицу (например рубли из большого списка валют), выбрать целевую единицу (снова из большого списка) и только потом предложит записать значение.

Программа не должна заставлять пользователя держать в голове слишком много информации. С учётом того, что пользователь и так в данный момент может совершать сложную интеллектуальную работу.

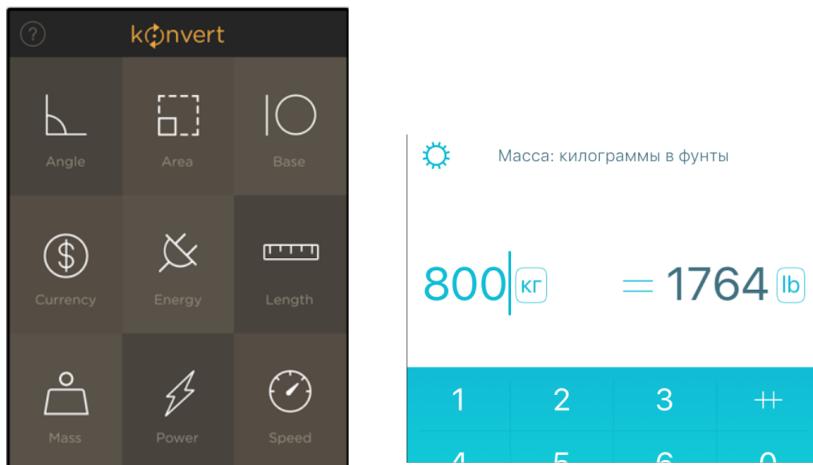


Рис. 1.7. Принцип «сначала данные»: программа не должна заставлять пользователя держать в голове данные, вынуждая сначала задать настройки, выбрать режим (программа справа) или фильтры. На изображении справа пользователь сначала вводит число, а потом задает исходные единицы измерения и выбирает целевые единицы измерения [1].

Это же и касается случаев, когда пользователю нужно сделать выбор – удержать в голове большое число вариантов довольно трудно (смотрите также параграф 6.3 о законе Хика). Поэтому в отдельных случаях можно либо сразу делать выбор за пользователя либо предлагать ему небольшое количество готовых вариантов.

1.4 Ментальные модели

Ментальные модели⁵ представляют понимание человека как нечто функционирует [24]. Примеры ментальных моделей: способ завязывания шнурков, способ завести автомобиль; местонахождение ближайшей кофейни; способ использования дрели; разблокировка телефона и настройка будильника на часах (рис. 1.8).

Человек учится использовать объект, основываясь на наблюдениях и опыте использования. Например вместо сложной таблицы, где перечислены все возможные ставки и сроки вклада на сайте банка есть интерактивный калькулятор (рис. 1.9). С ним легко эксперименти-

⁵Модель – это упрощённое представление действительного объекта и/или протекающих в нём процессов



Рис. 1.8. В некоторых случаях ментальные модели могут появляться быстро: как разблокировать телефон? В других случаях ментальные модели могут возникать в процессе активного изучения устройства: Как настроить будильник на 9:00 в этих наручных часах?

ровать меняя срок и сумму вклада. Итоговая сумма рассчитывается автоматически при изменении параметров. Это помогает пользователю понять как устроен вклад – т.е. сформировать ментальную модель. Мы строим ментальные модели основываясь на объекте (иногда достаточно просто посмотреть на него, например рис. 1.10), читая руководство пользователя или спрашивая других людей.

Разные люди имеют разные ментальные модели одних и тех же объектов – то есть люди имеют ментальные модели с разным уровнем абстракции. Кто-то понимает как работает лазерная мышь, как устройство отслеживает изменение положения и как передаёт эти данные. Но у большинства людей просто понимают только общий принцип работы устройства.

Ментальные модели могут быть ошибочными: Земля плоская, нельзя выключать компьютер нажав на кнопку включения на системном блоке.

Ментальные модели человека могут быть ошибочными, но при этом работать (рис. 1.11). Если представлять, что внутри фотоаппарата сидит чёртик, способный рисовать увиденное за миллисекунды, то это не обязательно помешает фотоаппаратом пользоваться.

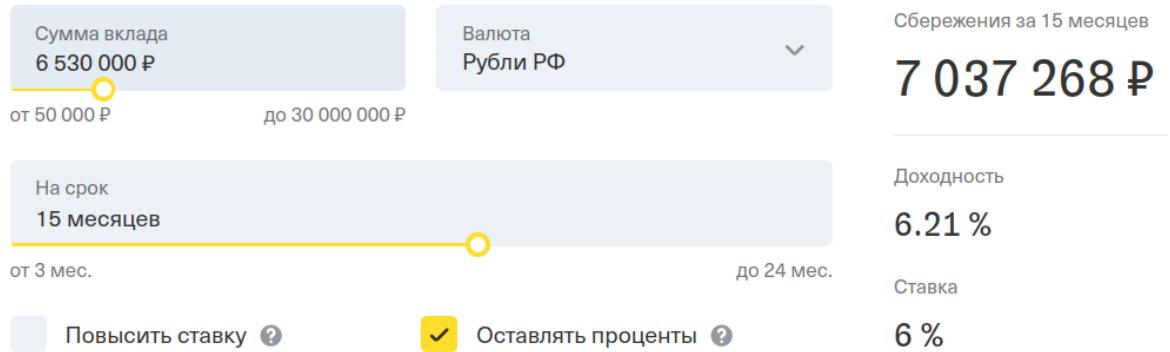


Рис. 1.9. Калькулятор доходности вклада. Вместо сложной таблицы, где перечислены все возможные ставки и сроки вклада на сайте банка есть интерактивный калькулятор. С ним легко экспериментировать меняя ползунками срок и суммы вклада, итоговая сумма рассчитывается автоматически при изменении параметров. Это помогает пользователю понять как устроен вклад, т.е. сформировать ментальную модель.

1. [Download](#)

2. [Download](#)

3.

Download

Рис. 1.10. Легко узнаваемые ментальные модели: текст, ссылка и кнопка

Модель реализации описывает фактическое устройство чего либо.



Рис. 1.11. Модель реализации и возможная ментальная модель

Задача дизайнера интерфейса придумать принципы взаимодействия пользователя и программы. Создать ментальные модели более простые чем модель реализации, но при этом обеспечивающие эффективное взаимодействие пользователя и программы. Такие модели, предлагаемые дизайнером, называются **моделями представления** или моделями дизайнера.

Модели модели могут не объяснять все особенности устройства или программы, а быть отдельный слоем абстракции – принцип сокрытия сложности. Можно создавать программы, решающие сложные задачи, сколько угодно сложно устроенные внутри, но с достаточно простым интерфейсом.

Модели представления должны быть логичными, применяться последовательно и содержать как можно меньше исключений чтобы пользователю было легче их понять. Иконка программы на главном экране телефона воспринимается как программа потому, что легко связать факт тапа на иконку и запуск программы. Тем более, что

анимация открытия этой программы может показывать как она как бы вырастает из этой иконки. Значит пользователь может предположить, что иконка обладает свойствами программы (хотя бы частично) и через неё, можно неким образом удалить программу из устройства. Долгое нажатие на иконку открывает меню действий: удалить иконку, удалить приложение, открыть экран "О приложении". Значит такое же действие можно совершить, например с любым элементов на главном экране телефона, например виджетом который показывает время или даже попробовать открыть меню так же зажав имя контакта в телефонной книге.

Непоследовательное использование терминов и обозначений в программе вносит путаницу, требует от пользователя держать в голове дополнительную информацию. Например использование одного и того же значка лупы для обозначения разных действий: открытие окна поиска и изменения масштаба; использование значка в виде креста для удаления данных и для отмены действия.

В хорошем интерфейсе модель представления легко понять или она уже соответствует ментальной модели пользователя. Программы же обычно имеют довольно сложное внутреннее устройство. Принципы их построения скорее всего не знакомы пользователю. Поэтому, модель представления не должна полностью повторять модель реализации. Более того, модель представления может существенно отличаться от модели реализации.

Проклятие знания – это когнитивное искажение⁶: более информированным людям чрезвычайно сложно рассматривать какую-либо проблему с точки зрения менее информированных.

Разработчику программы трудно представить, какие проблемы в понимании программы могут возникнуть у пользователя. Ведь разработчик не только понимает, как взаимодействовать с программой, но и как программа устроена изнутри. Это одна из причин, невысокого мнения программистов о пользователях. Это мешает

⁶см. также ru.wikipedia.org/wiki/Список_когнитивных_искажений

программисту стать на место пользователя, а значит спроектировать хороший интерфейс. Поэтому тестирование интерфейса (см. главу 5.2) с участием людей не занятых в работе над проектом – это неотъемлемая часть разработки ПИ.

Ментальные модели редко образуются после чтения инструкции из-за **парадокса активного пользователя**.

Парадокс впервые был представлен в исследовании Мэри Бет Россон и Джона Кэрролла, исследователями из IBM, в 1980-х годах. Наблюдая за новыми пользователями, было определено неожиданное на тот момент поведение: пользователи редко читали руководство по продукту. Вместо этого пользователь начинал взаимодействовать с продуктом, чтобы опробовать его, даже если это означало столкновение с ошибками, которых можно избежать, прочитав инструкцию [37].

Парадокс заключается в том, что пользователи могли бы получить выгоду в долгосрочной перспективе сперва разобравшись как пользоваться программой (например изучив руководство, в том или ином виде). Это могло бы оградить их от ошибок и впустую потраченного времени в попытке решить свои задачи неправильным способом. Подход: сделать интерфейс не заботясь о его сложности и понятности, а потом объяснить принципы его работы в руководстве пользователя здесь работает плохо.

Из этого следуют важные выводы. Пользователи не всегда поступают рационально. Но желание как можно быстрее приступить к использованию программы понятно – для пользователя конечная цель – решить свои задачи используя программу, а не изучить программу.

Не все программы возможно сделать понятными для новичка без руководства пользователя. Но при проектировании интерфейса стоит учитывать, что многие пользователи предпочтут научится пользоваться программой на ходу.

Разработчику не стоит слишком сильно рассчитывать на то, что он сможет изменить пользователя. Сможет донести до него существенную информацию о внутреннем устройстве программы и о принципах её работы, сделать пользователя внимательным, заставить поступать рационально, держать в памяти всю необходимую информацию, выполнять все действия быстро и точно. При этом такой идеальный пользователь ещё будет решать собственные задачи в программе и заниматься её обслуживанием, обновлять, разбираться как исправить возникающие ошибки. Но разработчик может спроектировать интерфейс учитывая человеческие слабости: ограниченность и переменчивость внимания, возможно нежелание изучать программу, неправильные ментальные модели.

Программы вместо человека должны держать всё под контролем, хранить необходимые данные, оберегать пользователя от совершения ошибок, а если он их совершил, то давать возможность исправить минимальными усилиями. При решении сложных задач, программа должна помогать пользователю освоить новые ментальные модели и полагаться на имеющиеся, использовать способность человека к формированию привычек для ускорения взаимодействия с программой.

Вопросы

1. Что такое локус внимания? Что такое интерференция?
2. При каких условиях человек может выполнять несколько задач одновременно?
3. Что такое состояние потока?
4. Охарактеризуйте кратковременную память? Какие выводы о ПИ следуют из этого?
5. Что такое автоматические задачи? Как они формируются?
6. Как можно использовать автоматические задачи?
7. Что такое ментальная модель? Приведите примеры.
8. Приведите примеры появления новых ментальных моделей в интерфейсах программ и устройств. Как они изначально были восприняты? Что можно сказать о них теперь?
9. Что такое модель реализации?
10. Что такое модель представления?
11. Как эти понятия относятся друг к другу?
12. Что такое проклятие знания? Как оно мешает программисту разрабатывать интерфейс?
13. Как должны соотносится ментальная модель и модель представления?
14. Что такое парадокс активного пользователя?

