версия от 24 октября 2023 г.

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-машинное 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-машинное 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

ISBN 978-5-9293-3012-4 © Забайкальский государственный университет

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 A/B-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 на столько 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 столиков 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 покупкам [1].

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, котором 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 антипримерами, 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 Канеман[1] 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 модно быстрее вернутся 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-

[1]Канеман – 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, чаше 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 обнаружения[2] 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.

[2]о *выборе* см. раздел 6.3 описывающий закон Хика
[2]см. также preattentive attributes

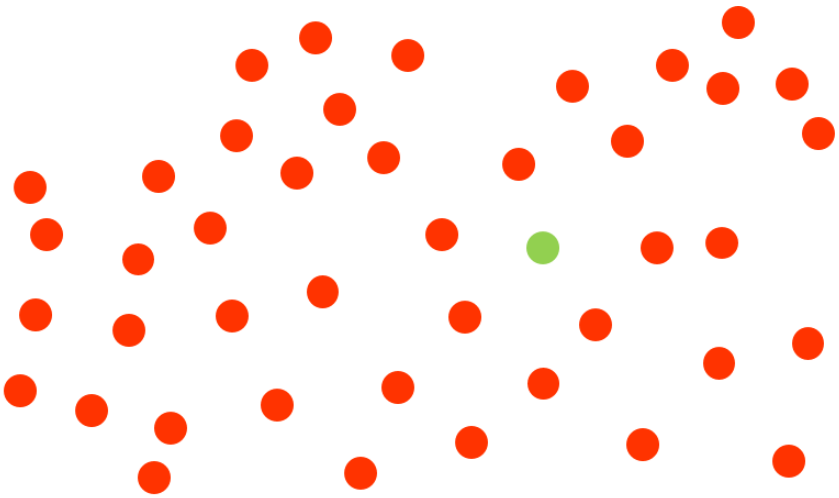


Рис. 1.1. Благодаря *эффекту выскакивания* люди быстро, автоматически находят зелёный круг среди множества других кругов одного цвета.

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 вернутся 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 ***локусом***[3] ***внимания*** (locus of attention) [28].

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

[3] лат. locus — место, область

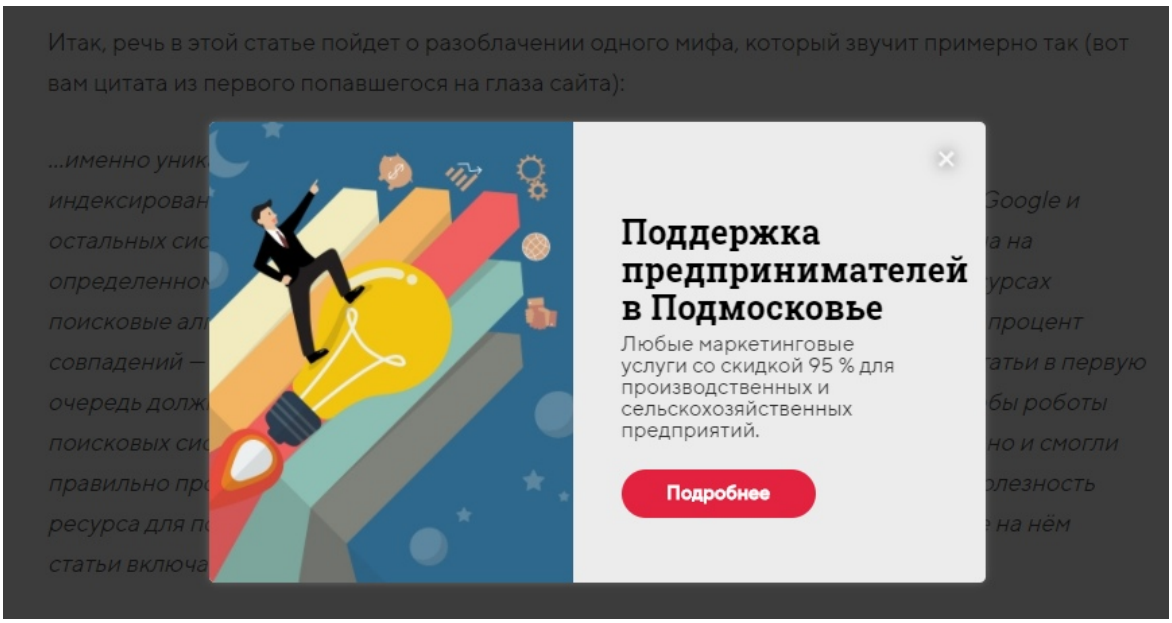


Рис. 1.2. Всплывающее во время просмотра страницы – это *прерывание опыта* – способ привлечь внимание

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 внимание 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 польхователей 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 выбивает 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 получаеются, 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 автоматичных действий и, во-вторых, развивают 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 автоматичных 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-

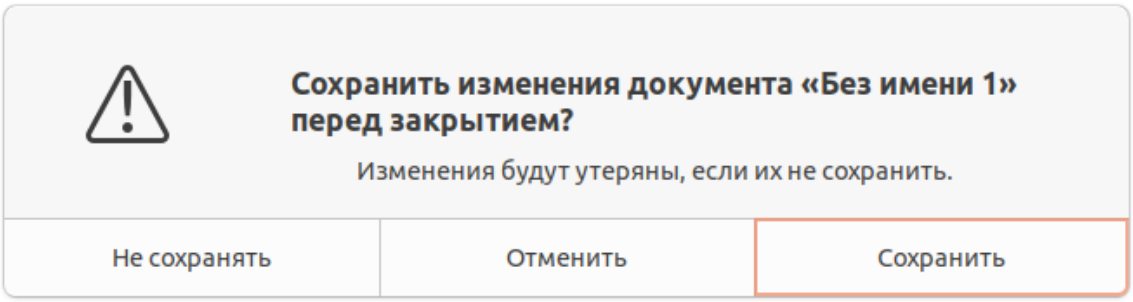


Рис. 1.3. Если дизайн диалоговых окон для типичных запросов содержит одинаковый порядок кнопок, то пользователю будет легко сформировать привычку нажимать на нужную кнопку практически не читая содержимое.

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 OK) 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 смещению локуса внимания пользователя с самой задачи 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 пользо-

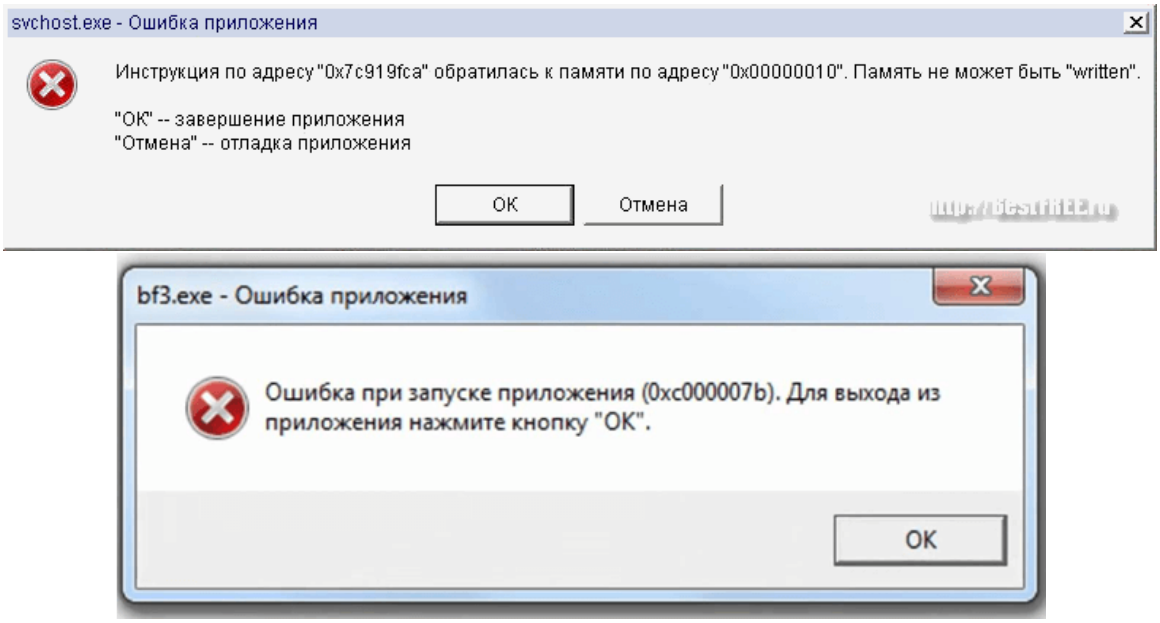


Рис. 1.4. Злоупотребление диалоговыми окнами с бесполезными для пользователя сообщениями часто приводит к формированию привычки закрывать любые диалоговые окна не вникая в содержимое сообщения.

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 автоматичных 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 горючую клавишу.

[4]автор вынужден цензурировать некоторые места учебного пособия, посвящённого исключительно дизайну интерфйесов 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 элемнтов 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 вернутся 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.

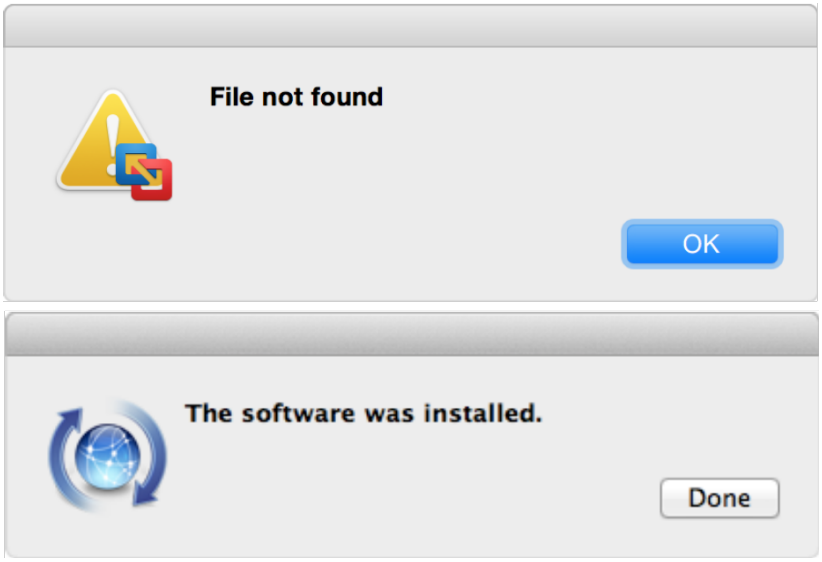


Рис. 1.6. Программа не должна заставлять пользоваться держать в голове информацию, если она сама может её легко хранить и отображать. Если человек отвлёкся, то ему придётся вспоминать о каком файле (верхнее окно) и о какой программе (нижнее окно) ему сообщают [1].

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 может совершать сложную интеллектуальную работу.

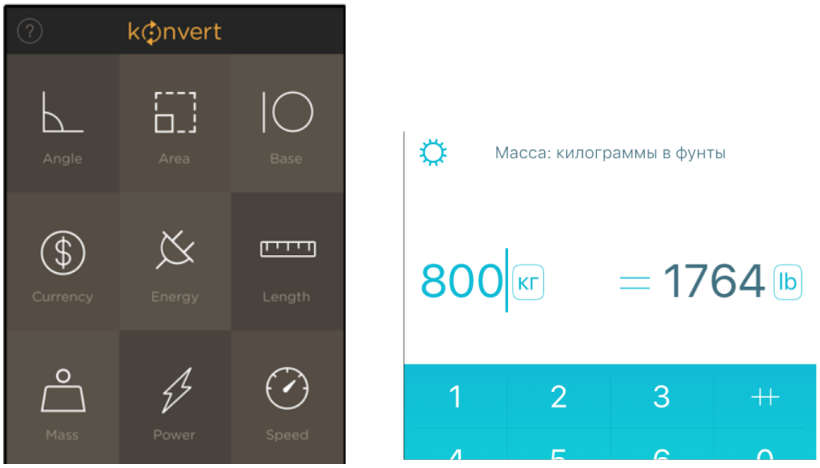


Рис. 1.7. Принцип «сначала данные»: программа не должна заставлять пользователя держать в голове данные, вынуждая сначала задать настройки, выбрать режим (программа справа) или фильтры. На изображении справа пользователь сначала вводит число, а потом задает исходные единицы измерения и выбирает целевые 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 модели[5] 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-

[5]Модель – 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.