Литература

- Дизайн привычных вещей / Дон Норман; пер. с англ. Анастасии Семиной. — [2-е изд, обн. и доп.] — М.: Манн, Иванов и Фербер, 2018.
- Канеман Даниэль, Думай медленно... решай быстро. Россия: АСТ, 2020. 908 с.

2 Пользовательский интерфейс

2.1 Понятие интерфейса

Интерфейс – граница между двумя функциональными объектами, требования к которой определяются стандартом; совокупность средств, методов и правил взаимодействия (управления, контроля и т. д.) между элементами системы [54].

В случае интерфейса пользователя под элементами системы понимают пользователя и, в общем случае, программно-аппаратный комплекс. *Интерфейс пользователя* (ПИ, UI – User Interface) – интерфейс, обеспечивающий передачу информации между пользователем и программно-аппаратными компонентами компьютерной системы¹.

Далее будет идти речь об интерфейсе исключительно пользовательском. Для краткости под программой будем понимать ОС различных устройств, прикладные программы и сайты.

Как видно из определений главная цель ПИ – передача информации между человеком и программой. Эта передача может быть односторонней. Пассажиры получают информацию с табло со временем отправления и прибытия поездов не могут его поменять. Интерфейс может не передавать никакой информации пользователю, но быть доступным для действия пользователя как кнопка вызова лифта без индикации. ПИ может быть невидимым. Например, интерфейс голосового робота оператора мобильной связи.

¹другой важный вид интерфейса в программной инженерии – программный (API), позволяющий взаимодействовать программам между собой

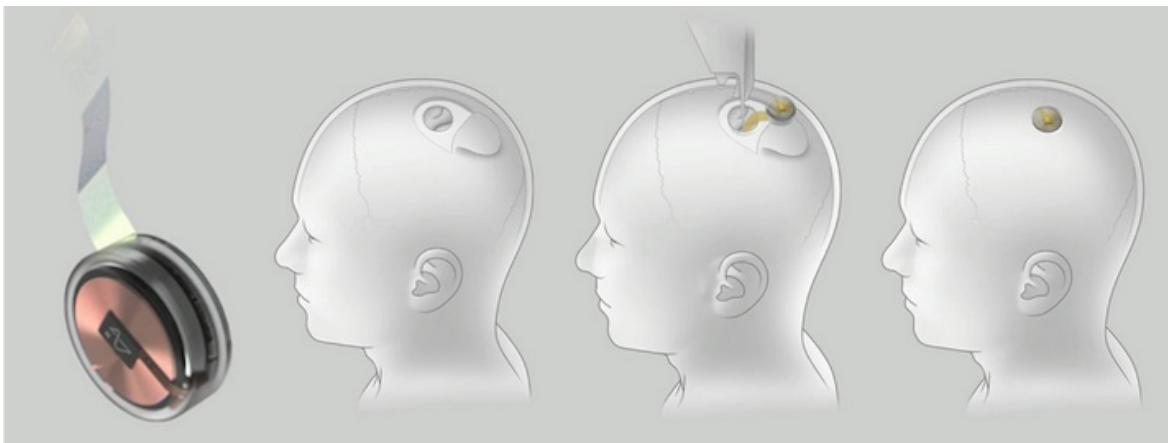


Рис. 2.1. Имплантируемый нейрокомпьютерный интерфейс Neuralink.

2.2 Классификация видов ПИ

Виды пользовательских интерфейсов.

- Визуальный;
 - Текстовый – Text-based Interface (TUI);
 - Графический (ГПИ или ГИП) – graphical user interface (GUI);
- Тактильный (Haptic or kinesthetic communication);
- Жестовый;
- Голосовой;
- Материальный (осзательный) - tangible user interface (TUI);
- Нейрокомпьютерный интерфейс.

Зачастую, конкретный ПИ является комбинацией нескольких видов интерфейса.

Нейрокомпьютерный интерфейс – система, созданная для прямого обмена информацией между мозгом и электронным устройством (например, компьютером). В односторонних интерфейсах внешние устройства могут либо принимать сигналы от мозга (см. рис 2.1), либо посыпать ему сигналы (например, имитируя сетчатку глаза при восстановлении зрения электронным имплантатом). Двунаправленные интерфейсы позволяют мозгу и внешним устройствам обмениваться информацией в обоих направлениях. В основе нейрокомпьютерного интерфейса часто используется метод биологической обратной связи.



Рис. 2.2. Музыкальный инструмент «reacTable» – электроакустический электронный музыкальный инструмент. Положение и ориентация специальных блоков на интерактивной поверхности влияет на генерируемый аудиосигнал. [youtube.com/watch?v=I9AeUISg-Og](https://www.youtube.com/watch?v=I9AeUISg-Og)

Материальный ПИ (tangible user interface, TUI) – это разновидность интерфейса пользователя, в котором взаимодействие человека с электронными устройствами происходит при помощи материальных предметов и конструкций. Например музыкальный инструмент «reacTable» (рис. 2.2) – электроакустический электронный музыкальный инструмент. Положение и ориентация специальных блоков на интерактивной поверхности влияет на генерируемый аудиосигнал. SandScape – это виртуальный ландшафт, формируемый пользователем из песка (рис. 2.3).

Жестовый ПИ используется в устройствах с сенсорными экранами – мобильных устройствах, интерактивных панелях, где основные жесты – это касание (тап), свайп, поворот и раздвигание двух пальцев. Здесь жестовый ПИ сочетается с графическим интерфейсом. Тачпад или компьютерная мышь – тоже часть жестового ПИ. Компьютерные игры могут использовать устройства захвата движений: PlayStation Move, Wii Remote и др.; распознавать жесты с видео: Microsoft Kinect, TrackIR, Azure Kinect DK (рис 2.4) и др.



Рис. 2.3. SandScape – виртуальный ландшафт и его объекты, взаимодействуют с пользователем, формирующим этот ландшафт с помощью песка. Демонстрация: vimeo.com/44538789

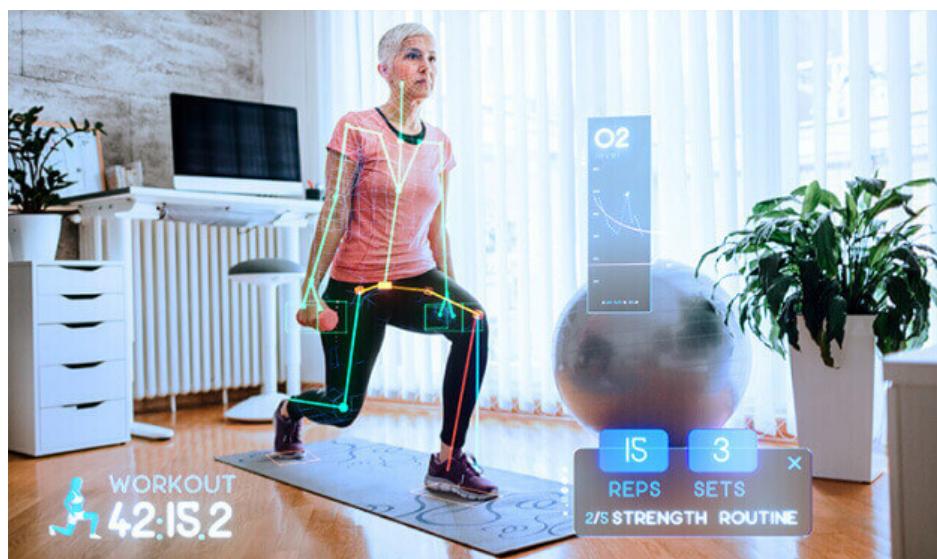


Рис. 2.4. Azure Kinect DK. Система компьютерного зрения распознаёт положение тела. Демонстрация: youtube.com/watch?v=OoJWmmOlws8



Рис. 2.5. Игровой контроллер с вибрацией – пример тактильного ПИ

Тактильный ПИ предполагает тактильную обратную связь с пользователем. Самый распространённый пример – вибрация игровых контроллеров (рис. 2.5) и телефона.

Текстовый пользовательский интерфейс (Text user interface, TUI; Character User Interface, CUI) – разновидность интерфейса пользователя, использующая при вводе-выводе и представлении информации исключительно набор буквенно-цифровых символов и символов псевдографики.

Интерфейс командной строки (Command line interface, CLI) разновидность текстового интерфейса (CUI), в котором инструкции компьютеру даются в основном путём ввода с клавиатуры текстовых строк (команд). Также известен под названием консоль.

Такой вид ПИ обладает техническим преимуществами. В нём легко автоматизировать действия, он не требует к вычислительным ресурсам. Возможность автоматизации можно рассматривать как дополнительную функцию, которая идёт вместе с интерфейсом. Второе преимущество улучшает субъективный пользовательский опыт на низко производительных устройствах и во время сетевого доступа (например, по SSH) при низкой скорости сети.

С таким типом интерфейса можно выполнять сложные задачи быстрее² чем с ГИП, особенно если учесть возможность автоматического

²для оценки времени на совершение действий смотрите параграф 6.4

```

~ ➤ cd testproject
~/testproject ↵ master ➤ gco detached-head-state -q
~/testproject ↵ fdffaf6 ➤ touch dirty-working-directory
~/testproject ↵ fdffaf6± ➤ cd
~ ➤ ssh milly
Welcome to Ubuntu 11.04 (GNU/Linux 2.6.18-308.8.2.el5.028stab101.1 x86_64)
Last login: Wed Sep 26 03:42:49 2012 from 71-215-222-90.mpls.qwest.net
agnoster@milly: ~
Connection to milly.agnoster.net closed.
~ ➤ sudo -s
Password:
↳ root@Arya ~ ➤ top &
[1] 34523
[1] + 34523 suspended (tty output) top
↳ root@Arya ~ ➤ rm no-such-file
rm: no-such-file: No such file or directory
✗ ↳ root@Arya ~ ➤ kill %_
[1] + 34523 terminated top
↳ root@Arya ~ ➤
~ ➤

```

Рис. 2.6. ZSH – командная оболочка для UNIX, с гибким механизмом автодополнения команд, исправления опечаток, цветовым кодированием состояний репозиториев git.

дополнения команд. Командные оболочки, например ZSH (рис. 2.6) до сих пор широко используются в UNIX-подобных ОС. Популярность таких оболочек (вместе с большим набором утилит) высока, особенно по сравнению с командной строкой Windows. Популярный консольный текстовый редактор vim тоже использует командный интерфейс.

Но такой тип интерфейса требует от пользователя многих ментальных моделей – знания команд, их параметров и принципов работы с ними. Эти ментальные модели сложнее формируются (плохая *изучаемость, learnability*³), в отличие от случая с ГИП, где можно изучить программу рассматривая ПИ и взаимодействуя с ней. Часто здесь нет и хорошей обратной связи – важного атрибута почти любого ПИ. Своебразное переходное звено от текстовых интерфейсов к географическим – интерфейсы с псевдографикой и текстовым исполнением меню и кнопок (рис. 2.7).

Но перечисленные недостатки не существенны, для части пользователей. Поэтому такой вид интерфейса широко распространён в программах для семейства ОС Linux. Его наличие ценится многими

³см. параграф 5.1 о критериях качества интерфейса

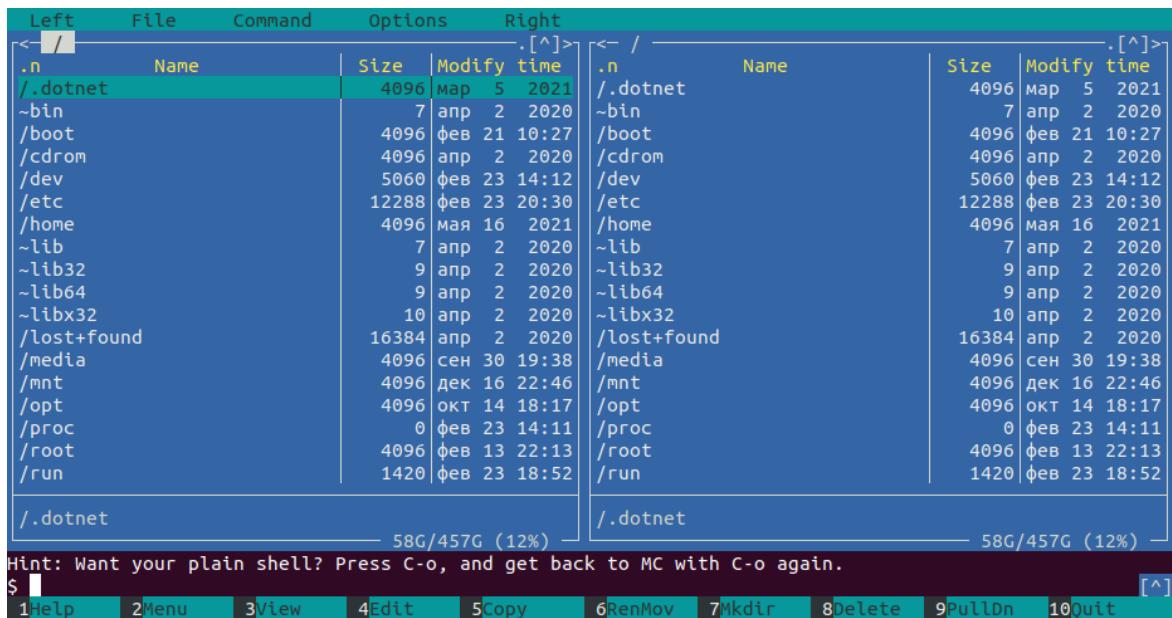


Рис. 2.7. GNU Midnight Commander – файловый менеджер с текстовым ПИ

специалистами. Ещё одно подтверждение тому, что перед созданием продукта важно знать свою целевую аудиторию.

Текстовый ПИ широко распространён в диалоговых системах. Например SMS-запросы, чат-боты для взаимодействия с сервисами заказа услуг, технической поддержки.

Графический пользовательский интерфейс исторически сменил текстовый, но не вытеснил до конца. Здесь нет необходимости знать текстовые команды, название многих утилит их аргументов и параметров, вводить их с абсолютной точностью.

Количество возможных способов взаимодействия с программой (возьмём все возможные комбинации символов команд и их аргументов в текстовом ПИ) в текстовом интерфейсе огромно. В графическом интерфейсе неделимых способов взаимодействия гораздо меньше. Как правило это клики мышью, прокрутка, клик и перемещение. Клавиатура чаще используется для ввода данных и реже для ввода команд (горячие клавиши). Благодаря появлению пользовательского интерфейса сократился лексикон взаимодействия (*interaction vocabulary*). При этом элементы интерфейса теперь видны на экране. А диапазон решаемых задач с помощью графического ПИ

как минимум не меньше чем для текстового. Далее в этом пособии будет рассматриваться преимущественно графический пользовательский интерфейс.

2.2.1 Реализация интерфейса командной строки

Программы с интерфейсом командной строки принято запускать из консоли (терминала, командной строки). Ведь такие программы часто быстро выполняются, а пользователь может изучать их вывод уже после того как они завершились.

Существуют два способа взаимодействия таких программ с пользователем: интерактивный, где пользователь вводит данные по запросу программы во время её выполнения; запуск с аргументами, сразу задающими все необходимые входные данные.

Большинство типичных консольных утилит вроде `ls` (просмотр списка файлов) или `cd` работают только с аргументами командной строки. Например:

```
> cd c:\User
```

`c:\User` – аргумент.

```
> ls c:\User -a
```

`c:\User, -a` – аргументы.

Компиляторы практически всегда имеют командный интерфейс на основе аргументов. Причем такой интерфейс подходит и для взаимодействия с пользователем и со средой разработки, которая так или иначе вызывает компилятор.

Например программу на диалекте Free Pascal можно скомпилировать так:

```
> fpc my_program.pas -FE my_program.exe
```

где `fpc` – имя компилятора; аргументы `my_program.pas` – имя файла с исходным кодом, `-FE` – ключ, за которым обязательно должно следовать название выходного файла, получаемого после компиляции. В примере это `my_program.exe`. Часто добавляют ключи `-CX` и `-XX` для уменьшения размера файла. Причём многие программы с подобным интерфейсом позволяют указывать такие ключи в произвольном порядке.

Как правило многие программы с таким интерфейсом способны реагировать на специальный аргумент `--help` (Linux) или `/help` (Windows) или ключ `-h` (Linux) `/h` (Windows) выводя подсказку о других аргументах и информацию о программе для пользователя.

`myprogram -h`

или

`myprogram --help`

Для лучшего понимания интерфейса командной строки рекомендуется изучить основы командной строки Windows и терминала Linux.

Приведём пример программы на диалекте Free Pascal, которая обрабатывает аргументы командной строки. Программа вычисляет длину гипотенузы, если заданы длины катетов.

```
program hypotenuse;

uses SysUtils; // для StrToFloat

const
  // короткий вариант справки
  HELP_BRIEF :string = 'hypotenuse <a> <b>';

  // справка, которая будет выведена пользователю
  HELP: string = 'hypotenuse <a> <b>;' + #13#10 +
    'Вычисляет длину гипотенузы по длинам катетов a и b' + #13#10 +
    'hypotenuse [-help|-h] -- вывод текущей справки';
  // #13#10 -- символы обозначающие конец строки, переход на новую
  // обязательные аргументы принято указывать в угловых скобках <>;
  // не обязательные в квадратных [], символ | означает ИЛИ

var
  a,b, c: real; // катет, катет, гипотенуза
```

```

begin
  // ParamCount : LongInt -- служебная переменная из системного модуля;
  // хранит количество аргументов командной строки
  // самым первым аргументом считается имя программы

  if ParamCount <> 2 then
    writeln(HELP_BRIEF)

  // запрос справки от пользователя
  else if (ParamCount = 1) and (
    (ParamStr(1) = '--help') or
    (ParamStr(1) = '-h')) then
    writeln(HELP)

  // основной алгоритм программы
  else
    begin
      a := StrToFloat(ParamStr(1));
      b := StrToFloat(ParamStr(2));
      // ParamStr(l: LongInt):string; -- возвращает строку,
      // содержащую параметр номер l,

      // todo: проверить значения a и b
      c := sqrt(a*a + b*b);

      writeln('Гипотенуза: ', c:0:2);
    end;
end.

```

Запуск программы из командной строки:

```

> .\hypotenuse.exe
hypotenuse <a> <b>;
Вычисляет длину гипотенузы по длинам катетов a и b

> .\hypotenuse.exe 3 4
Гипотенуза: 5.00

```

2.3 Парадигмы интерфейса

Существуют три основных парадигмы интерфейса [18]:

- метафорическая,
- идиоматическая,
- парадигма реализации.

Интерфейсы, построенные согласно парадигме реализации, опираются на понимание того, как работает продукт, то есть основываются на модели реализации, но не на модели представления. Интерфейсы могут сочетать в себе все три парадигмы. В самом своём начале ЧМВ как дисциплина в основном полагались на понимание работы программы пользователем. До сих пор многие программы с графическим интерфейсом следуют такому принципу. Они имеют по кнопке на каждую функцию и по диалоговому окну на каждый модуль кода, а их команды и процессы являются точным отражением внутренних структур данных и алгоритмов. Подобные программы просто создавать и тестировать. Но они никак не стремятся помочь пользователю, заставляя его тратить больше времени на изучении программы, а не на решении своих задач в ней. Иногда такой подход приводит к созданию сайтов, которые повторяют бизнес-процессы и структуру фирмы, которой принадлежит сайт.

Метафорические интерфейсы основаны на интуитивных представлениях о работе продукта, то есть на узнавании принципов и элементов заложенных в интерфейс. Такие интерфейсы полагаются на модель представления. Многие понятия и элементы интерфейса в современных программах построены на метафорах: папка, рабочий стол, корзина, переключатель (radio button) и т.д. Яркий пример интерфейса повсюду следующего этому подходу – Microsoft Bob (рис. 2.8). Приложения, встроенные в Bob, представлены соответствующими элементами интерьера: например, при нажатии на часы открывает календарь, а ручки и бумага представляют программу составления писем. Часто приводится требование к интерфейсу – «интуитивно понятный», то есть простой в использовании или простой для понимания. Однако это требование ничего не говорит о том, на сколько эффективно с программой может взаимодействовать не новичок.

Дизайн интерфейсов двухтысячных годов широко использовал скевоморфизм. **Скевоморфизм** – это тенденция в дизайне, в основе которой лежит реалистичное изображение объектов. В скевоморф-



Рис. 2.8. Метафорический интерфейс. Microsoft Bob был выпущен для Windows 3.1. Приложения, встроенные в Bob, представлены соответствующими элементами интерьера: например, при нажатии на часы открывает календарь, а ручки и бумага представляют программу составления писем.



Рис. 2.9. Скевомофизм в iOS: приложение калькулятор подражает калькулятору Braun, созданному Дитером Рамсом для фирмы Braun.

ной графике показан объём предметов: свет, тени, блики и текстуры (рис. 2.9).

Идеоматические интерфейсы основаны на обучении пользователя тому, как достичь результата, что является для людей естественным процессом. Идиоматическое проектирование, которое Тед Нельсон (Ted Nelson) назвал «проектированием принципов», основано на том, как человек изучает и применяет идиомы и речевые обороты. Идиоматические пользовательские интерфейсы решают проблемы, с которыми не справляются предыдущие два типа интерфейсов, поскольку сосредоточиваются не на технических знаниях и не на интуитивных представлениях о функциях, а на изучении простых, неметафорических визуальных и поведенческих идиом, необходимых пользователю для достижения целей и решения задач.

Многие элементы интерфейса – это скорее идиомы чем метафоры: окно, заголовок окна, кнопка закрытия, гиперссылка, выпадающий список. Компьютерная мышь – эта метафора, которая описывает внешний вид устройства, но не принцип его использования, однако пользователь легко может понять принцип её работы – идиоматическое освоение.

Идиомы легко запоминаются и их можно понять их контекста. Большая часть того, что мы знаем, усвоена нами без понимания: лица людей, общественные и личные отношения, мелодии, названия брендов, расположение комнат и мебели в нашем доме или офисе. Мы не понимаем, почему чье-то лицо выглядит так, а не иначе; мы просто знаем это лицо.

2.4 Модальность

Жест — действие или последовательность действий, которые человек не разделяет на составляющие, а выполняет как бы единым движением. Для опытного пользователя ввод короткого слова с клавиатуры — один жест. Для начинающего отдельным жестом будет нажатие каждой клавиши [1].

Один и тот же жест может вызывать разные действия в разных состояниях (режимах) интерфейса. Кнопка ↲ в текстовом редакторе начинает новую строку, а в чате отправляет сообщение. Педаль газа обычно значит «ехать вперёд», но если включена задняя передача, то уже «ехать назад».

Интерфейс называется **модальным**, если в нём есть состояния, которые человек не осознаёт во время жеста, но в которых этот жест интерпретируется по-разному.

Ошибка, совершенная человеком из-за того, что он не осознавал, в каком состоянии находился интерфейс, и получил не тот результат жеста, которого ожидал, называется модальной ошибкой⁴.

Модальная ошибка привела к крушению самолёта рейса 214 «Азиана-эйрлайнс» в июле 2013 года. Пилот не осознавал, что в текущем режиме автопилота не работает автомат тяги⁵, и продолжал управлять самолётом, не следя за скоростью [1].

⁴о двух видах ошибок говорится в п.5 параграфа 5.1 Юзабилити

⁵автомат тяги — это система, которая в автоматическом режиме управляет тягой двигателей



Рис. 2.10. Курсоры и значки на иконках, показывающие режим. В iOS, в этом режиме иконки подрагивают [1].