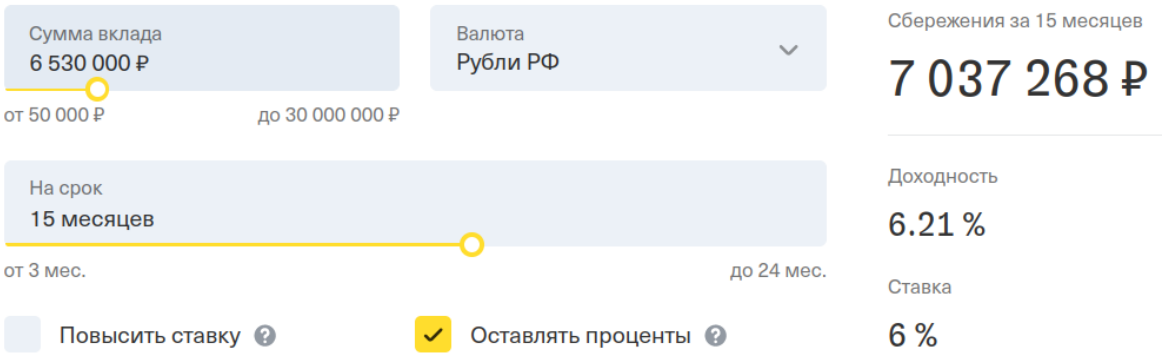


Рис. 1.9. Калькулятор доходности вклада. Вместо сложной таблицы, где перечислены все возможные ставки и сроки вклада на сайте банка есть интерактивный калькулятор. С ним легко экспериментировать меня ползунками срок и суммы вклада, итоговая сумма рассчитывается автоматически при изменении параметров.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

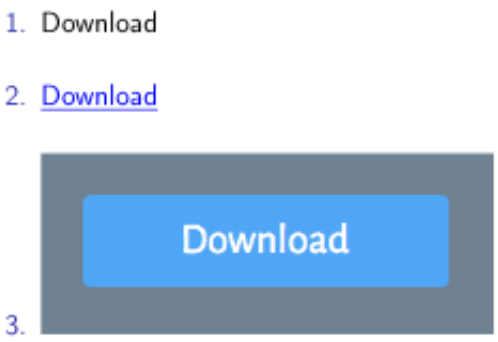


Рис. 1.10. Легко узнаваемые ментальные модели: текст, ссылка и кнопка

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.

Непоследовательно 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 внутренее 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 искажение[6]: 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

[6]см. 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 попутке 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 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 выводу о ПИ следуют из этого?
5. Что такое автоматичные задачи? Как они формируются?
6. Как можно использовать автоматичные 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 системы[1].

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.

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

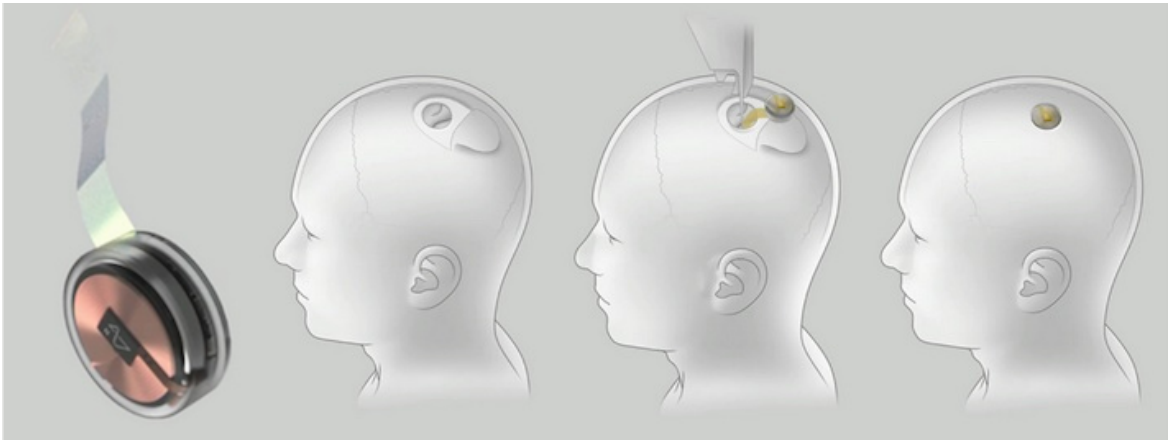


Рис. 2.1. Имплантируемый нейрокомпьютерный интерфейс Neuralink.

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 (осязательный) - 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 однонаправленных интерфейсах внешние устройства могут либо принимать сигналы от мозга (см. рис 2.1), либо посылать 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

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.

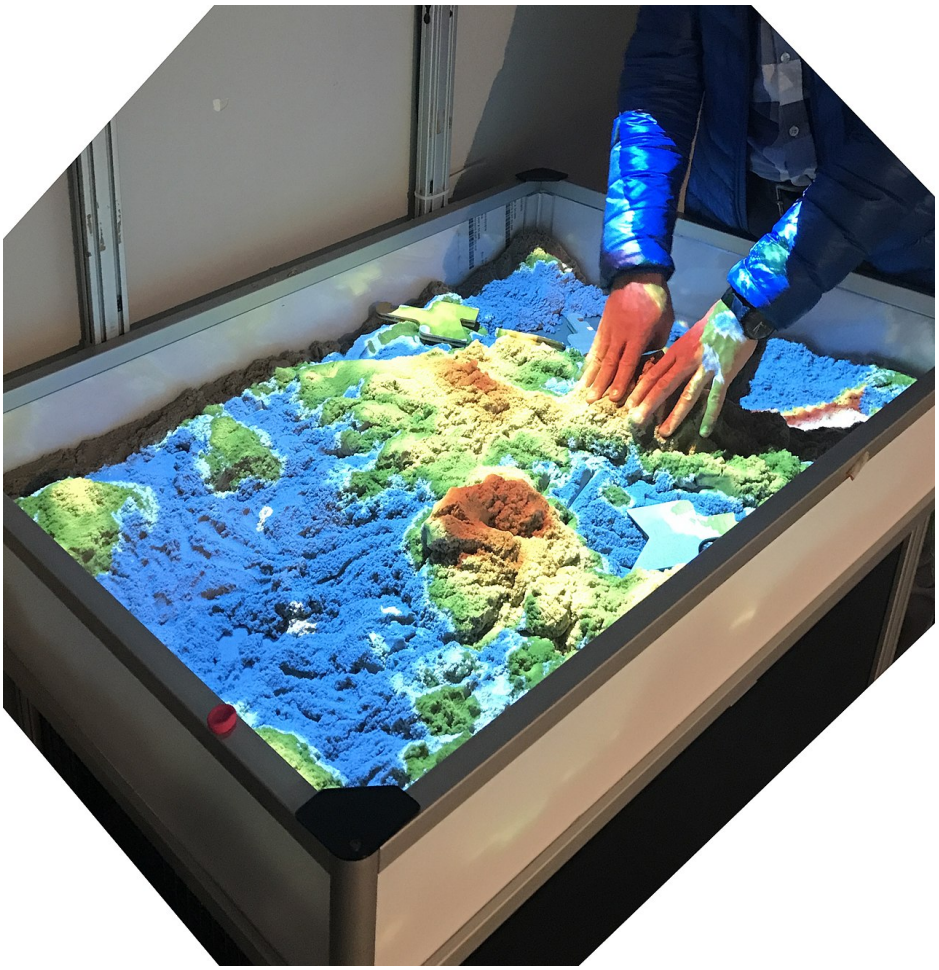


Рис. 2.3. SandScape – виртуальный ландшафт и его объекты, взаимодействуют с пользователем, формирующим этот ландшафт с помощью песка. Демонстрация: vimeo.com/44538789

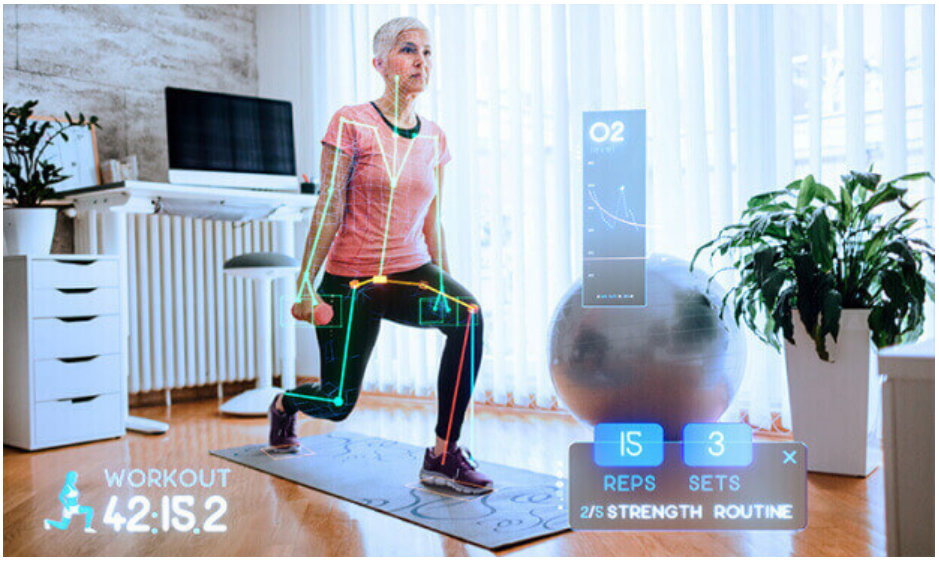


Рис. 2.4. Azure Kinect DK. Система компьютерного зрения распознаёт положение тела. Демонстрация: youtube.com/watch?v=OoJWmmOlws8



Рис. 2.5. Игровой контролер 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 требователен 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 быстрее[2] чем с ГИП, особенно если учесть возможность автоматического

[2]для оценки времени на совершение действий смотрите параграф 6.4

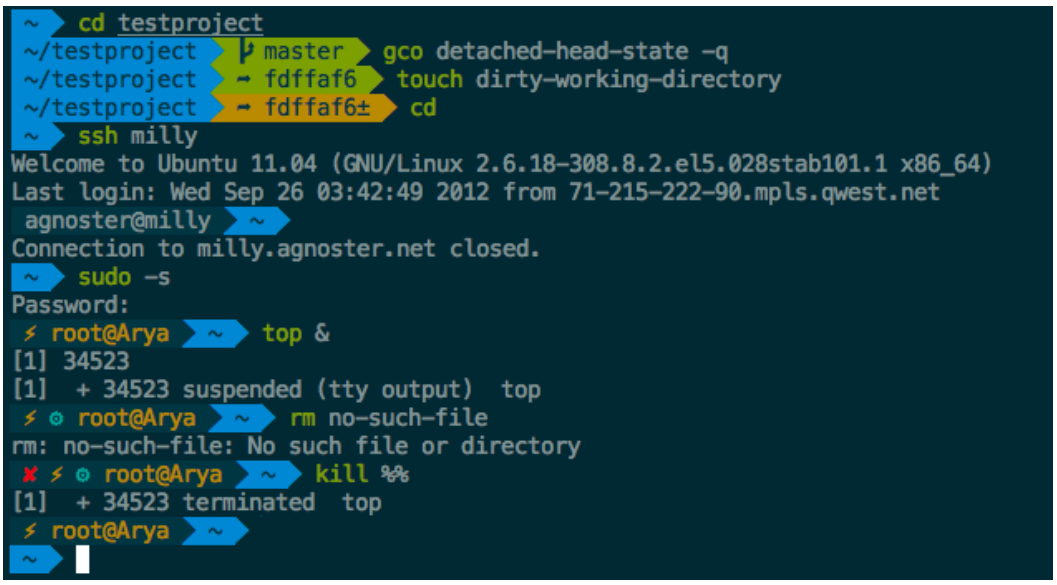


Рис. 2.6. ZSH – командная оболочка для UNIX, с гибким механизмом автодополнения команд, исправления опечаток, цветовым кодированием состояний репозиториев git.

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. Своеобразное 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

[3]см. параграф 5.1 о критериях качества интерфейса

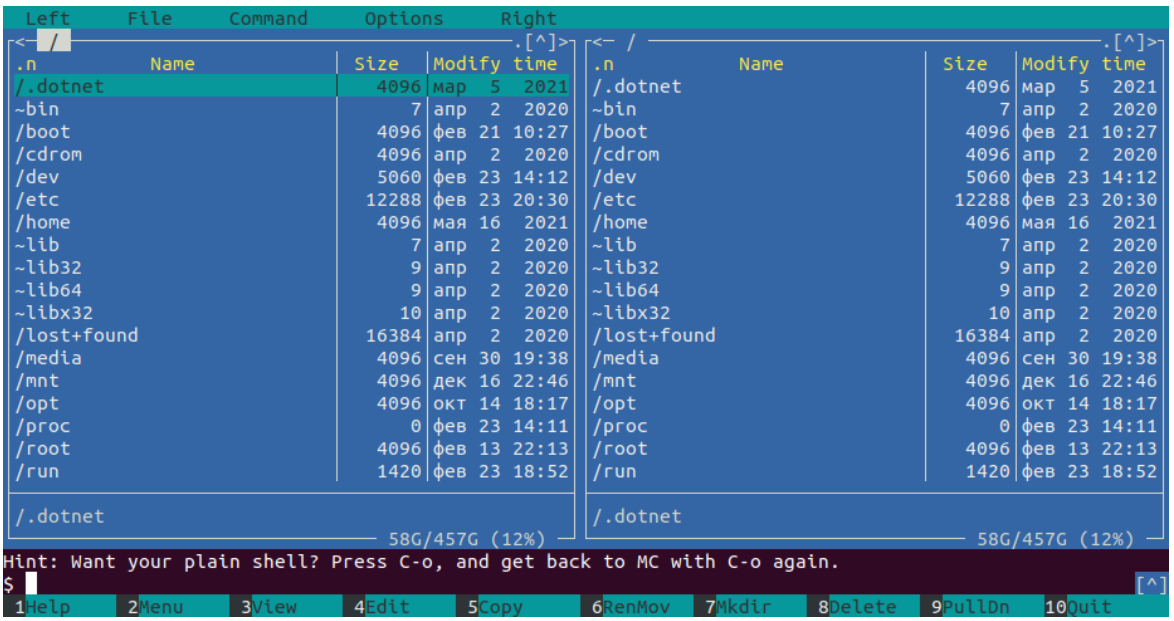


Рис. 2.7. GNU Midnight Commander – файловый менеджер с текстовым ПИ

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 понимая 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, совершённая человеком из-за того, что он не осознавал, в каком состоянии находился интерфейс, и получил не тот результат жеста, которого ожидал, называется модальной ошибкой[4].