Режим часто не является локусом внимания пользователя. Поэтому мы часто ошибаемся вводя текст не переключив раскладку. Если режим плохо считывается пользователем, то это приводит к непредсказуемой (с точки зрения пользователя) работе программы и увеличению числа ошибок. Если при проектировании интерфейса нельзя избежать режимов, то стоит сделать их заметными. Например изменить указатель (рис. 2.10) или добавить новые значки и анимацию на элементы интерфейса.

Квазирежимом Джек Раскин называет такое состояние интерфейса, в котором пользователь его удерживает физически. Квазирежим невозможно не заметить.

Кнопка CapsLock переключает между режимами ввода строчных и прописных букв. Это приводит к модальным ошибкам: если забыть отключить, по ошибке пишешь большими буквами.

Кнопка Shift включает квазирежим ввода прописных букв — буквы пишутся прописными лишь пока кнопка зажата. Ввод отдельной прописной с шифтом воспринимается человеком не как работа в другом режиме, а как использование другого жеста. Если же человеку понадобится ввести заглавными целое слово, то он будет постоянно физически ощущать особый режим клавиатуры.

Вопросы

1. Что такое интерфейс? Пользовательский интерфейс (ПИ)?
2. Относятся ли к пользовательскому интерфейсу статичные элементы (текст, изображения) в окне программы или на веб-странице?
3. Относятся ли к пользовательскому интерфейсу горячие клавиши программ?
4. Приведите классификацию видов ПИ. Приведите примеры для каждого вида ПИ.
5. Какие виды пользовательского интерфейса реализованы в вашем телефоне?
6. Что такое командный интерфейс?
7. Опишите достоинства и недостатки графического и текстового интерфейса.
8. Какие парадигмы пользовательского интерфейса вы знаете?
Опишите каждую из них.
9. Что такое скевоморфизм?
10. Что такое модальность? Что такое квазирежим?
11. Как режимы сказываются на количестве ошибок, которое совершает пользователь? Как снизить это количество?

Литература

- Раскин Д., Интерфейс: новые направления в проектировании компьютерных систем. – Пер. с англ. – СПб: Символ-Плюс, 2007. – 272 с.
- Бирман И. Б. Пользовательский интерфейс. – М.: Изд-во Бюро Горбунова, 2017.
- Алан Купер, Рейманн Роберт М., Основы проектирования взаимодействия. – Пер. с англ. – СПб.: Символ-Плюс, 2018, 720 с.

3 Понятие дизайна и визуальное восприятие

3.1 Общие представления о дизайне

Точного определения понятия дизайн не существует. Поэтому приведём наиболее общее.

Дизайн – комплексный инструмент создания и оптимизации многосторонних потребительских качеств продукта – изделий, услуг, процессов и среды, – наиболее полно отвечающих потребностям человека и общества [22]. Широта определения обусловлена большим количеством отраслей дизайна (рис. 3.1).

Дизайн часто сравнивают с близкой областью – искусством. Но в отличии от последнего, дизайн всегда призван решать некую задачу.

Стоит подчеркнуть, что дизайн в широком смысле не ограничивается только эстетическими качествами продукта. Например, рекламный баннер должен быть не только эстетичным, но и в понятной форме доносить нужную информацию до человека. Быть заметным и, наконец, выполнить свою основную функцию – заинтересовать потенциального клиента.

Дизайн не обязательно предполагает статичность. Дизайн – это то не то, как предмет выглядит, а то, как он работает¹. Дизайн взаимодействия с пользователем (отдельная отрасль дизайна) – хороший тому пример. Модель представления – тоже объект дизайна.

¹известная цитата Стива Джобса – не определение дизайна, а акцент на отдельный аспект понятия

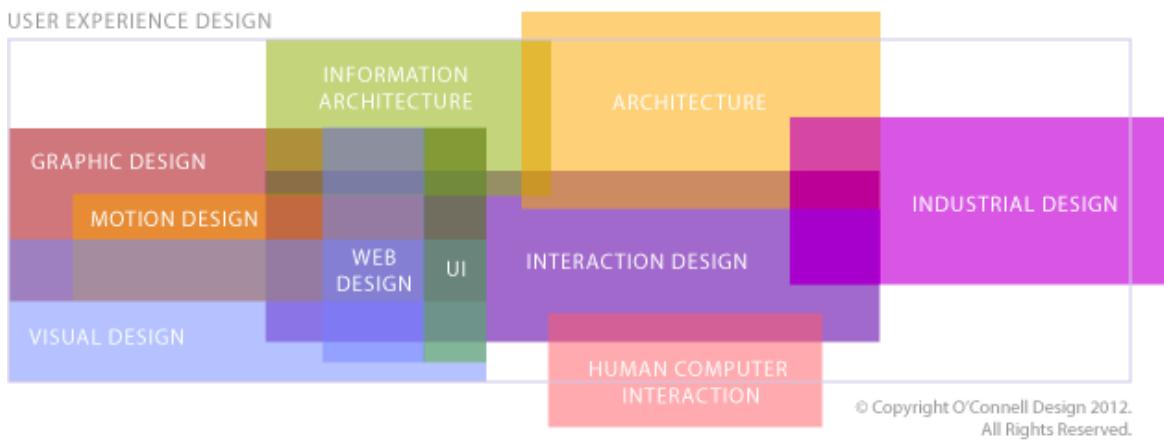


Рис. 3.1. Отрасли дизайна

Дизайн-система – набор компонентов, правил, предписаний инструментов для повышения качества и скорости разработки продуктов, а также эффективной поддержки существующих.

Дизайн-система помогает систематизировать работу дизайнера за счёт наборов правил, сделать дизайн большого продукта или нескольких продуктов одной компании узнаваемым и целостным.

Дизайн-система может в себя включать:

- Руководство по стилю, в том числе цветовые схемы (рис. 3.2);
- Библиотеку готовых компонентов (UI-кит), шаблоны;
- Наборы UX-паттернов (как организовывать навигацию, диалоговые окна, формы и т.д.);
- Документацию, правила, рекомендации (например material design [49], Apple [45]).

Некоторые компании публикуют свои дизайн-системы, например mail.ru (рис. 3.3) или Google (Material Design, рис. 3.4). Готовые наборы базовых элементов интерфейса помогают ускорить разработку макетов. Многие дизайнеры публикуют такие наборы, в частности их можно найти на странице figma.com/community.

Дизайн-система часто содержит наборы правил для расположения элементов интерфейса, шрифтов, способов ввода данных и т.д. В Интернете существует большое число рекомендаций относительно дизайна вообще и визуального дизайна в частности. Стоит отно-

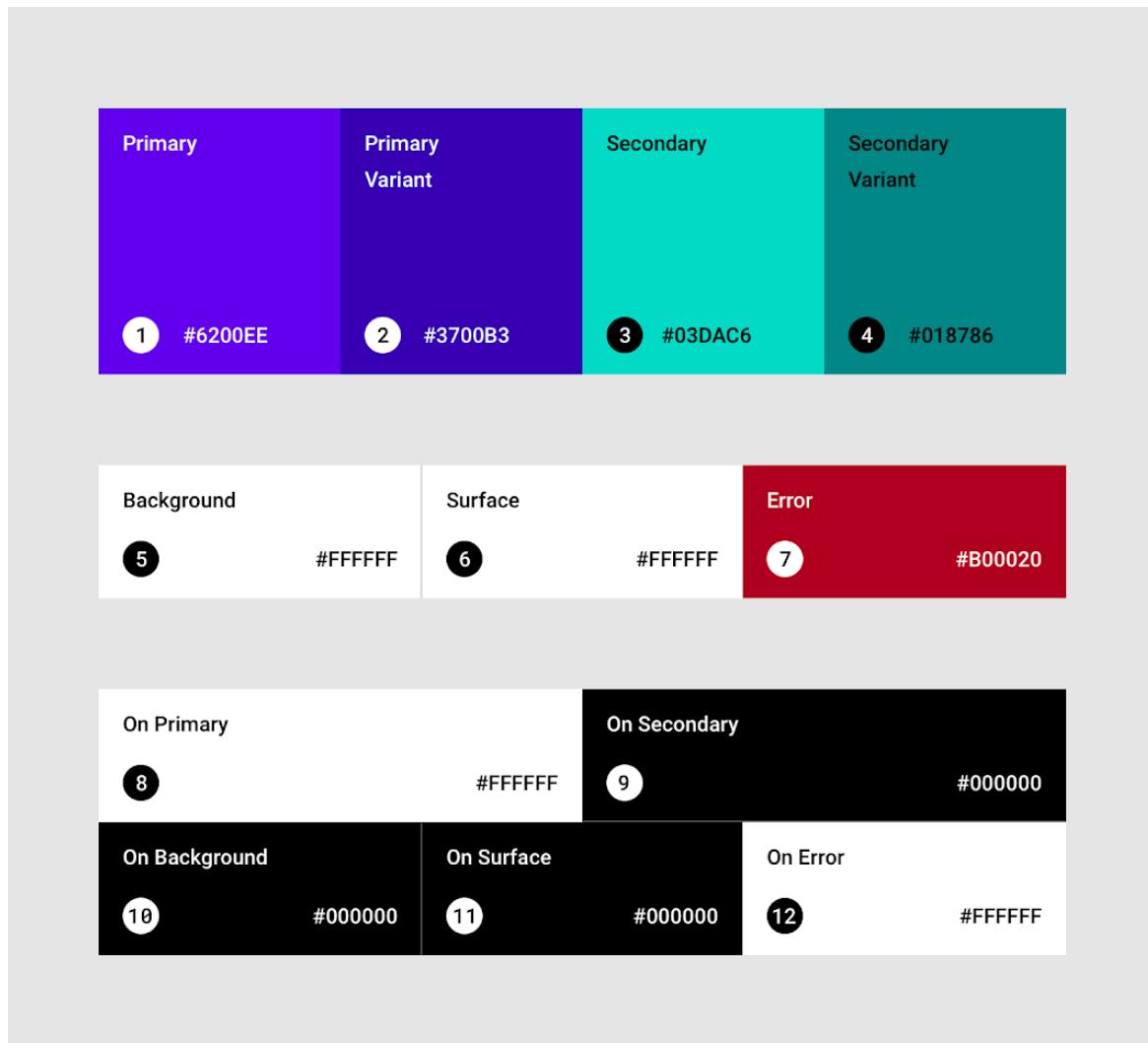


Рис. 3.2. Цветовая схема рекомендованная Material Design

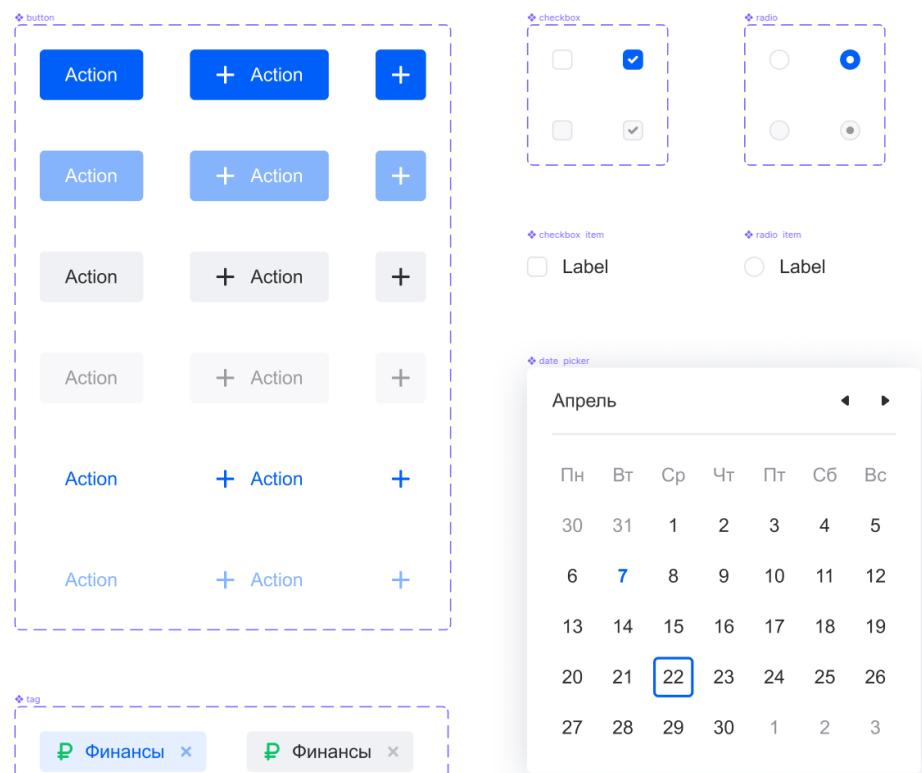


Рис. 3.3. Фрагмент библиотеки готовых компонентов дизайна-системы Paradigm от mail.ru.

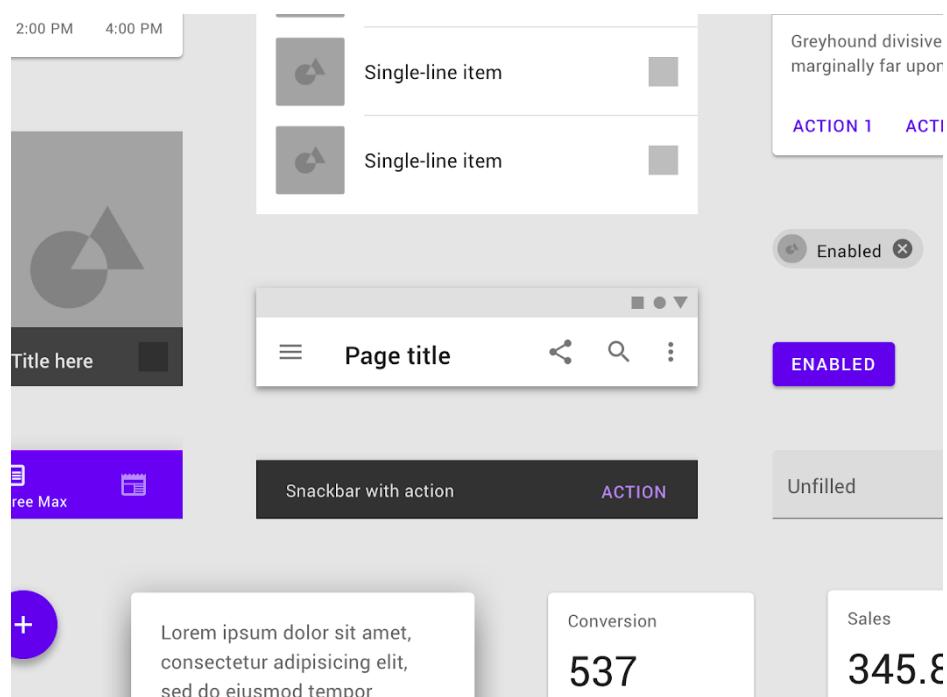


Рис. 3.4. Фрагмент библиотеки готовых компонентов дизайна-системы Material Design

сится к ним в долей скепсиса и рассматривать в лучшем случае как лучшую практику (*best practice*), так как большинство материалов не содержат ссылки на научные публикации и отдельные исследования специалистов. Например широко распространено утверждение “шрифты с засечками читаются легче, чем шрифты без засечек” (о шрифтах и типографике говорится в разделе 7). Однако убедительных доказательств этому нет.

3.2 Зрительное восприятие

Как было отмечено, запечатление окружающего мира человека условно можно разделить на три уровня:

- ощущение – отражение отдельных свойств и предмета;
- восприятие² – целостный образ совокупности свойств предмета;
- представление – сохранившийся в сознании образ предмета, который воспринимался раньше.

Дизайн призван воздействовать на человека на всех трёх уровнях.

Гештальтпсихология (нем. Gestalt – личность; образ, форма) – общецисиологическое направление, связанное с попытками объяснения прежде всего восприятия, мышления и личности. В качестве основного объясняющего принципа гештальтпсихология выдвигает принцип целостности. Первичными данными психологии являются целостные структуры – гештальты. Примером принципа целостности может служить треугольник Канижа (рис. 3.5). На рисунке нет четко отчерченного белого треугольника, но человеческое восприятие достраивает его.

Примером противоположной работы восприятия может служит расстройство восприятия – предметная агнозия: человек видит пред-

²зрительное восприятие – целостный образ совокупности свойств предмета построенный на основе зрительной информации

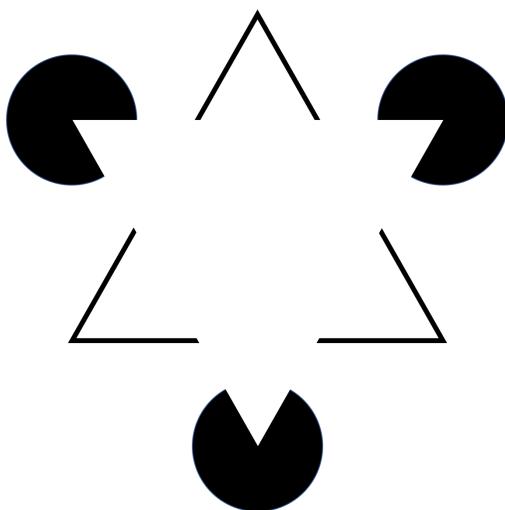


Рис. 3.5. Треугольник Канижа.

На изображении нет четко отчерченного белого треугольника, но человеческое восприятие достраивает его.

меты как сумму отдельных частей, но не может составить целостный образ предмета³.

Целостность восприятия и его упорядоченность достигаются благодаря следующим принципам:

- Близость (Law of Proximity) – стимулы, расположенные рядом, имеют тенденцию восприниматься вместе (рис. 3.6);
- Схожесть (Law of Similarity) – стимулы, схожие по размеру, очертаниям, цвету или форме, имеют тенденцию восприниматься вместе;
- Целостность – восприятие имеет тенденцию к упрощению и целостности;
- Замкнутость – отражает тенденцию завершать фигуру так, что она приобретает полную форму;
- Смежность – близость стимулов во времени и пространстве. Смежность может предопределять восприятие, когда одно событие вызывает другое;
- Общая зона – принципы гештальта формируют повседневное восприятие наравне с обучением и прошлым опытом; предвосхищающие мысли и ожидания также активно руководят нашей интерпретацией ощущений.

³расстройства восприятия описаны в научно-популярной книге Оливера Сакса «Человек, который принял жену за шляпу»

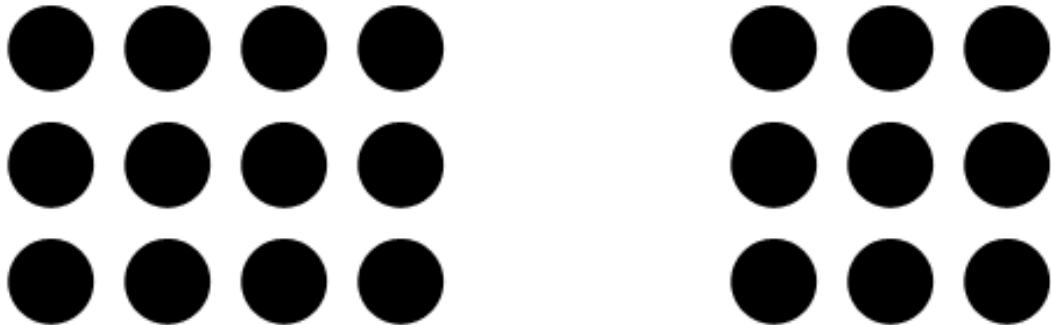


Рис. 3.6. Пример восприятия согласно принципу близости

Принцип близости, называемый ещё теорией близости, возможно один из самых важных принципов восприятия в дизайне. Мы чаще всего неосознанно связываем близко расположенные объекты по смыслу, а удалённые – нет. Очевидный факт: текст расположенный ближе всего к картинке воспринимается как её название или подпись к ней (рис. 3.7), на самом деле следствие психического закона восприятия.

Дilemma делать ли подпись к картинке снизу или сверху, имеет простое разрешение в первом приближении – подпись должна быть ближе к своей картинке, чем ко всем остальным.

Если в дизайне есть чёткая и ясная система задающая расположение объектов, то человек её почти наверняка заметит. Поэтому в дизайне широко используются размещение объектов по сетке (см. параграфы 4.2.7 и 4.2.9). Например поэтому, в таблицах не всегда нужно изображать линии, разделяющие строки и столбцы, если выравнивание текста говорит само за себя (рис. 3.5).

Большое значение имеет пустота, «воздух» – он отделяет объекты друг от друга. Расстояние между отдельными группами элементов – большое, между элементами в группе – меньше (рис. 3.8). Этот подход помогает выстраивать визуальную иерархию (рис. 3.9).

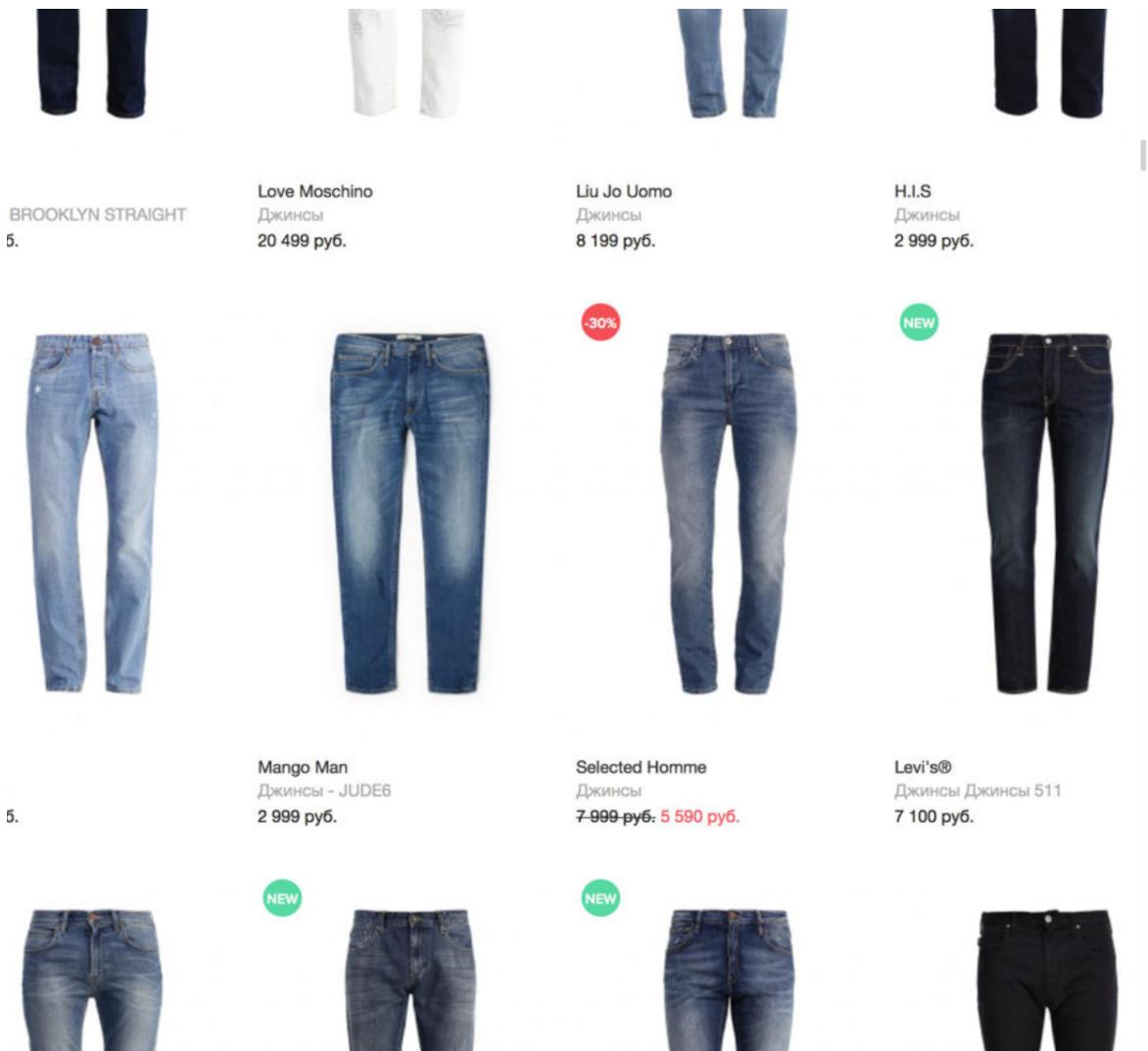


Рис. 3.7. Дизайнер не использовал принцип близости: нельзя понять к каким изображениями относятся подписи. Нужно сделать подписи ближе к связанным изображениям.