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 тяги[5], и продолжал управлять самолётом, не следя за скоростью [1].

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

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

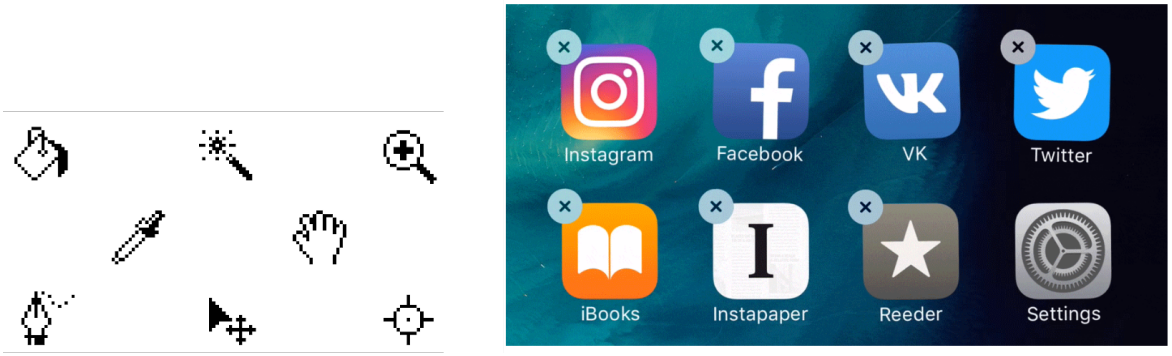


Рис. 2.10. Курсоры и значки на иконках, показывающие режим. В iOS, в этом режиме иконки подрагивают [1].

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 Джеф 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 дизйан 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 работает[1]. Дизайн взаимодействия с пользователем (отдельная отрасль дизайна) – хороший тому пример. Модель представления – тоже объект дизайна.

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

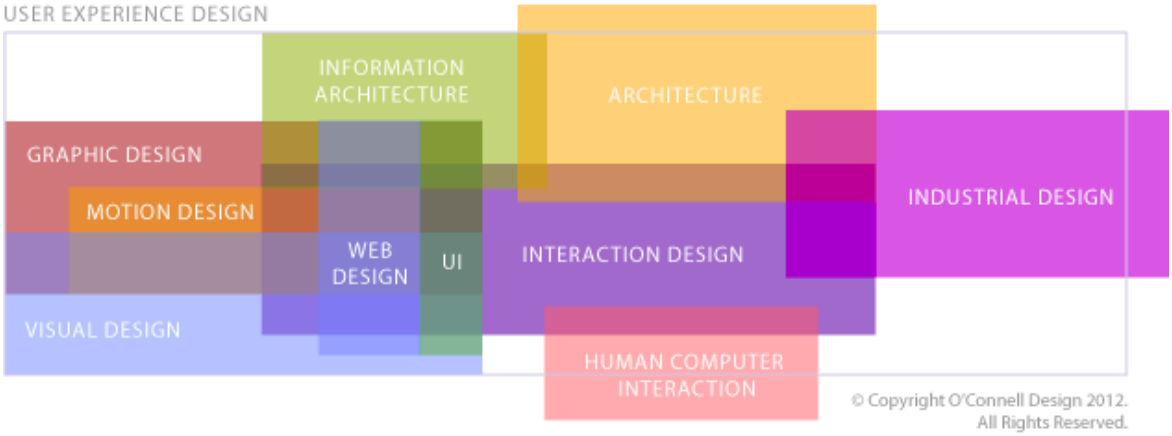


Рис. 3.1. Отрасли дизайна

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-

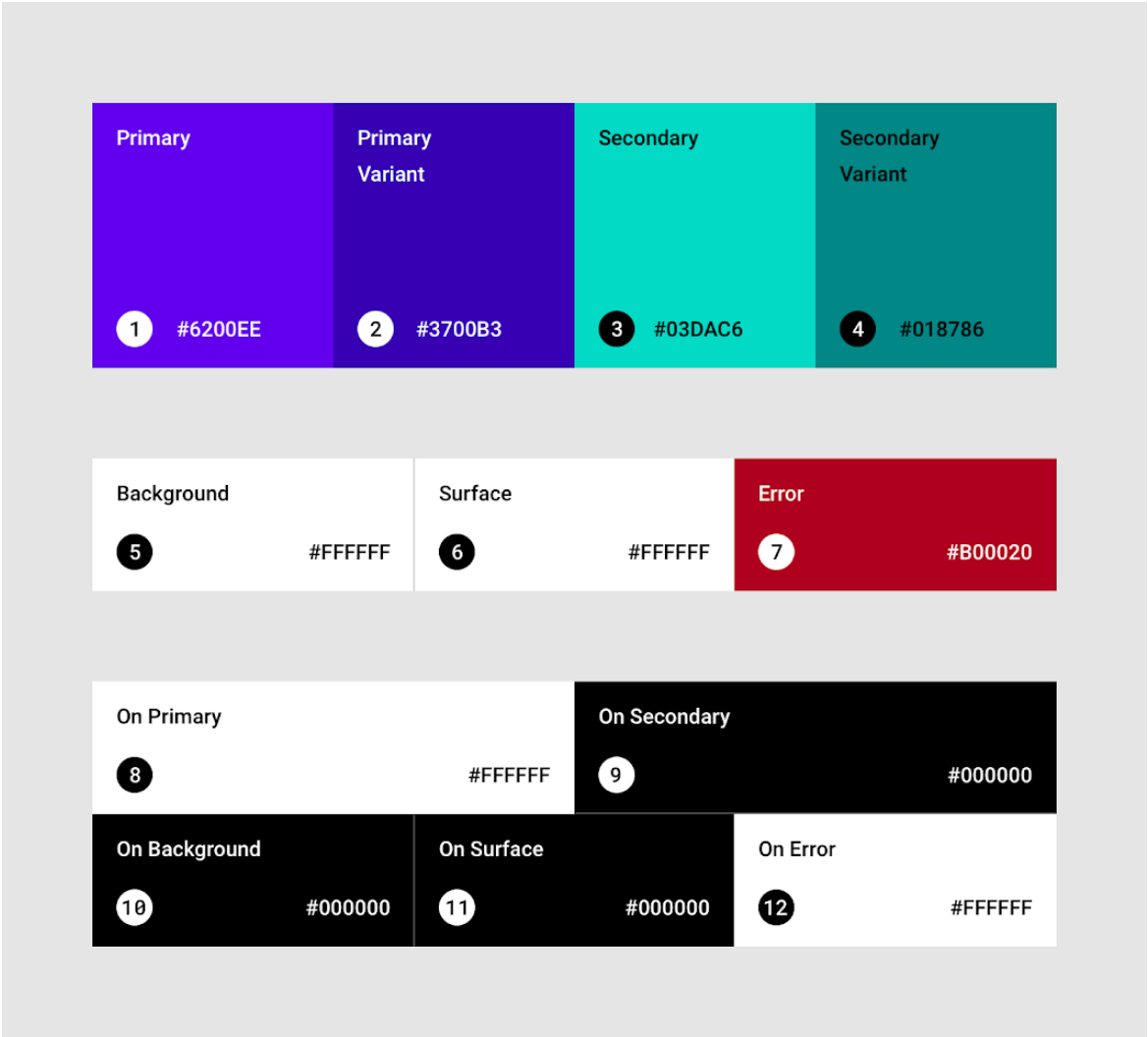


Рис. 3.2. Цветовая схема рекомендованная Material Design

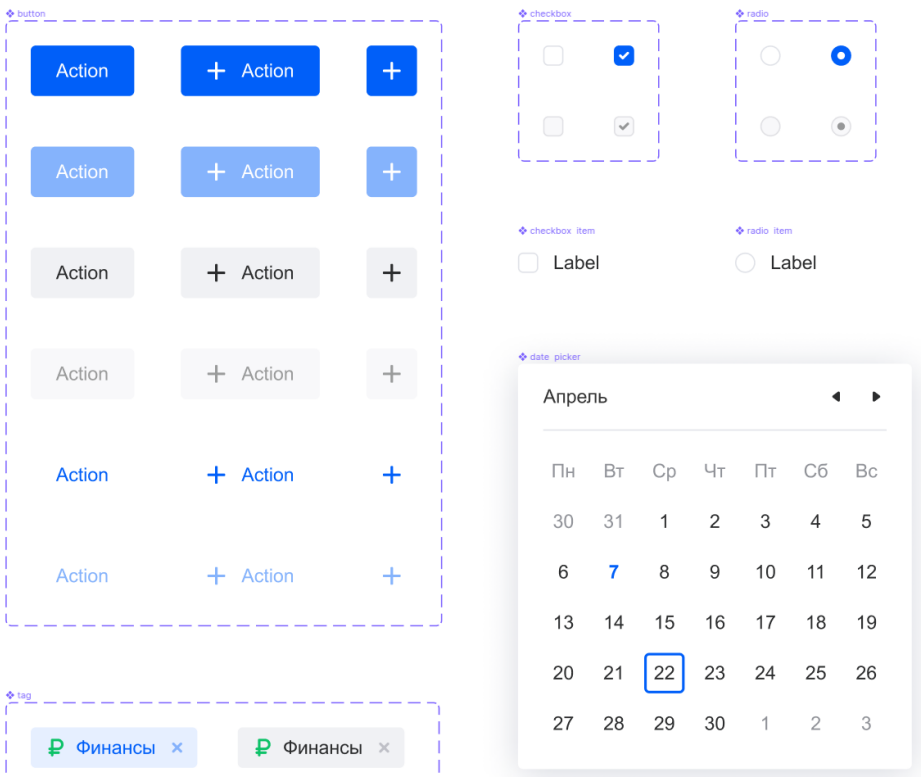


Рис. 3.3. Фрагмент библиотеки готовых компонентов дизайн-системы Paradigm от mail.ru.

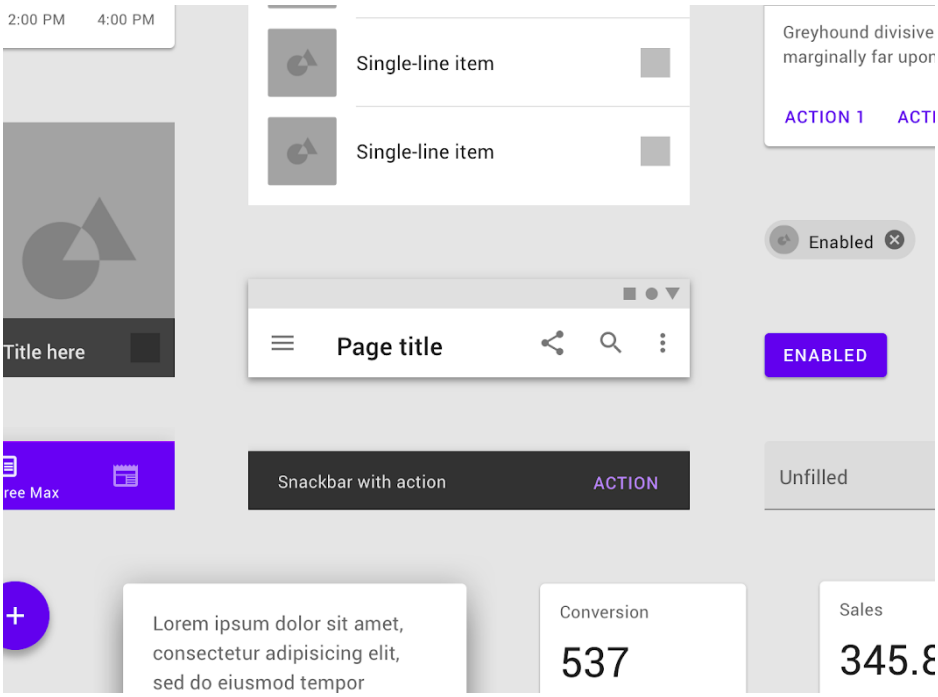


Рис. 3.4. Фрагмент библиотеки готовых компонентов дизайн-системы Material Design

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;
- восприятие[2] – 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) — общепсихологическое направление, связанное с попытками объяснения прежде всего восприятия, мышления и личности. В качестве основного объяснительного 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-

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

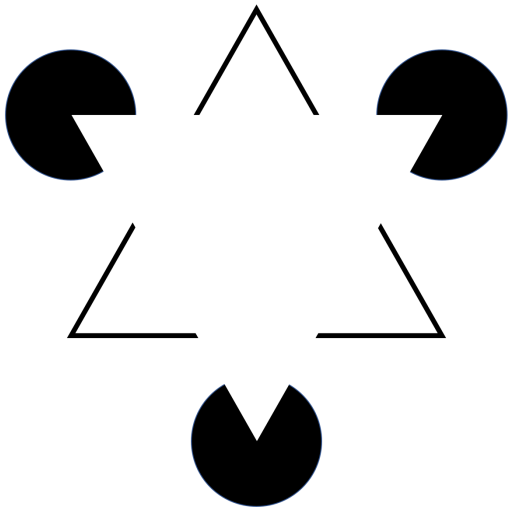


Рис. 3.5. Треугольник Канижа. На изображении нет четко отчерченного белого треугольника, но человеческое восприятие достраивает его.

меты как сумму отдельных частей, но не может составить целостный образ предмета[3].

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 научением и прошлым опытом; предвосхищающие мысли и ожидания также активно руководят нашей интерпретацией ощущений.

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

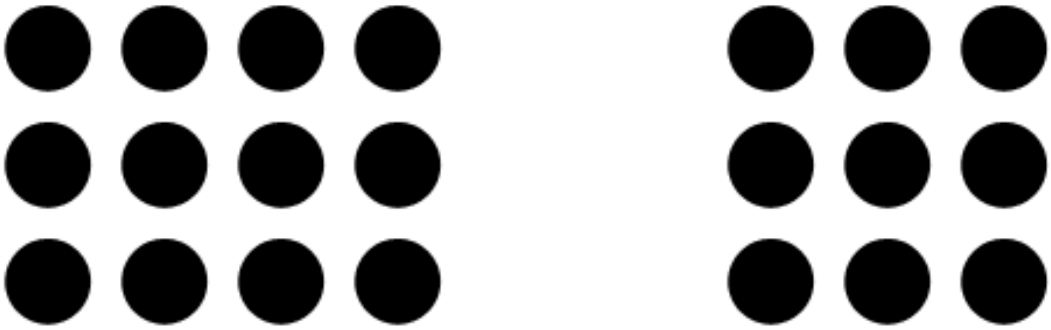


Рис. 3.6. Пример восприятия согласно принципу близости

Принцип близости, называемый ещё теорией близости, возможно один из самых важный 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.

Дилемма 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).

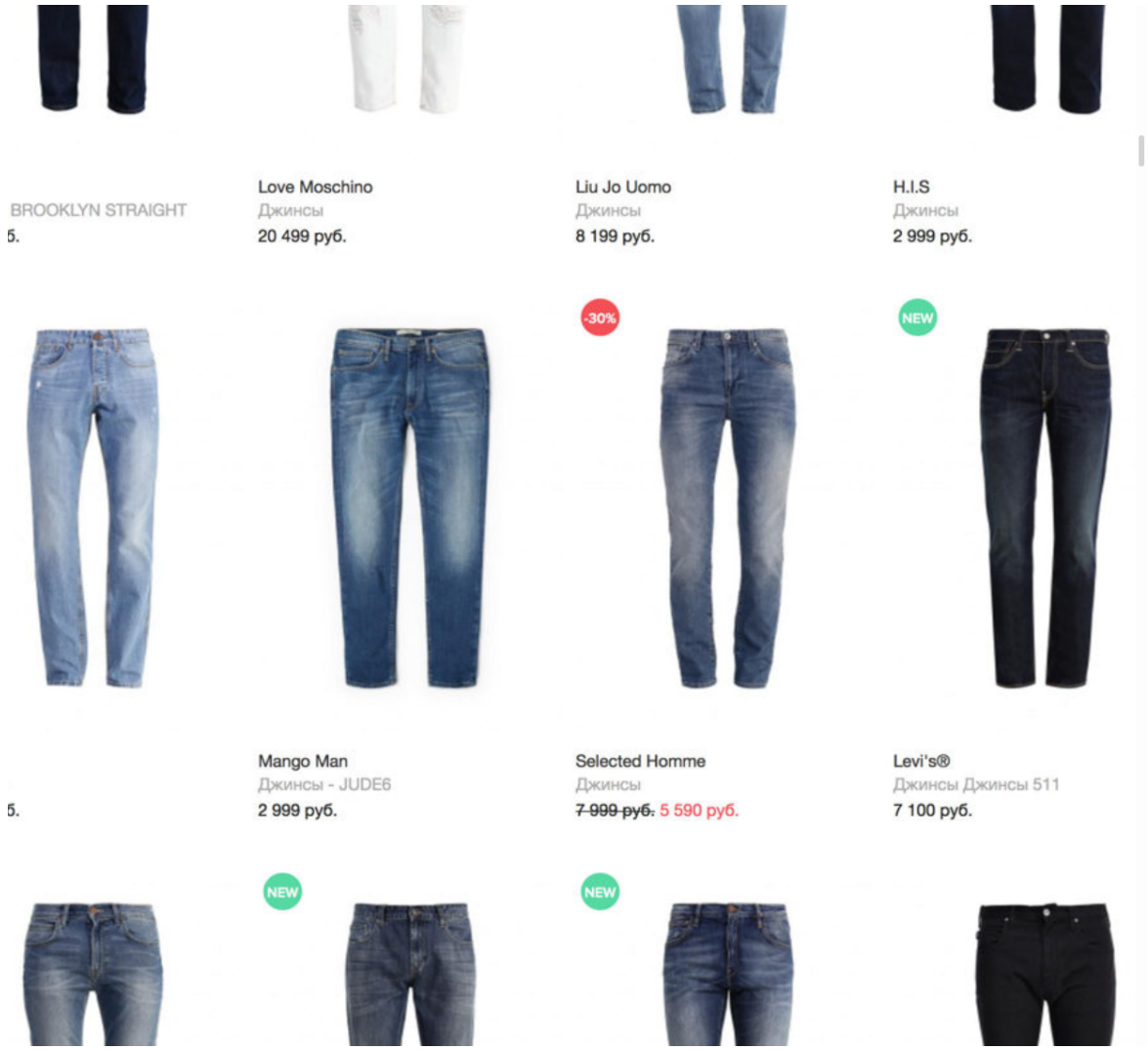


Рис. 3.7. Дизайнер не использовал принцип близости: нельзя понять к каким изображениями относятся подписи. Нужно сделать подписи ближе к связанным изображениям.

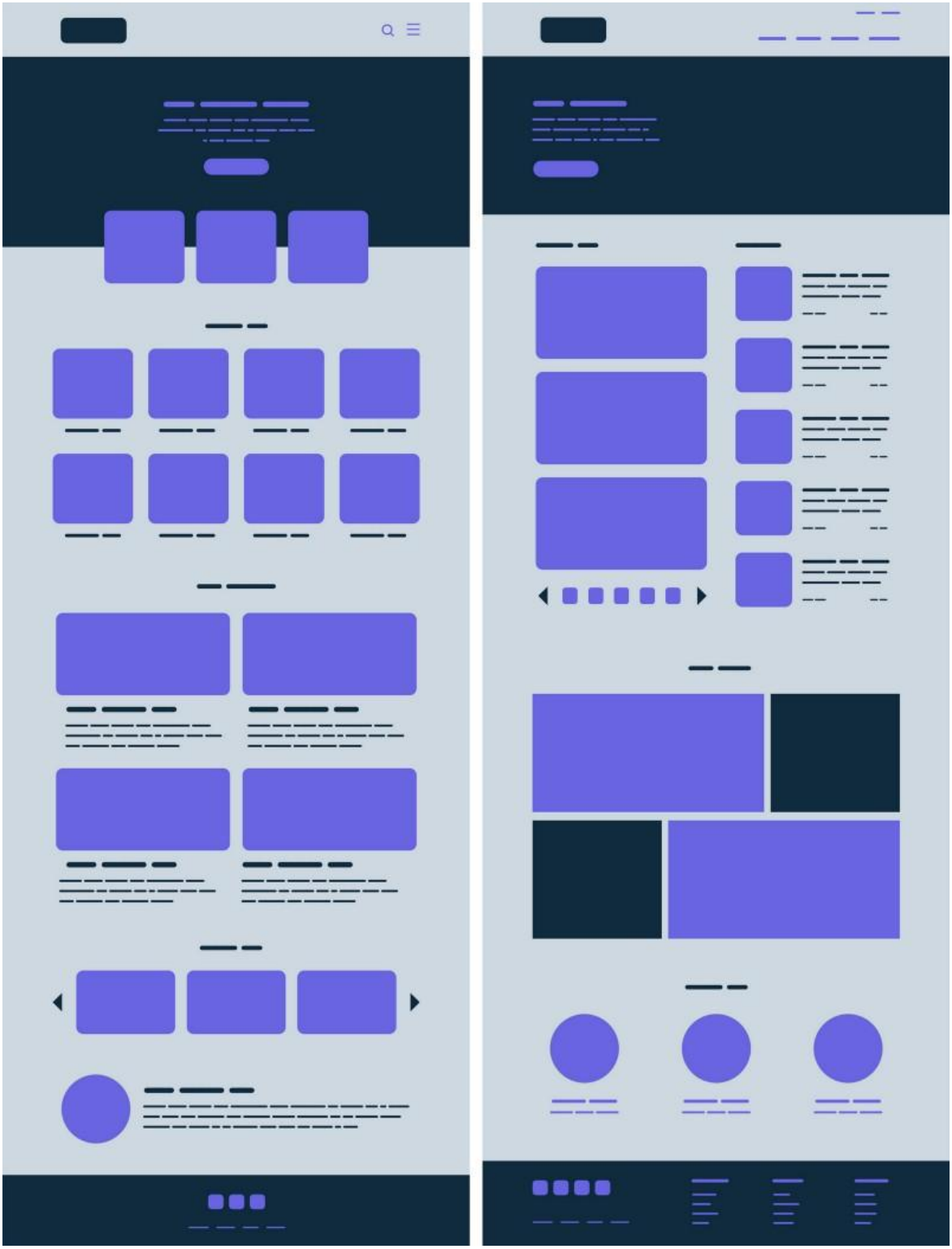


Рис. 3.8. Принцип близости в грубом макете интерфейса. Большое значение имеет пустота,воздух – он отделяет объекты друг от друга. Расстояние между отдельными группами элементов – большое, между элементами в группе – меньше.

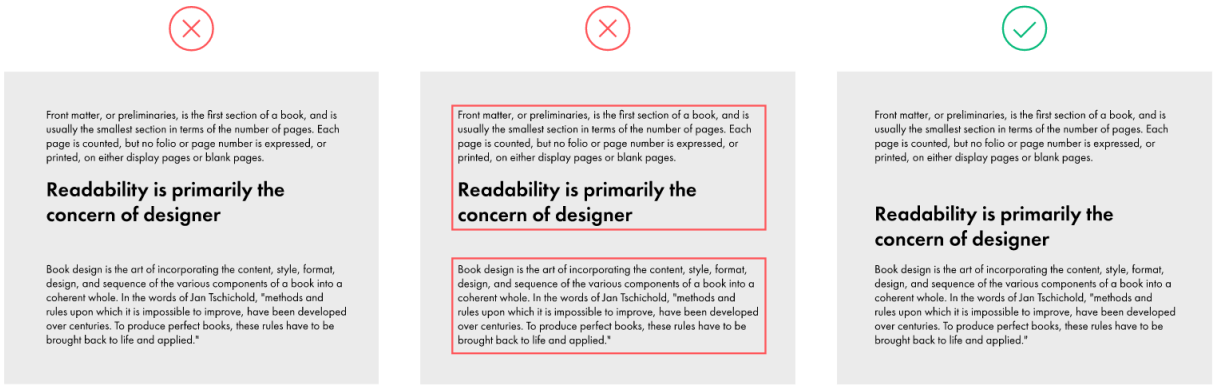


Рис. 3.9. Частая ошибка в визуальной иерархии: заголовок текста “висит“ между абзацами, в то время как должен быть ближе к следующему абзацу, к которому он и относится.

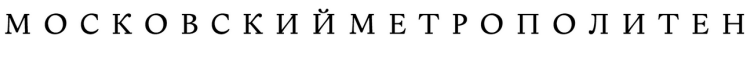


Рис. 3.10. Несоблюдение (сверху) и почти наглядное соблюдение (снизу) следствия принципа близости: внутренние расстояния (между буквами в слове) должны быть меньше внешних (между словами)

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 разграничивающих 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.

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

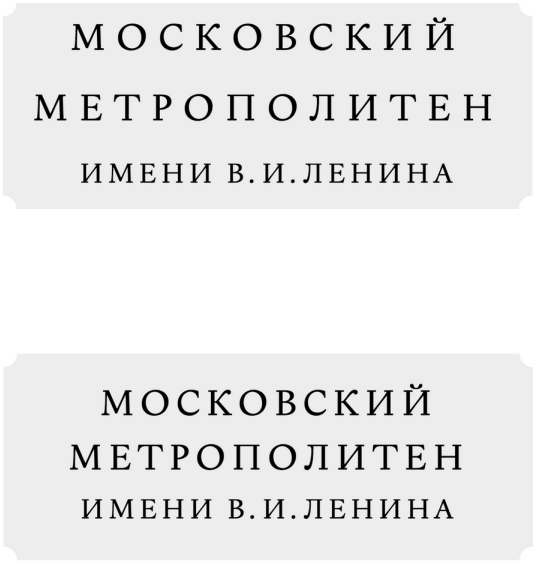


Рис. 3.11. Верху: принцип “внутреннее ≤ внешнее“ нарушен: расстояния внутри слова (между буквами) и особенно между словами надписи больше чем между словами и границами области. Текст на нижнем изображении выглядит более целостным, чем на верхнем

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 шум[5] 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 искажать. На этом построены многие оптические иллюзии. Поэтому в дизайне следует руководствоваться не только правильными геометрическими построениями, но и учитывать то как это воспринимается человеком (рис. 3.16). На рисунке правый треугольник расположен так, что половина его площади лежит слева от вертикальной 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. Слишком 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).