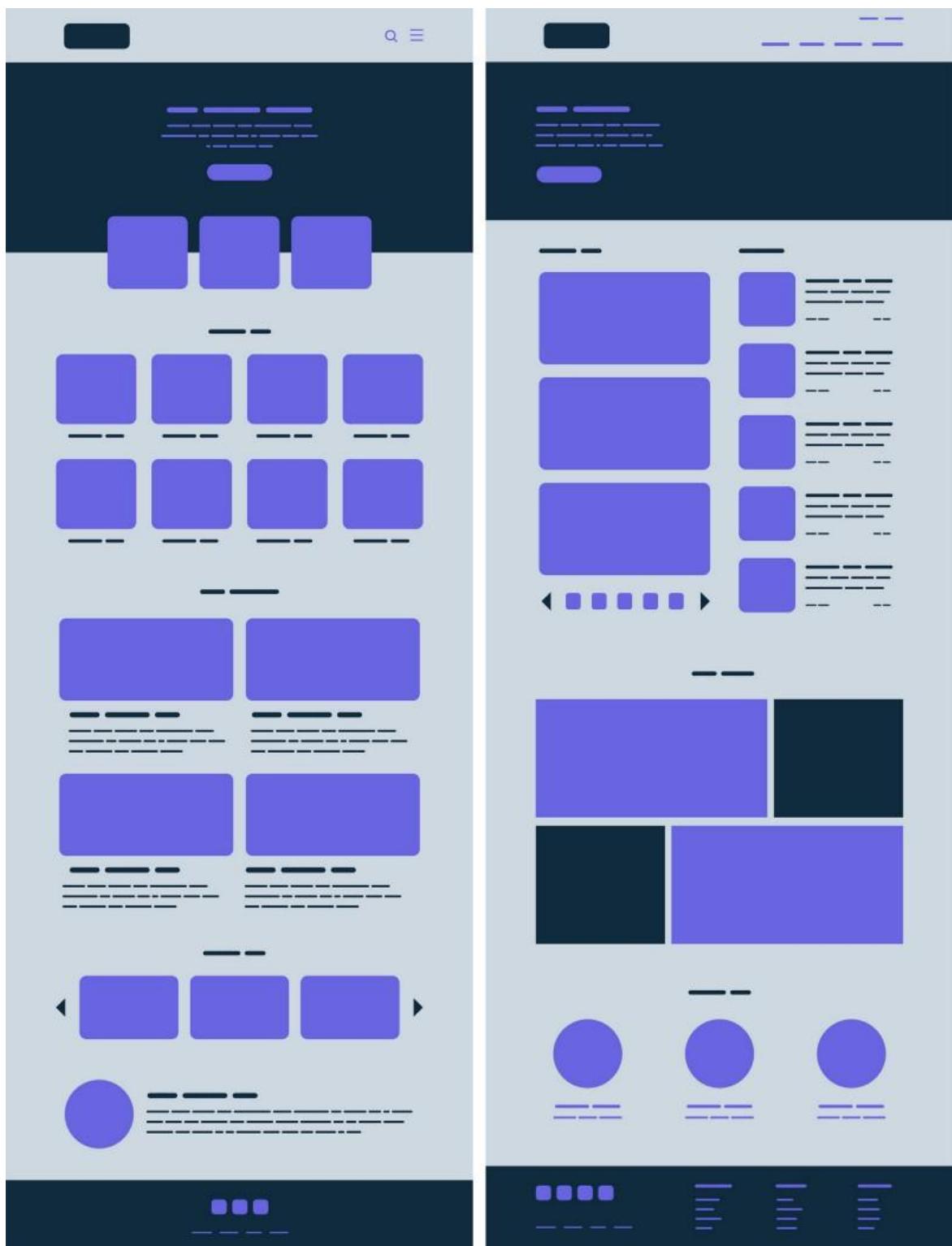


Рис. 3.8. Принцип близости в грубом макете интерфейса. Большое значение имеет пустота, воздух – он отделяет объекты друг от друга. Расстояние между отдельными группами элементов – большое, между элементами в группе – меньше.

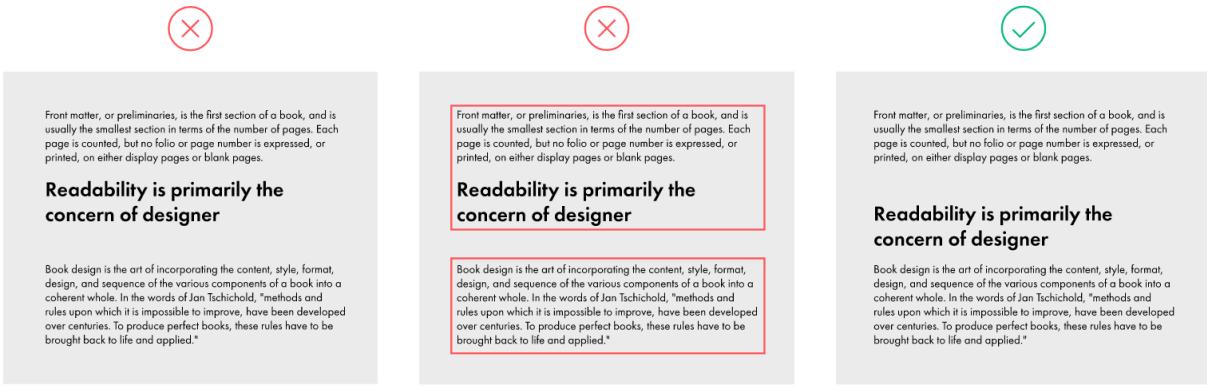


Рис. 3.9. Частая ошибка в визуальной иерархии: заголовок текста “висит” между абзацами, в то время как должен быть ближе к следующему абзацу, к которому он и относится.

М О С К О В С К И Й М Е Т Р О П О Л И Т Е Н

М О С К О В С К И Й М Е Т Р О П О Л И Т Е Н

Рис. 3.10. Несоблюдение (сверху) и почти наглядное соблюдение (снизу) следствия принципа близости: внутренние расстояния (между буквами в слове) должны быть меньше внешних (между словами)

Правило «внутренние расстояния должны быть меньше внешних» – следствие закона близости [8]. Важно следить за тем, чтобы в надписях из нескольких строк расстояние между буквами было заметно меньше чем расстояние между строками (рис. 3.10).

За счёт использования принципов непрерывности и близости можно избавиться от лишних графических элементов (рис. 3.12 и 3.13). Колонки таблицы были выровнены и хорошо определяются без разграничитывающих линий, отделяемые друг от друга пустотой (расстоянием)⁴. Равномерный белый фон уменьшает количество визуального мусора. Однако, в больших таблицах стоит сохранять чередование фонового цвета строк или выделять строку, на которую наведён указатель.

⁴Анимация пошагового улучшения таблицы:
<https://raw.githubusercontent.com/ivtipm/HCI/master/etc/table-improvement.gif>

МОСКОВСКИЙ
МЕТРОПОЛИТЕН
ИМЕНИ В.И.ЛЕНИНА

МОСКОВСКИЙ
МЕТРОПОЛИТЕН
ИМЕНИ В.И.ЛЕНИНА

Рис. 3.11. Верху: принцип “внутреннее ≤ внешнее” нарушен: расстояния внутри слова (между буквами) и особенно между словами надписи больше чем между словами и границами области. Текст на нижнем изображении выглядит более целостным, чем на верхнем

| Role | Name | Year of the... | Debut | Number of Fans | Takedown Rate |
|----------------------|------------------------------------|----------------|----------|----------------|---------------|
| Face (The Hero) | The Ultimate Warrior | Tiger | May-2011 | 97320.00 | 86.2 |
| Face (The Hero) | Hulk Hogan | Oxen | Jan-2008 | 988551.00 | 61.978 |
| Face (The Hero) | Macho Man Randy Savage | Monkey | Feb-2008 | 157618.00 | 59.29 |
| Face (The Hero) | Hacksaw Jim Duggan | Pig | Mar-2008 | 30300.00 | 53.4332 |
| Face (The Hero) | Superfly Jimmy Snuka | Dragon | Mar-2008 | 12341.00 | 52.7 |
| Heel (The Bad Guy) | Rowdy Roddy Piper | Rooster | Jun-1968 | 71645.00 | 45.4 |
| Heel (The Bad Guy) | The Million Dollar Man Ted DiBiase | Rat | Apr-1975 | 449342.00 | 43.7689 |
| Heel (The Bad Guy) | Mr. Perfect Curt Henning | Rat | May-1980 | 13773.00 | 38 |
| Heel (The Bad Guy) | Jake the Snake Roberts | Snake | Jul-1975 | 5609.00 | 37.99 |
| Jobber (The Unknown) | Brad Smith | Sheep | Aug-2008 | 1103.00 | 36.316 |
| Jobber (The Unknown) | Ted Duncan | Sheep | Aug-2008 | 200.00 | 33.61 |
| Jobber (The Unknown) | Joey the Uber Nerd Cherdarchuk | Snake | Aug-2008 | 5.00 | 21.0196 |

Рис. 3.12. Таблица с ошибками в дизайне. Текст выровнен по центру, что приводит к необходимости разделять колонки линиями, что, в свою очередь, увеличивает количество визуального мусора. Числа выровнены по центру, а не по разрядам (по правому краю).

| Role | Name | Year of the... | Debut | Thousands of Fans | Takedown Rate |
|----------------------|------------------------------------|----------------|----------|-------------------|---------------|
| Face (The Hero) | The Ultimate Warrior | Tiger | May-2011 | 97.3 | 86.2 |
| | Hulk Hogan | Oxen | Jan-2008 | 988.6 | 62.0 |
| | Macho Man Randy Savage | Monkey | Feb-2008 | 157.6 | 59.3 |
| | Hacksaw Jim Duggan | Pig | Mar-2008 | 30.3 | 53.4 |
| | Superfly Jimmy Snuka | Dragon | Mar-2008 | 12.3 | 52.7 |
| Heel (The Bad Guy) | Rowdy Roddy Piper | Rooster | Jun-1968 | 71.6 | 45.4 |
| | The Million Dollar Man Ted DiBiase | Rat | Apr-1975 | 449.3 | 43.8 |
| | Mr. Perfect Curt Henning | Rat | May-1980 | 13.8 | 38.0 |
| | Jake the Snake Roberts | Snake | Jul-1975 | 5.6 | 38.0 |
| Jobber (The Unknown) | Brad Smith | Sheep | Aug-2008 | 1.1 | 36.3 |
| | Ted Duncan | Sheep | Aug-2008 | 0.2 | 33.6 |
| | Joey the Uber Nerd Cherdarchuk | Snake | Aug-2008 | 0.0 | 21.0 |

Рис. 3.13. Таблица после исправления ошибок в дизайне.