[5] см. также понятие data-ink ratio

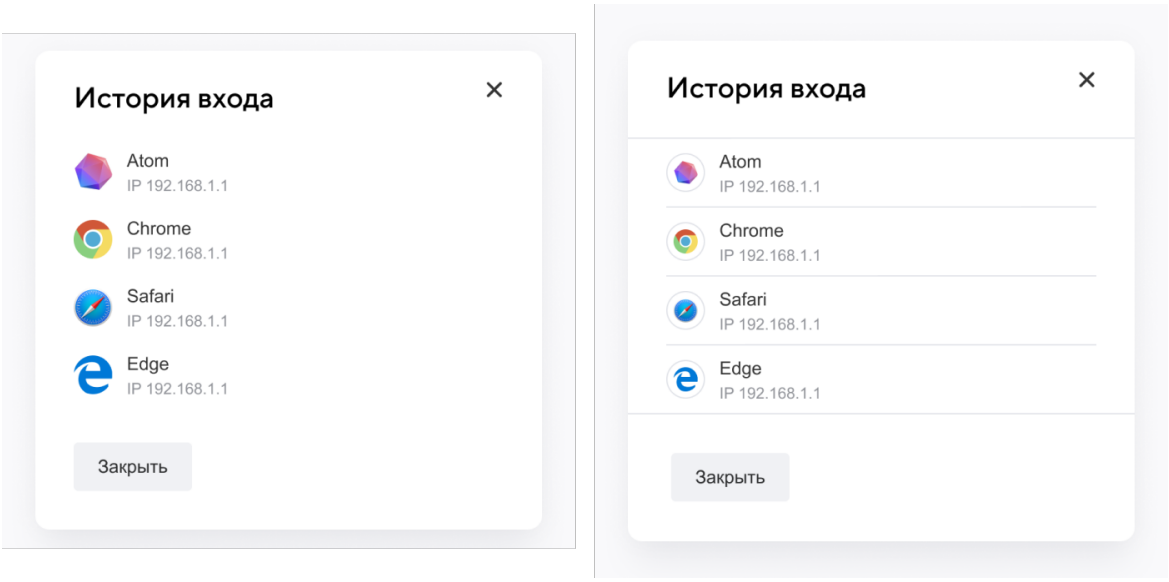


Рис. 3.14. Отделение объектов друг от друга с помощью только пустого пространства лучше, чем с помощью разделителей. Они могут восприниматься как визуальный шум и способствовать обратной задаче – объединения разных групп объектов. Изображение из [52]

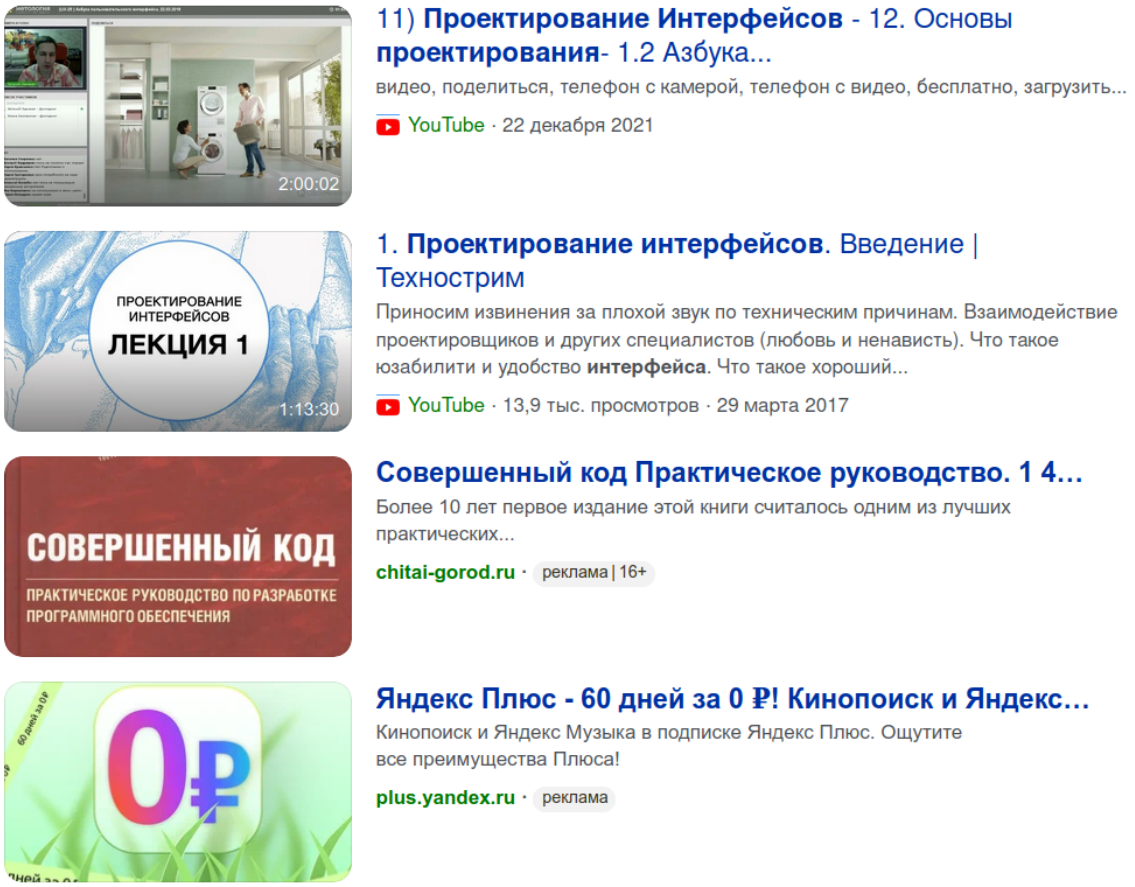


Рис. 3.15. Принцип схожести используется чтобы выдать реклама в поисковой выдачи за результат поиска

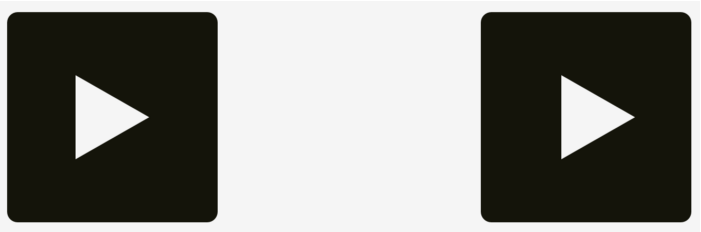


Рис. 3.16. Выравнивание фигур по геометрическому центру (слева) и по визуальному центру (справа). Правый треугольник расположен так, что половина его площади лежит слева от вертикальной оси симметрии квадрата.



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

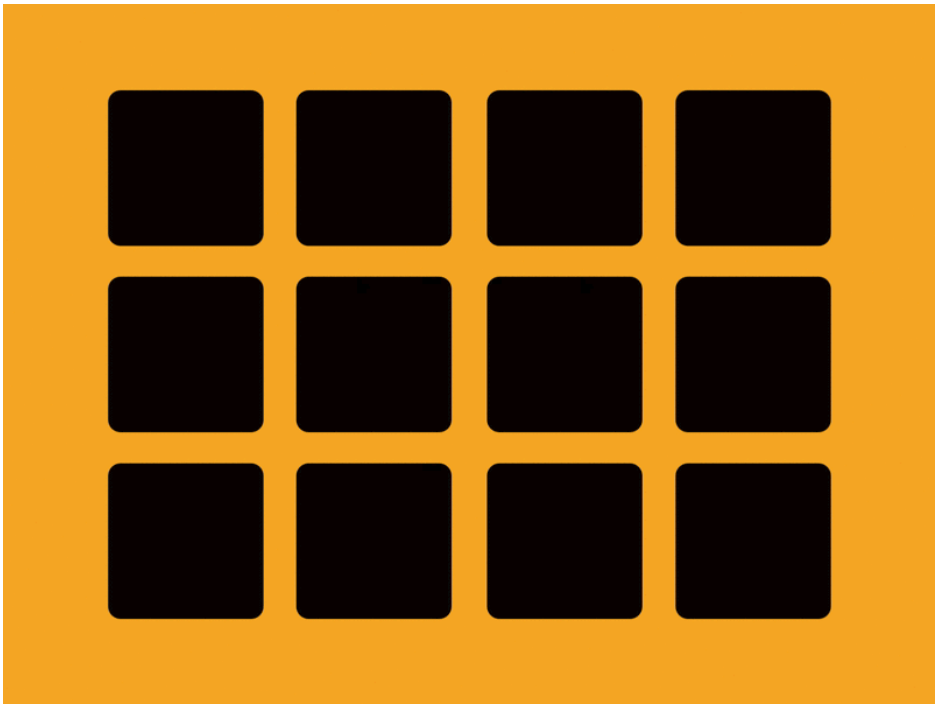


Рис. 3.18. Решётка Германа: места на пересечении линий, разделяющих квадраты, воспринимаются серыми крыгами

3.3 Цветовые модели

Модель RGB (red, green, blue — красный, зелёный, синий) – аддитивная[6] 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 октета[7], 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,

[6] в 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), субстрактивные (например CMYK) для отражённого, например в печати.

[7] октет – 8 двоичных разрядов, байт

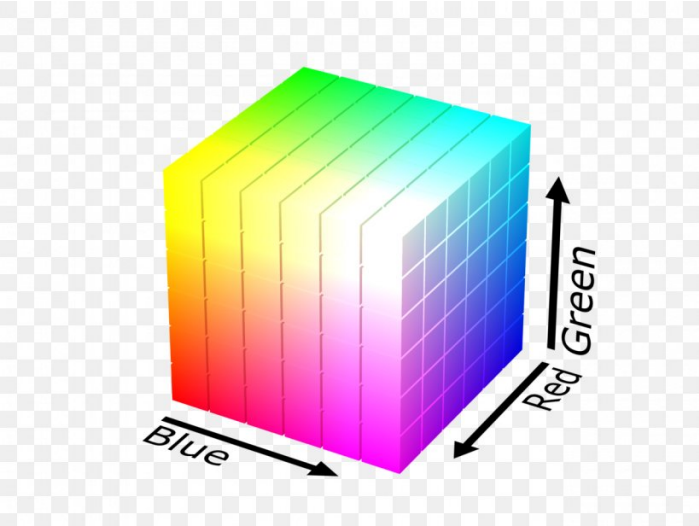


Рис. 3.19. Три цветовые координаты в модели RGB. Скрытая вершина куба окрашена в чёрный.

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

- Hue — цветовой тон, (например, красный, зелёный или сине-голубой). 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 редакторы[8],

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

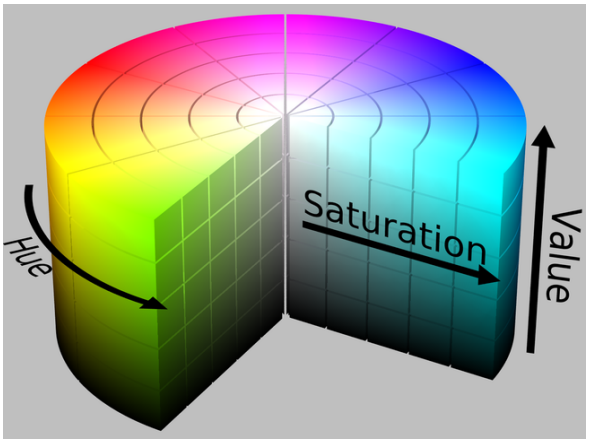


Рис. 3.20. Цветовая модель HSV (HSB). Чтобы изменить только насыщенность (saturation) или только светлоту (value) нужно изменить единственное число в коде цвета.

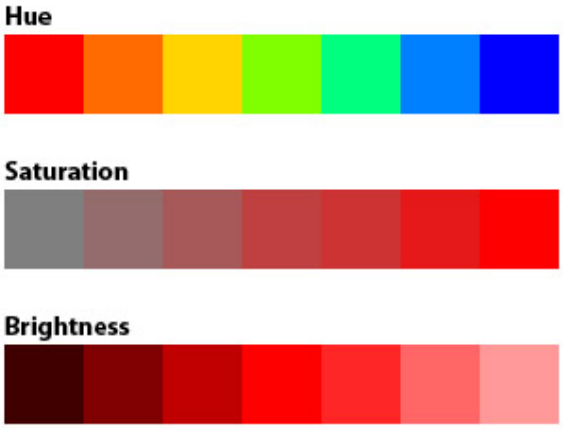


Рис. 3.21. Пример изменения насыщенности (Saturation) и яркости (Brightness) для одного красного тона (Hue)

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.

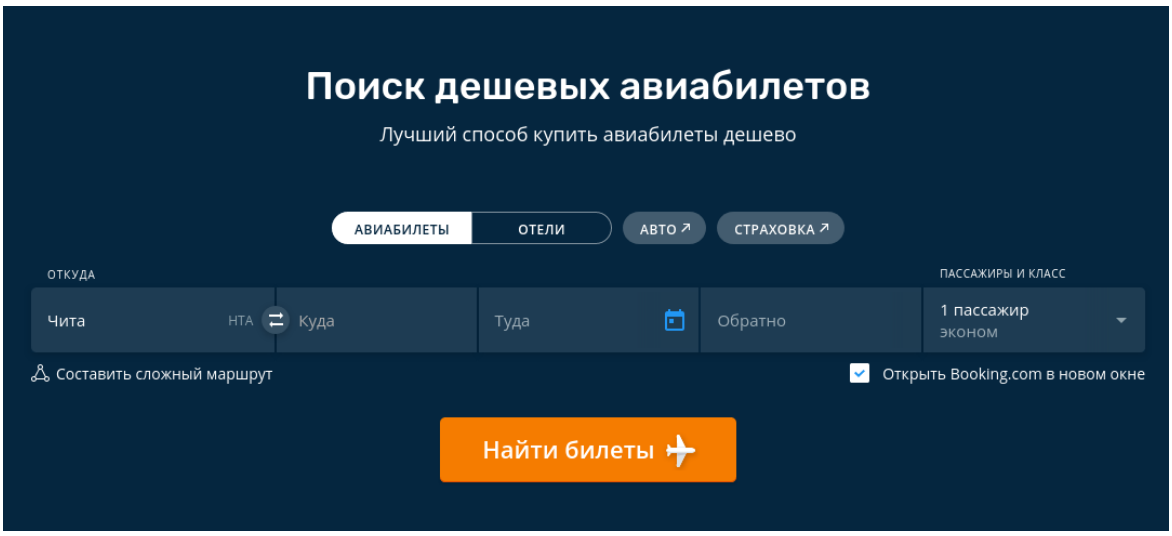


Рис. 3.22. В основе акцентирования внимания цветом – эффект выскакивания. Эффект хорошо работает, когда особых визуальных стимулов (чаще всего фигур) мало.

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

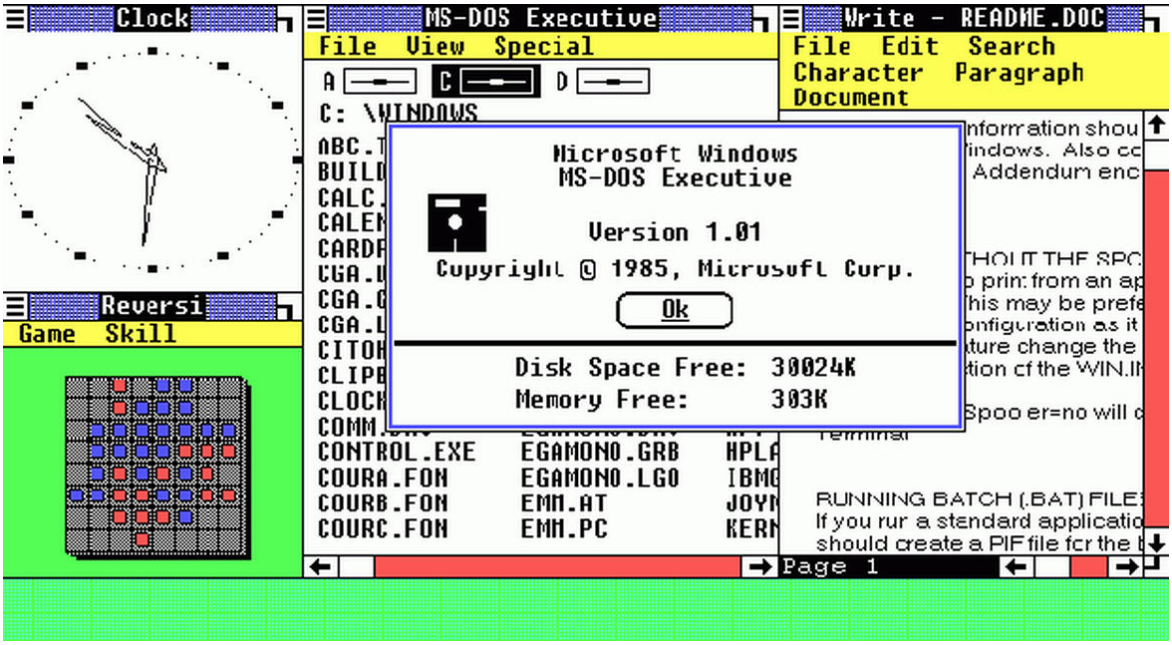


Рис. 3.23. Дизайн интерфейса Windows 1 был создан преимущественно без участия дизайнеров. Абсолютное большинство современных интерфейсов, не смотря на возросшие технические возможности дисплеев, имеют более сдержанную палитру. Задача пользовательского интерфейса – не привлекать к себе внимание, а помочь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

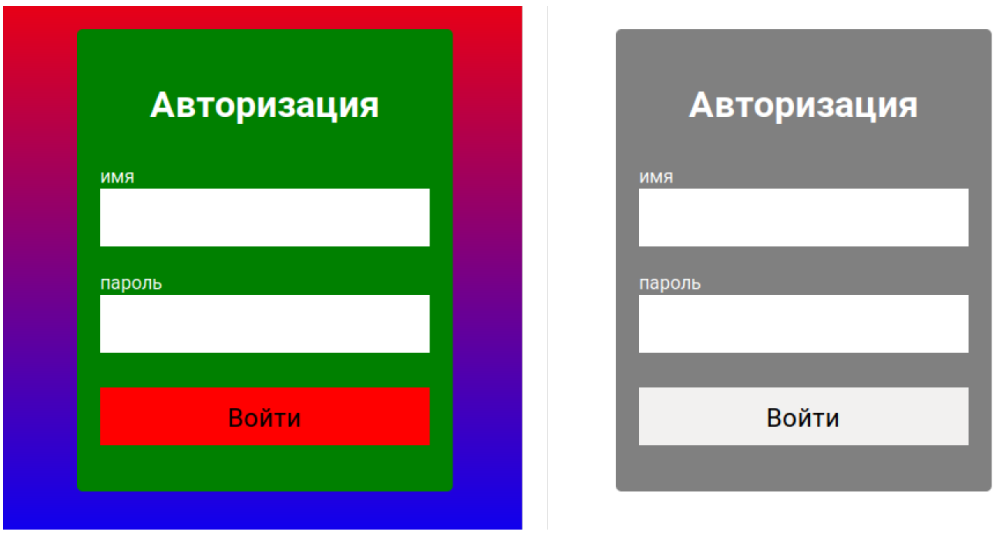


Рис. 3.24. Большое количество визуальных стимулов, даже при сохранении высокого контраста и различимости может проигрывать в эстетическом плане “скучным“ цветовым схемам.

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-страницах[9]. 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?

[9]в 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.

4 Проектирование UX

4.1 Проектирование пользовательского интерфейса

Набор элементов интерфейса и правил взаимодействия описывают систему человек-машина лишь частично.

С одним и тем же интерфейсом взаимодействуют пользователи с разными целями. С одним и тем же интерфейсом взаимодействуют пользователи с разным опытом. Пользователи имеют разные ожидания и представления относительно интерфейса. Пользователи по-разному оценивают продукт: кому-то он нравится, кому-то нет. Введём понятие, которое как можно более полно характеризует взаимодействие пользователя и продукта.

Опыт пользователя, опыт взаимодействия (User eXperience, UX) – это восприятие[1] и ответные действия пользователя, возникающие в результате использования и/или предстоящего использования продукции, системы или услуги.

Более лаконичное определение дал Д. Норман. ***User Experience*** – все аспекты взаимодействия конечного пользователя с компанией, её услугами и продукцией [47].

UX более субъективное понятие, чем *интерфейс пользователя* потому что включает в себя ещё и реакцию пользователя.

Программа может иметь идеальный UI, который позволяет решать задачи пользователя быстро и эффективно. Но если она медленно работает – это пример плохого UX, при хорошем UI.

[1] целостный образ предмета

Книжный магазин с отличным поиском, аннотациями и удобными способами оплаты. Но небольшой выбор книг – снова плохой UX, пользователь, скорее всего, не получит того, чего желает.

Программа может иметь хороший, удобный и понятный UI, но долго запускаться, неожиданно закрываться, иметь не достаточную (для своей области применения) функциональность, а значит UX может быть не таким же хорошим как UI.

С другой стороны, программа может соответствовать ожиданиям пользователя, вызывать хорошие эмоции при не слишком выверенном UI[2]. Таким образом, можно создать положительною субъективную оценку от продукта, который имеет недостатки.

Нередко хороший или плохой UX становится следствием когнитивных искажений. Например, эффект IKEA – это когнитивное искажение, которое появляется, когда покупатели непропорционально высоко оценивают значимость (ценность) товаров, которые они создают отчасти сами (например, собирают из деталей). Выполнение длительной операции (запуска программы, скачивание обновления, сохранение настроек) воспринимается пользователем приятнее, если он видит индикацию прогресса. Например, полосу прогресса, изменяющиеся сообщения и т.п.

Некоторые операции, которые выполняются (по мнению пользователей) слишком быстро снабжают более длительной, по сравнению с фактическим временем выполнения, анимацией выполнения. Из-за мгновенного выполнения важной операции, у пользователя может сложится впечатление, что она не выполнена или выполнена с ошибкой.

Поэтому важно проектировать не только UI, но и учитывать другие аспекты взаимодействия человека и программы.

[2]как и бренд ≠ качество товара

4.2 Проектирование UX

4.2.1 Уровни UX

Таким образом, задача создания продукта может рассматривается как задача создания положительного UX. Опыт пользователя плохо формализуется в силу того, что может отличатся от пользователя к пользователю и описывает в том числе субъективную сторону взаимодействия пользователя и продукта. Тем не менее можно выделить общие характеристики если не для всех пользователей, то для большей части целевой аудитории.

Пять уровней UX – это концептуальная модель, предложенная Джессом Гарреттом (Jesse James Garrett) для ***проектирования опыта пользователя*** веб-приложений [4]. Однако её можно расширить и на проектирование продукта вообще.

Процесс разработки и планирования UX разбивается на несколько этапов (рис. 4.1), каждый из которых рассматривает UX на разном уровне абстракции:

- Уровень поверхности (surface);
- Уровень компоновки (skeleton);
- Уровень структуры (structure);
- Уровень возможностей (scope);
- Уровень стратегии (strategy).

Самый первый и абстрактный уровень – понимание пользователей и их нужд. Продукт с хорошим интерфейсом, который не решает никаких задач пользователй – бесполезен. Нужно понять и цели заказчика, если он и пользователи не одно и то же лицо. Цель пользователя интернет магазина – купить нужный товар. Цель владельца – продавать товары. Цели пользователей и заказчика могут быть одинаковыми или немного различаться. Владельцы интернет-магазина хотят привлечь больше покупателей и продать больше товара, но не все покупатели согласны потратить лишние деньги на покупки или получать рекламную рассылку.

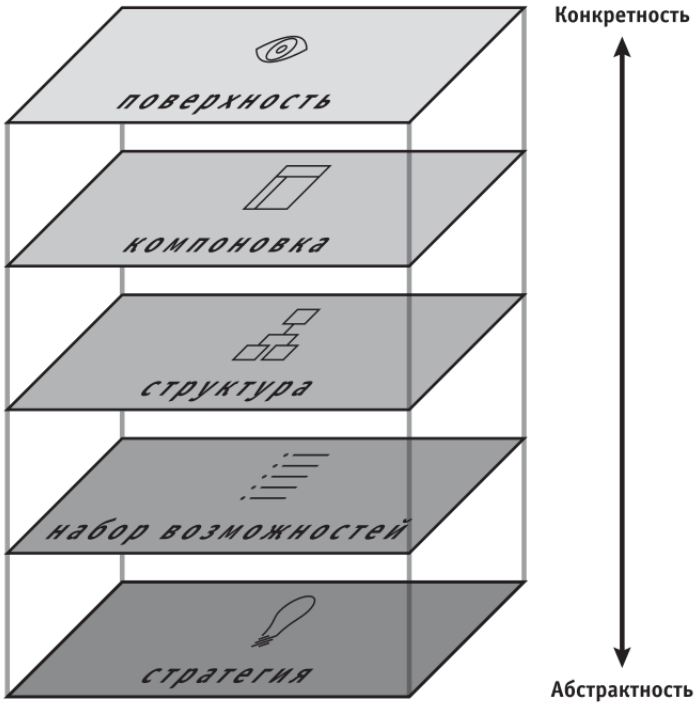


Рис. 4.1. Пять уровней проектирования UX

Когда понятно, чего хотят пользователи, нужно продумать функции продукта и информацию, которую он предоставит пользователю. Всё это должно позволить пользователям достичь своих целей – это уровень возможностей.

Когда готовы требования к функциям и информационному наполнению нужно понять, на какие окна экраны или страницы разделить программу или сайт. То есть описать информационную архитектуру. Это особенно актуально для сайтов, где обычно много страниц.