Разделяющие линии могут восприниматься как визуальный шум⁵ и способствовать обратной задаче – объединения разных групп объектов (рис. 3.14).

Принцип схожести, в частности, помогает незаметно для части пользователей помешать рекламные ссылки в поисковую выдачу (рис. 3.15).

Восприятие способно не только дополнять увиденное, но и иска- жать. На этом построены многие оптические иллюзии. Поэтому в дизайне следует руководствоваться не только правильными геометрическими построениями, но и учитывать то как это воспри- нимается человеком (рис. 3.16). На рисунке правый треугольник расположен так, что половина его площади лежит слева от верти- кальной оси симметрии квадрата. Буквы с закруглёнными частями (рис. 3.17) часто увеличивают в размерах, чтобы они воспринимались одинаковыми по высоте с буквами без закруглённых частей. Слиш- ком большой контраст между основными элементами и фоном, плотное расположение может вызывать эффект решётки Германа: в местах между ячейками решётки видны серые пятна (рис. 3.18).

⁵см. также понятие data-ink ratio

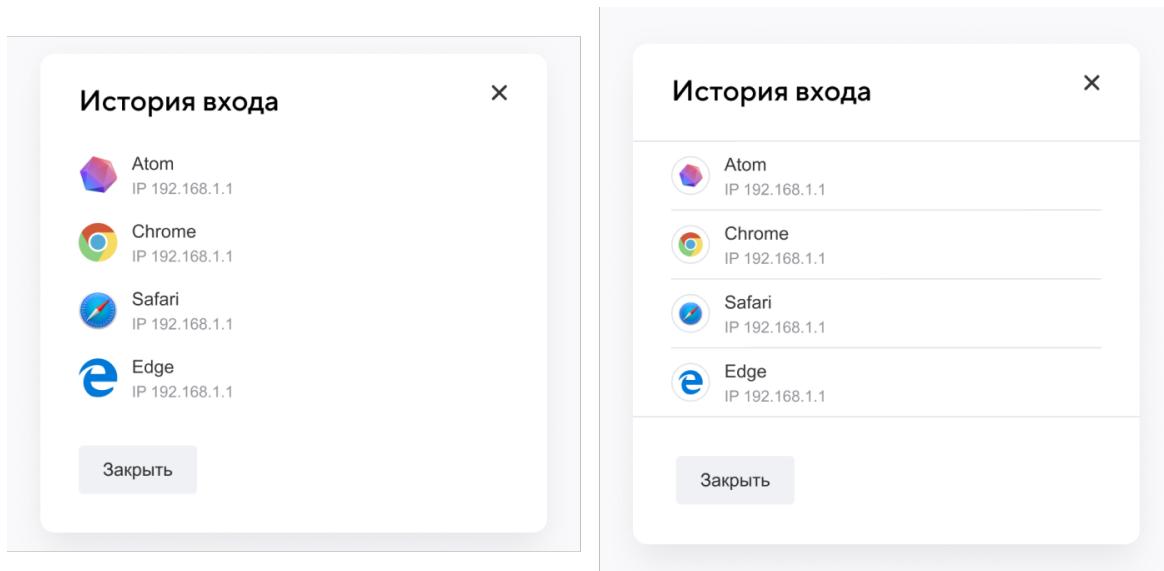


Рис. 3.14. Отделение объектов друг от друга с помощью только пустого пространства лучше, чем с помощью разделителей. Они могут восприниматься как визуальный шум и способствовать обратной задаче – объединения разных групп объектов. Изображение из [52]

11) Проектирование Интерфейсов - 12. Основы проектирования- 1.2 Азбука...

видео, поделиться, телефон с камерой, телефон с видео, бесплатно, загрузить...

YouTube · 22 декабря 2021

1. Проектирование интерфейсов. Введение | Технострим

Приносим извинения за плохой звук по техническим причинам. Взаимодействие проектировщиков и других специалистов (любовь и ненависть). Что такое юзабилити и удобство **интерфейса**. Что такое хороший...

YouTube · 13,9 тыс. просмотров · 29 марта 2017

Совершенный код Практическое руководство. 1 4...

Более 10 лет первое издание этой книги считалось одним из лучших практических...

chitai-gorod.ru · реклама | 16+

Яндекс Плюс - 60 дней за 0 ₽! Кинопоиск и Яндекс...

Кинопоиск и Яндекс Музыка в подписке Яндекс Плюс. Ощутите все преимущества Плюса!

plus.yandex.ru · реклама

Рис. 3.15. Принцип схожести используется чтобы выдать реклама в поисковой выдаче за результат поиска

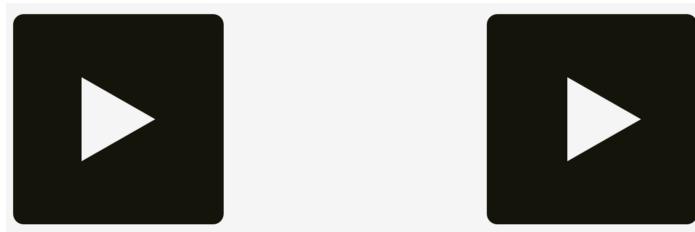


Рис. 3.16. Выравнивание фигур по геометрическому центру (слева) и по визуальному центру (справа). Правый треугольник расположен так, что половина его площади лежит слева от вертикальной оси симметрии квадрата.



Рис. 3.17. Буквы без выносных элементов, но с округлыми элементами слегка выходят за базовую линию (нижняя линия) и за медиану (верхняя линия) чтобы их размер не воспринимался меньшим, чем он есть на самом деле

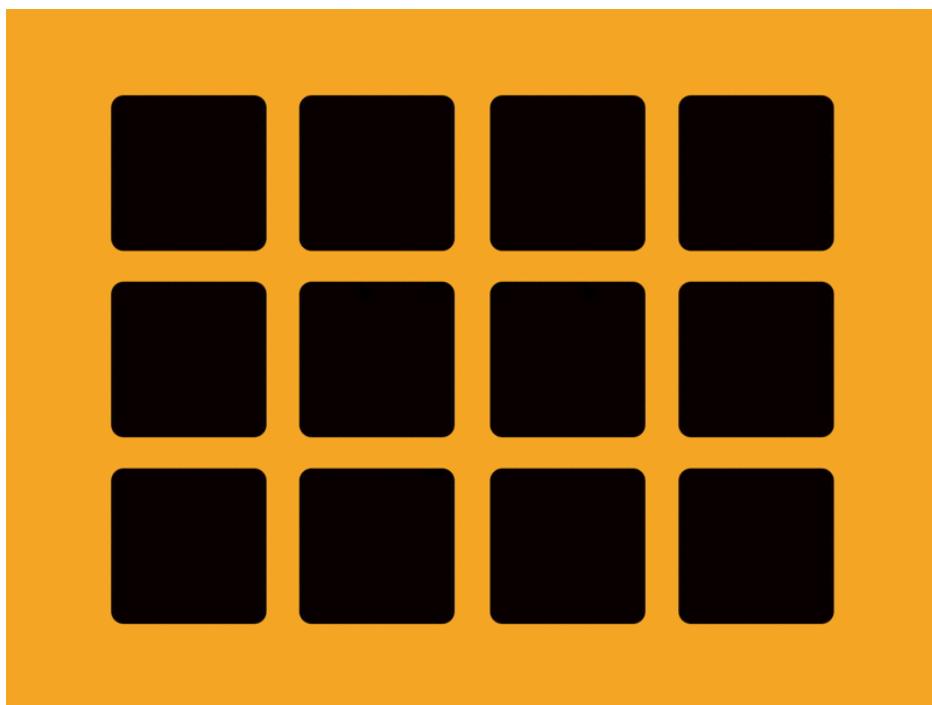


Рис. 3.18. Решётка Германа: места на пересечении линий, разделяющих квадраты, воспринимаются серыми крыгами

3.3 Цветовые модели

Модель RGB (red, green, blue — красный, зелёный, синий) – аддитивная⁶ цветовая модель (рис. 3.19), описывающая способ кодирования цвета для цветовоспроизведения с помощью трёх цветов, которые принято называть основными.

Интенсивность каждого цвета представляется в виде одного октета⁷, значения которого обозначаются для удобства целыми числами от 0 до 255 включительно, где 0 — минимальная, а 255 — максимальная интенсивность. Октеты часто записывают подряд в шестадцатеричной кодировке. Например **#395fb6** представляет соответственно интенсивности красного, зелёного и синего в десятичной системе счисления 57, 95, 182.

Такое представление формально соответствует набору колбочек в глазу человека – клеток, чувствительных к красному зелёному и синему свету. Это же представление широко распространено в программном обеспечении и устройствах.

Но с точки зрения дизайнера, этот способ кодирования цвета не всегда удобен. Часто приходится изменять насыщенность и яркость конкретного цвета. Но как сделать цвет с кодом #395fb6 светлее? Как сделать этот цвет менее насыщенным? Необходимые изменения сходу трудно выразить в виде изменения интенсивности красного, зелёного и синего без дополнительных вычислений.

Модель HSV призвана решить этот недостаток. HSV (Hue, Saturation, Value — тон, насыщенность, значение) или HSB (англ. Hue,

⁶в аддитивной цветовой модели базовый цвет – чёрный (#000000); остальные цвета получаются после добавления интенсивности красного, зелёного и синего; смешение максимальных интенсивностей всех трёх в пропорции 1:1:1 даёт белый цвет (#ffffff). Аддитивные модели используются для испускаемого света (например в дисплеях), субстративные (например CMYK) для отражённого, например в печати.

⁷октет – 8 двоичных разрядов, байт

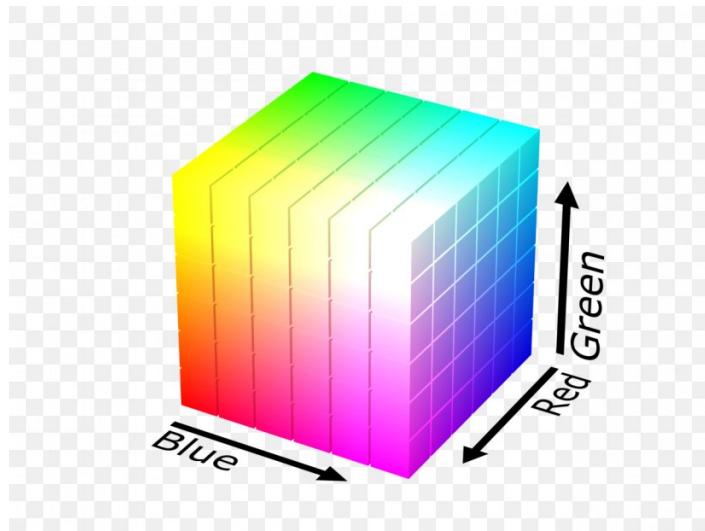


Рис. 3.19. Три цветовые координаты в модели RGB. Скрытая вершина куба окрашена в чёрный.

Saturation, Brightness — тон, насыщенность, яркость) — цветовая модель, в которой координатами цвета являются (см. рис 3.20):

- Hue — цветовой тон, (например, красный, зелёный или синеголубой). Варьируется в пределах 0—360°, однако иногда приводится к диапазону 0—100 или 0—1.
- Saturation — насыщенность. Варьируется в пределах 0—100 или 0—1. Чем больше этот параметр, тем «чище» цвет, поэтому этот параметр иногда называют чистотой цвета. А чем ближе этот параметр к нулю, тем ближе цвет к нейтральному серому.
- Value (значение цвета) или Brightness — яркость. Также задаётся в пределах 0—100 или 0—1.

Модель была создана Элви Рэем Смитом, одним из будущих сооснователей Pixar, в середине 1970-х. Она является нелинейным преобразованием модели RGB.