Определившись с содержимым каждой страницы или окна, определяют компоновку блоков из элементов интерфейса. Создают грубые макеты в которых отмечено положение основных групп элементов ПИ. Макетов обычно делают несколько вариантов. Их основная задача – понять, какой вариант компоновки больше подходит для решения задач пользователями.

Наконец выбирают наиболее удачные варианты макетов и дополняют их деталями, приводя уже конечный вид интерфейса пользова-

теля, добавляя имитацию всех или отдельных действий – получая прототип.

Полученные макеты или прототипы, совместно с необходимыми материалами – изображениями, набором иконок и описанием стилей или дизайн-системой передают непосредственно разработчикам (например, верстальщикам), которые будет реализовывать готовый продукт.

Разбивка проектирования UX, а с ним и продукта на эти этапы позволяет ограничиться принятием в самом начале более стратегических решений, не обращая внимания на детали реализации. Принимая решения на каждом уровне проектирования, разработчик сужает диапазон решений на каждом следующем уровне (рис. 4.2). Это упрощает проектирование. Решения, принятые на разных уровнях одновременно, могут быть плохо совместимы.

Такой подход не минимизирует риск появления ошибок планирования. Например, если страницы сайта проектируются или даже верстаются на этапе уточнения требований заказчика или исследования потенциальных пользователей (уровень стратегии), вполне вероятно, что часть созданных страниц окажутся лишними, а часть придётся переделать. Для типовых продуктов, например сайтов маленьких интернет-магазинов, риск переделок относительно невелик. Но только потому, что все типовые решения прошли все стадии проектирования в том или ином виде. Кроме того, есть риск, что заказчику и конечным пользователям типовое решение в чистом виде не подойдёт.

Опишем каждый из этапов разработки более подробно.

4.2.2 Уровень стратегии

Первый этап проектирования продукта – определиться с целями и задачами пользователей, в том числе потенциальными и понять цели заказчика (если он есть).

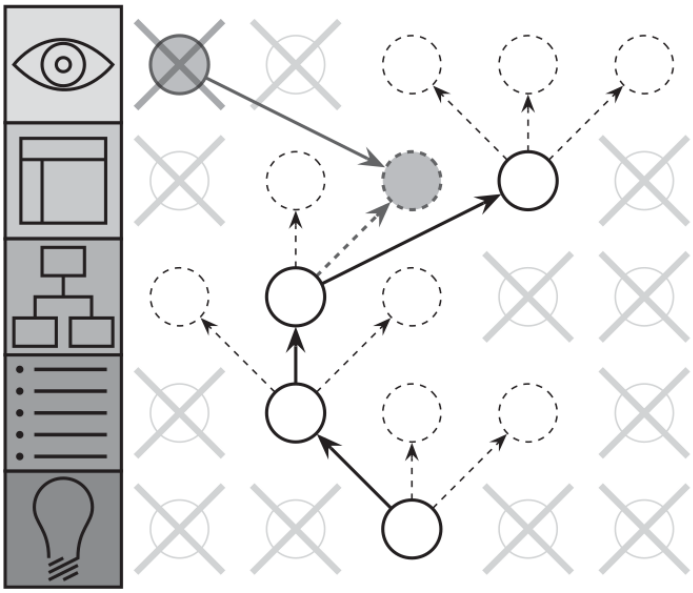


Рис. 4.2. Принимая решения на каждом уровне проектирования, разработчик сужает диапазон решений на каждом следующем уровне. Это упрощает проектирование. Решения, принятые на разных уровнях одновременно, могут быть не согласованы.

Здесь возможны несколько вариантов:

- заказчик и пользователи – одно лицо. Пример: разработка продукта для использования внутри компании заказчика. Например, ПО для ERP[3] системы.
- Заказчик отдельно, пользователи отдельно. Пример – разработка интернет магазина.
- Заказчиком выступает сам разработчик.

Изучение заказчика. Основной источник по стилю – брендбук[4]. Возможно, понадобится написать для заказчика техническое задание[5].

Заказчик не всегда стремится удовлетворить потребности пользователя. Иногда он стремится эти потребности создать. Например,

[3]Enterprise Resource Planning, планирование ресурсов предприятия. 1С: Предприятие – пример ERP ПО

[4]Брендбук – официальный документ компании, в котором описывается концепция бренда, и другие данные, включая полное руководство по фирменному стилю

[5]Техническое задание (ТЗ, техзадание) — документ или несколько документов, определяющих цель, структуру, свойства и методы какого-либо проекта, и исключающие двусмысленное толкование различными исполнителями

предлагает дополнительные товары в интернет-магазине, делает таргетированную рекламу.

Одна из проблем дизайна продуктов – дизайн продуктов должен быть предназначен для пользователей, но чтобы быть реализованным он должен понравиться заказчику. Который, как правило, некомпетентен в области проектирования интерфейсов.

Изучение целевой аудитории. В первую очередь нужно узнать о целях пользователей и задачах, которые они решают. Как правило, это можно узнать у заказчика, если он хочет использовать продукт сам или хорошо представляет целевую аудиторию. Если продукт узкоспециализированный то имеет смысл провести опрос или проинтервьюировать пользователей.

Пользователи могут предлагать способы решения своих задач. Но стоит помнить, что пользователи – не эксперты в области проектирования интерфейсов, с другой стороны, разработчик, скорее всего, тоже не эксперт в предметной области пользователей.

На этом этапе разумно изучить аналогичные продукты. В конце этого этапа нужно ответить на главные вопросы – для чего и для кого разрабатывается продукт.

4.2.3 Метод персонажей

Часто целевая аудитория продукта неоднородна, поэтому усреднённый пользователь будет слишком нереалистичной моделью. Поэтому целевую аудиторию разделяют на несколько групп по принципу схожести целей и опыта пользователей. Для каждой группы описывается типичный представитель – ***персонаж*** (персона). В зависимости от целевой аудитории составляют 2-6 персонажей.

Метод персонажей был предложен Аланом Купером. Он помогает структурировать целевую аудиторию продукта, задокументировать потребности и особенности значимых групп пользователей. Предпо-

лагается, что продукт проектируется для персонажей. Этот подход также используется в маркетинге.

Персонажи описываются на основе исследования потенциальных пользователей. Если данных о целевой аудитории недостаточно, то условный персонаж, описание которого частично состоит из гипотез, тоже чаще всего оказывается полезен.

Описание персонажа обычно включает в себя:

- Название группы целевой аудитории;
- Имя;
- Фотографию (аватар);
- Демографические данные: пол, возраст, образование, род деятельности, семейный статус, дети;
- Пользовательский опыт (опытный пользователь или новичок), используемые устройства;
- Цели – зачем пользователи будут использовать продукт?
- Мотивация – почему пользователь будет использовать именно ваш продукт?
- Барьеры – что мешает пользователю достигать своих целей?

Конкретный набор характеристик персонажа может менять в зависимости от продукта. Проектируя интернет-магазин подарков семейные статус и наличие детей будут иметь значение, а для сайта тематического интернет-издания скорее всего нет.

Помимо перечисленных характеристик персонажа иногда имеет смысл добавить описание контекста, в котором пользователь будет использовать продукт. ГИС часто используют на ходу, со смартфона, значит важно понимать, что интерфейс должен быть адаптирован для использования одной рукой, все детали должны быть различимы при ярком солнечном свете. Стоит указать какие аккаунты

имеет персонаж, в каких социальных сетях персонаж зарегистрирован. Для тематического новостного сайта тогда можно будет предусмотреть репост в социальную сеть и возможность регистрации на сайт через Гугл-аккаунт. Какими ещё продуктами попользуется персонаж? Если разрабатывается CRM-система, то возможно стоит создать интерфейс, знакомый пользователям популярных аналогов, например 1С. К целям персонажа как пользователя иногда имеет смысл добавить цели персонажа как личности.

Карточка персонажа может быть оформлена в текстовом редакторе, представлена в виде ментальной карты или создана в специальном сервисе, например uxpressia.com, hubspot.com/make-my-persona, xtensio.com.

Таким образом, персонажи помогают структурировать информацию о целевой аудитории и понять её потребности. Дают возможность оценивать проектные решения с точки зрения архетипичных пользователей и удерживают от соблазна проектировать для себя, но не для целевой аудитории. Наконец, способствуют эмпатии UX-дизайнера по отношению к пользователю.

Пример персонажа – пользователя сайта вуза.



Группа ЦА: Студент-заочник.
Дмитрий, 25 лет.
Работает системным администратором
Устройства: смартфон, Андроид 10;
ноутбук, Windows 10.

Цели:

- Хочет получить образование и диплом бакалавра в IT (программирование);
- Хочет знать даты сессии, расписание занятий и экзаменов;
- Хочет получать задания дистанционно;
- Хочет знать, правильно ли он понял задание;
- Хочет знать как оформить контрольные и курсовые работы;
- Хочет знать, правильно ли он начал выполнять задания.

Барьеры, демотивирующие факторы:

- Не любит узнавать неожиданные новости в последний момент, перед самой сессией (смена дат экзаменов, изменение заданий);
- Не запоминает, где находится нужная информация на сайте университета;
- Не хочет делать задания по “ненужным“ предметам;
- Иногда откладывает дела на последний момент.

Пользовательский опыт. Опытный пользователь, активно пользуется социальными сетями.

4.2.4 Диаграмма прецедентов

Для описания функциональных возможностей может использоваться ***use case*** диаграмма (***диаграмма прецедентов***, диаграмма вариантов использования) – UML-диаграмма, отражающая отношения между актёрами и прецедентами [2].

Она строится из трёх основных элементов:

- ***Система*** (рамки системы). Прямоугольник обозначающий систему. Включает в себе прецеденты. Иногда не приводится.
- ***Прецедент*** – возможность системы (часть её функциональности), благодаря которой пользователь может получить конкретный, измеримый и нужный ему результат.
  Обозначается овалом с описанием возможности из нескольких слов. Например: просмотр страниц сайта, комментирование, добавление постов.
- ***Роль*** (actor), которую пользователь играет по отношению к системе. Обозначается человечком с названием роли. Например: посетитель сайта, администратор сайта

Роль – модель пользователя , которая концентрируется на наборе возможностей, а не на характеристиках самих пользователей (персонажей). Потребности каждого персонажа должны быть реализованы через прецеденты. При этом одной роли могут соответствовать несколько персонажей, так же как и персонаж может менять роли.

Прецедент соответствует отдельному сервису системы, определяет один из вариантов её использования и описывает типичный способ взаимодействия пользователя с системой.

Для примера приведём фрагмент диаграммы вариантов использования (рис. 4.4) банковского приложения, которая иллюстрирует основные элементы и отношения между ними, но не претендует на полноту. У пользователя могут быть две роли: клиент, VIP клиент.

[5]Extend – расширение прецедента, include – включение. Отношение между “Клиентом“ и “VIP клиентом“ – обобщение. Для удобства восприятия не связанные по смыслу прецеденты прецеденты окрашены в разные цвета.