Цвет #395fb6 в представлении HSV: **222°, 69%, 71%**. В такой записи для изменения насыщенности и яркости нужно изменить только по одному параметру. Все современные графические редакторы⁸,

⁸Если в поисковой строке Google записать цвет в шестнадцатиричном RGB формате, например #002FA7 то откроется инструмент Палитра, где в том числе указанный цвет будет записан в других моделях

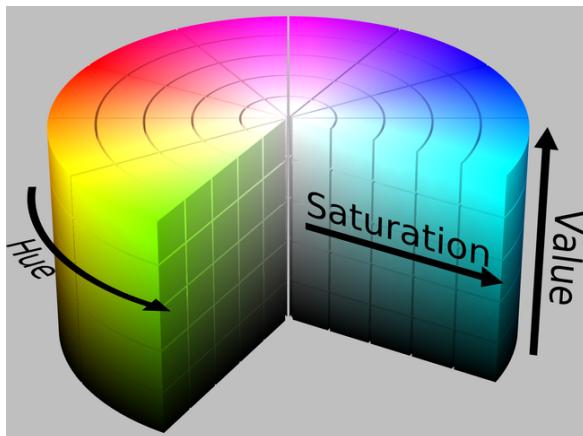


Рис. 3.20. Цветовая модель HSV (HSB). Чтобы изменить только насыщенность (saturation) или только светлоту (value) нужно изменить единственное число в коде цвета.

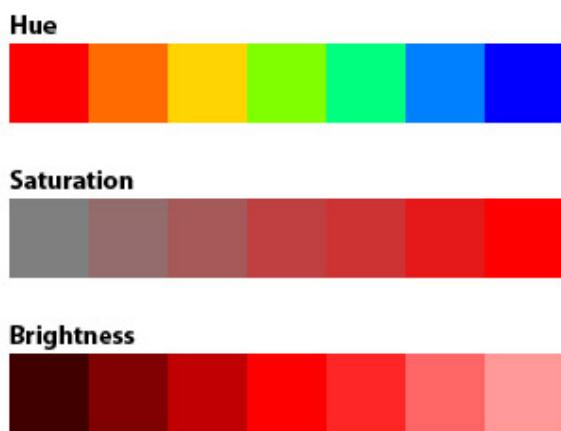


Рис. 3.21. Пример изменения насыщенности (Saturation) и яркости (Brightness) для одного красного тона (Hue)

программы для проектирования интерфейсов могут использовать такой же или похожий подход к заданию цвета. Обычно (рис. 6.13).

3.4 Цвет в интерфейсе

Интерфейс пользователя – это посредник между программой и человеком. Его цель – помочь решить задачу пользователя, но не привлекать к себе внимание и, тем более, не давать большую нагрузку на пользователя. Поэтому в интерфейсах стоит использовать цвет преимущественно как инструмент, помогающий пользователю взаимодействовать с программой (рис. 3.22). Насыщенные и яркие цвета привлекают внимание. Поэтому, такие цвета должны занимать небольшую площадь экрана, привлекая внимание только к самым важным элементам интерфейса. Фон же напротив, не должен привлекать внимания вовсе. При этом текст и другие элементы интерфейса на любом фоне должны давать высокий контраст.

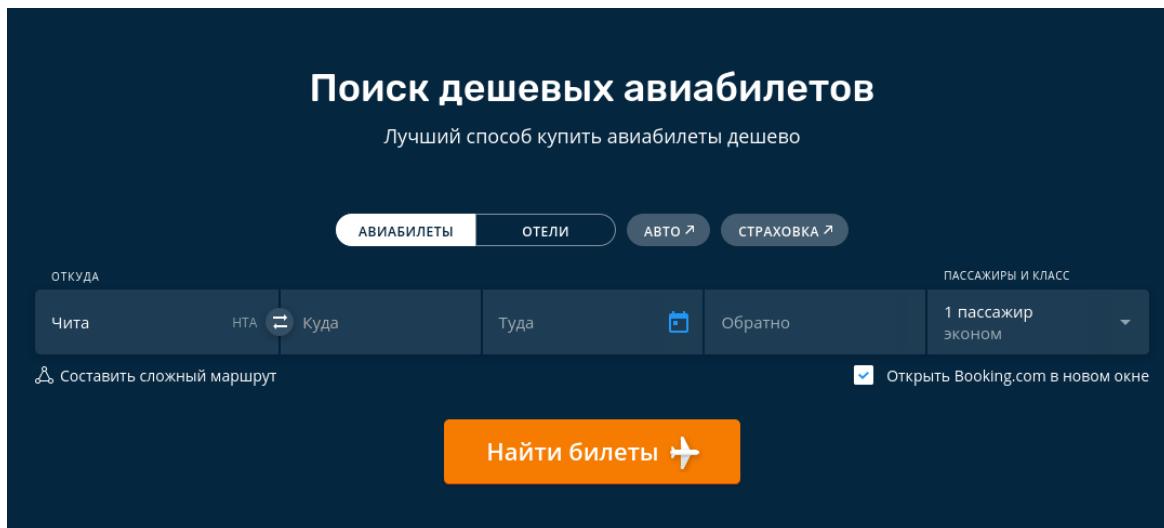


Рис. 3.22. В основе акцентирования внимания цветом – эффект выскакивания. Эффект хорошо работает, когда особых визуальных стимулов (чаще всего фигур) мало.

Дизайн интерфейса Windows 1 (рис. 3.23) был создан преимущественно без участия дизайнеров. Абсолютное большинство современных интерфейсов, несмотря на возросшие технические возможности дисплеев, имеют более сдержанную палитру.

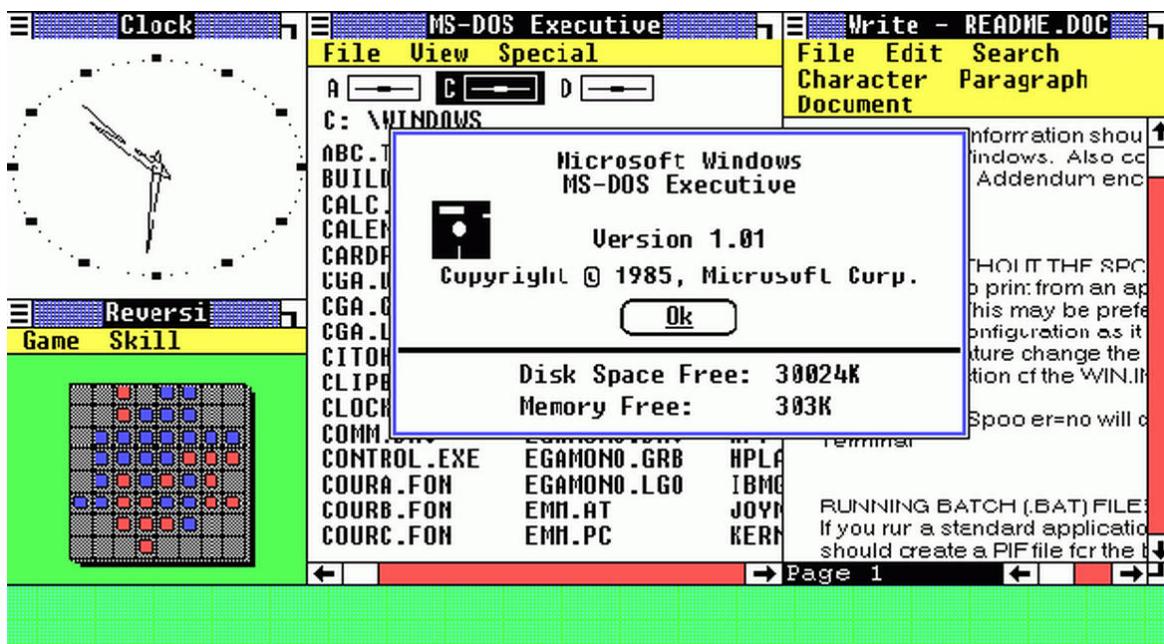


Рис. 3.23. Дизайн интерфейса Windows 1 был создан преимущественно без участия дизайнеров. Абсолютное большинство современных интерфейсов, несмотря на возросшие технические возможности дисплеев, имеют более сдержанную палитру. Задача пользовательского интерфейса – не привлекать к себе внимание, а помочь решить задачу пользователя.

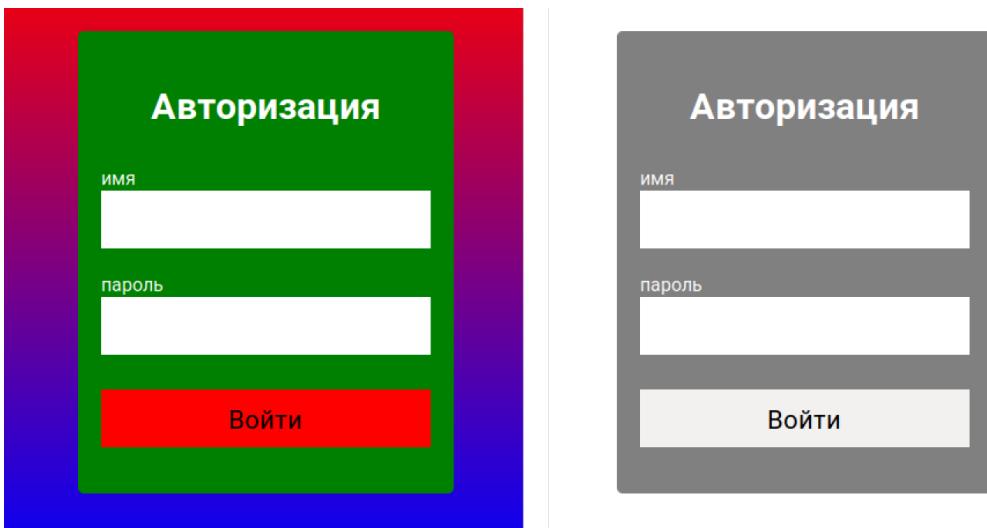


Рис. 3.24. Большое количество визуальных стимулов, даже при сохранении высокого контраста и различимости может проигрывать в эстетическом плане “скучным” цветовым схемам.

Подбор цветовых схем – сложная задача. Отправной точкой могут служить руководства по стилю бренда и платформы, для которой разрабатывается продукт. Например руководства по разработке ПИ для Android или iPhone. Ещё один вариант – это изучение цветовых схем похожих продуктов или макетов на dribbble.com, behance.net, веб-страницах⁹. Для макета или страницы можно составить набор цветов указав их назначение. Какой цвет можно считать основным? Какой цвет используется для привлечения внимания? Какой цвет фона? Как приблизительно соотносятся площади цветовых пятен всех этих цветов? Следует выбирать небольшой набор оттенков и при необходимости изменять яркость и насыщенность.

Вопросы

1. Что такое дизайн?
2. Что такое дизайн-система? Для чего она нужна? Как она может быть представлена?

⁹в браузерах можно воспользоваться инструментами разработчика и посмотреть код цвета в CSS; в Firefox встроен плагин EyeDropper, который показывает цвет пикселя под курсором

3. Какие дизайн системы вы изучали самостоятельно?
4. Что такое ощущение, восприятие и представление? Чем они отличаются?
5. Перечислите принципы целостного восприятия. Опишите каждый из них.
6. Как лучше всего отделять объекты друг от друга?
7. Что такое правило внутреннего и внешнего?
8. Какие существуют модели представления цвета?
9. В чём преимущества цветовых моделей HSV (HSL) перед RGB?
10. Изучите несколько сайтов. Какой цвет использован для фона? Какой цвет привлекает внимание? Какие ещё есть цвета? Как много площади окрашено каждым цветом?

Литература

- Голомбински, К. Добавь воздуха! Основы визуального дизайна для графики, веба и мультимедиа / К. Голомбински, Р. Хаген. — Питер, 2013. — 272 с.
- Горбунов А. С. Типографика и вёрстка. — М.: Изд-во Бюро Горбунова, 2015. - 175 с.