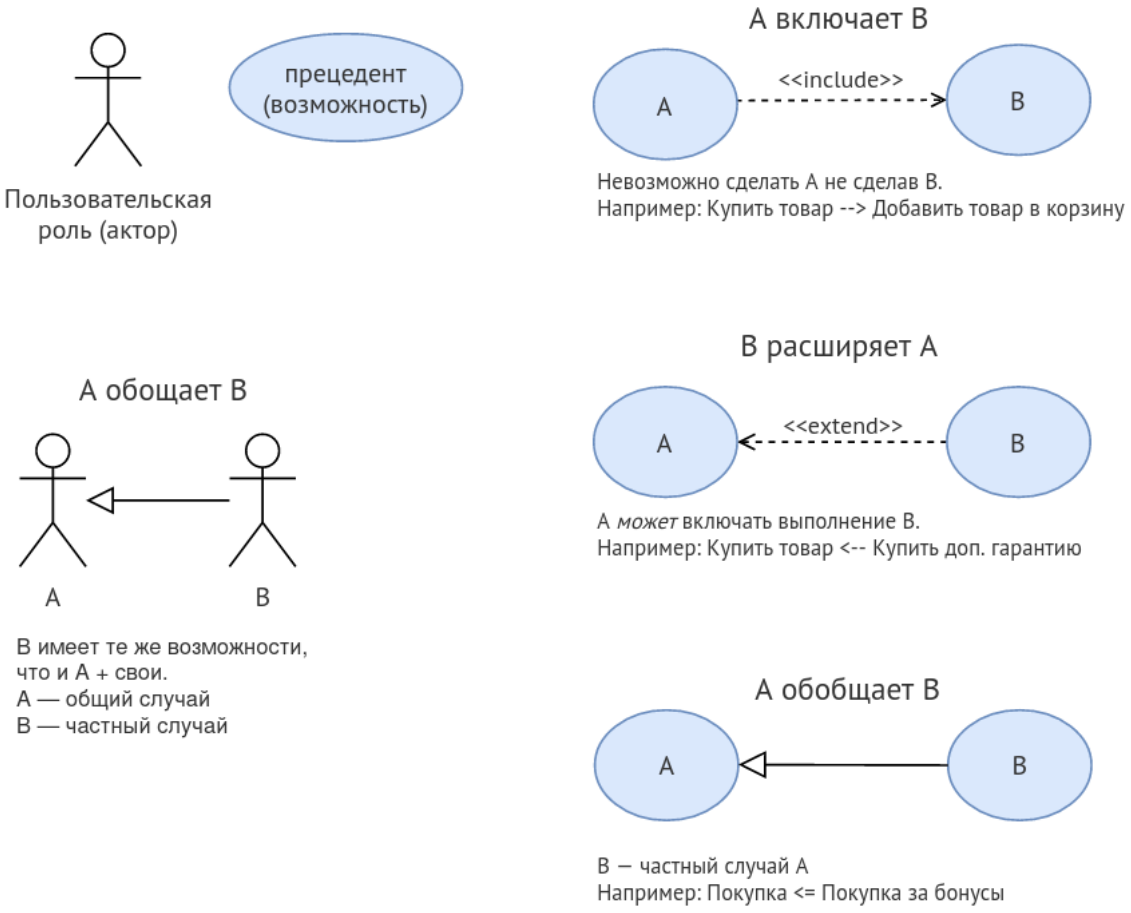


Рис. 4.3. Элементы диаграммы вариантов использования и отношения между ними

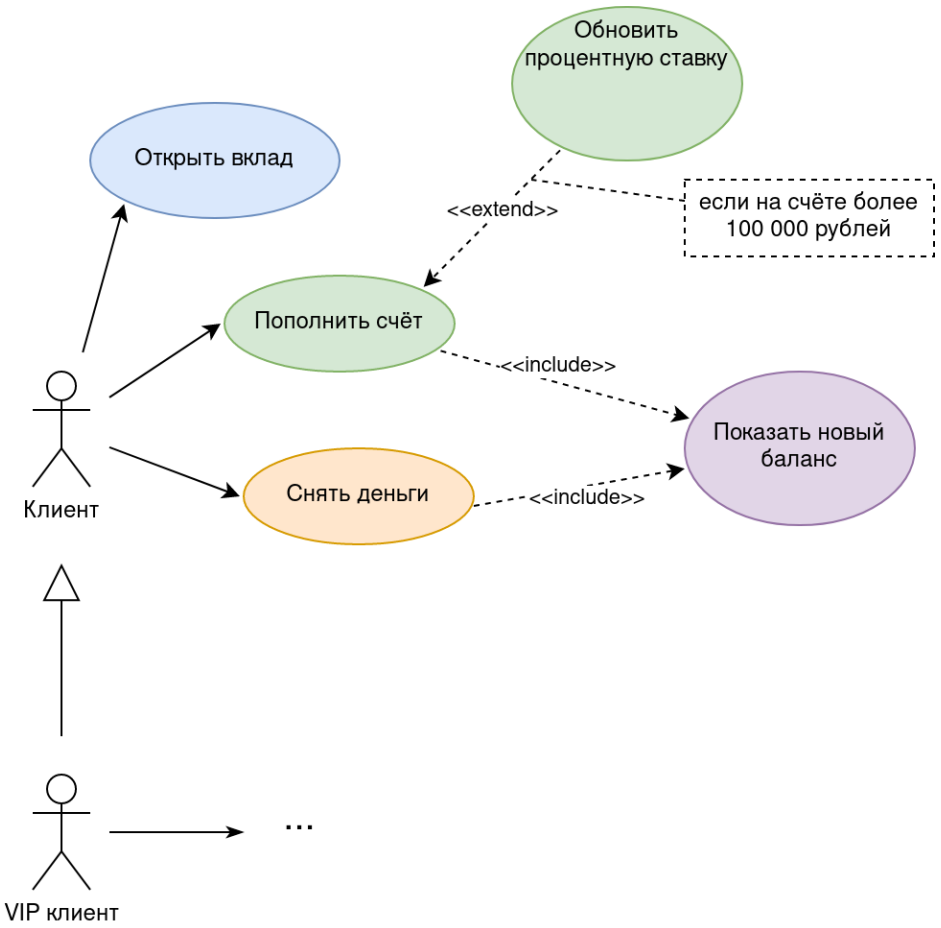


Рис. 4.4. Фрагмент диаграммы вариантов использования для банковского приложения. Extend – расширение прецедента, include – включение. Отношение между “Клиентом“ и “VIP клиентом“ – обобщение. Для удобства восприятия не связанные по смыслу прецеденты прецеденты окрашены в разные цвета.

Стрелка к первой роли – отношение обобщения – означает, что VIP клиент имеет те же возможности (дополняя их своими), что и обычный клиент.

Выполнение операций пополнения счёта и снятия средств обязательно включает (стрелка include к новому прецеденту) в себя отображение нового баланса. Нельзя изменить баланс без показа нового баланса. Обновление процентной ставки может расширить (extend) прецедент "Пополнение счёта". Но это происходит только при соблюдении условия (пунктирный прямоугольник). Можно пополнить счёт и при этом не изменять процентную ставку. Подобные отношения включения могут быть и безусловным: клиент может заказать документ с отчётом после изменения баланса. Отдельные группы прецедентов можно раскрасить, обозначив их разный смысл.

Диаграмма вариантов использования ничего не говорит об устройстве интерфейса. Но нужно понимать, какие роли и возможности будут у пользователей, чтобы проектировать интерфейс.

4.2.5 Уровень структуры

На этом уровне описывается информационная архитектура продукта в представлении пользователя, основные этапы (экраны, окна или страницы в зависимости от типа продукта) взаимодействия пользователя и программы. Далее будут рассмотрен самый популярный тип интерфейса пользователя – графический интерфейс. Создание текстового, голосового, или командного (частный случай текстового) пользовательских интерфейсов не вполне укладывается в стратегию, преложенную Гарреттом.

Информационная архитектура (Information architecture) – сочетание схем организации, предметизации и навигации, реализованных в информационной системе.

Во время создания продукта обычно описывается информационная архитектура в нескольких представлениях: диаграмма классов, модулей, структура БД и другие. Это модели реализации. Проектируя

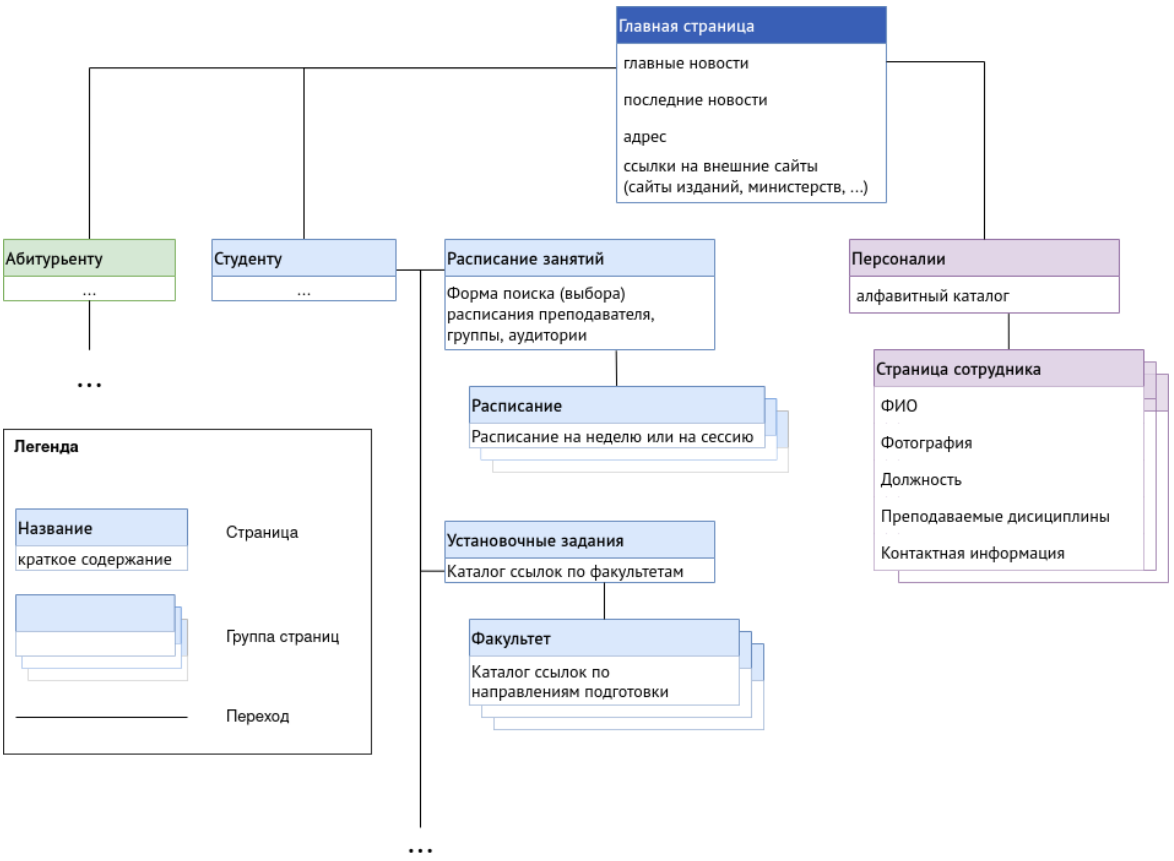


Рис. 4.5. Фрагмент диаграммы страниц сайта университета. “Страница сотрудника“, “Факультет“, “Расписание“ – набор страниц с одинаковым назначением, но разным содержанием.

пользовательский интерфейс, нужно описать модель представления – то, как продукт выглядит с точки зрения пользователя.

Информационная архитектура описывается диаграммой экранов, страниц или окон. Она состоит из блоков, обозначающих экран, с названием, кратким описанием их содержимого и переходов между этими экранами. Такая диаграмма похожа на карту сайта. Метод карточной сортировки (описаны в параграфе 5.2.4) – это один из способов построения понятной для пользователя информационной архитектуры.

По диаграмме экранов можно проследить выполнение всех прецедентов из диаграммы вариантов использования, оценить на сколько удобна и понятна такая структура для решения типичных задач персонажей.

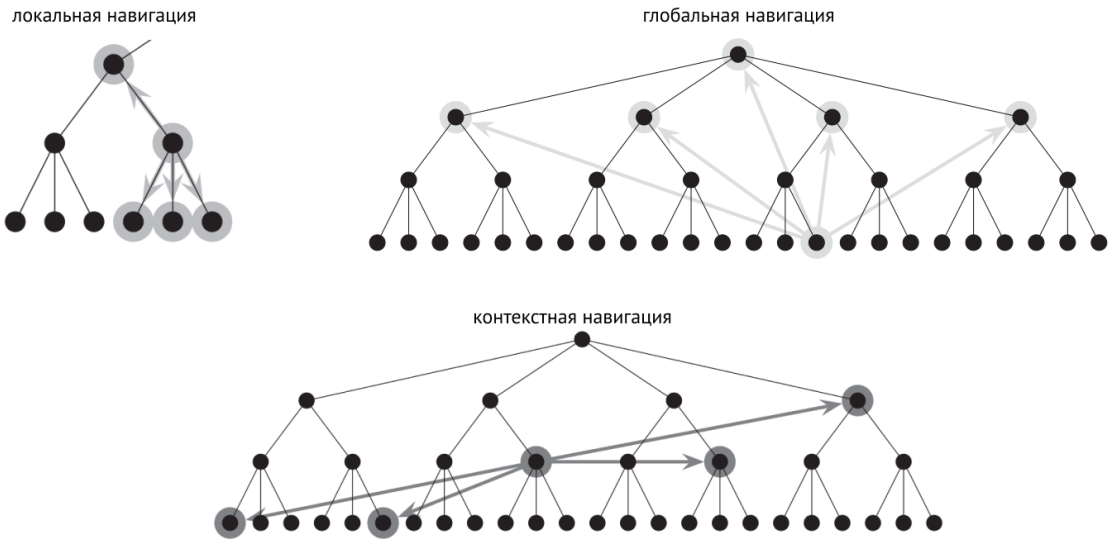


Рис. 4.6. Некоторые способы организации навигации

4.2.6 Дизайн навигации и диаграммы потоков

Навигация по окнам программы или страницам веб-сайта не обязательно должна проходить по тем же путям, которые соединяют разделы в логическую структуру. Особенно если эта структура иерархична (рис. 4.6). Путь от одного листа дерева к другому может быть длинным.

Поэтому помимо проектирования логической структуры следует заняться ***дизайном навигации***. Первый этап – организовать *локальную навигацию*. На каждой странице должна быть ссылка на страницу верхнего уровня и на страницы нижнего уровня (рис. 4.6). Например на странице подкатегории товаров в интернет-магазине должна быть ссылка на страницу категории и отдельные товары. Чтобы ускорить перемещение между разделами, которые слабо связаны друг с другом логически можно организовать отдельные “точки входа“, создав *глобальную навигацию*. Пример – глобальное меню со ссылками на разделы сайте, доступное на любой странице. Рассматривая каждую страницу или окно в отдельности, созать контекстную навигацию. Например на странице интернет-магазина с ноутбуками добавить ссылки на сопутствующие товары, которые часто покупают с ноутбуком и страницу с каталогом ПК.

Задача ***диаграмм потока*** (user flow) – представить структуру продукта в динамике, чтобы понять возможные проблемы и оптимизировать взаимодействие пользователя и продукта. Эти диаграммы похожи на запись алгоритма в виде блок-схемы. Такие диаграммы строятся на основе диаграммы информационной структуры и могут иметь разный уровень детализации:

- диаграмма потоков задач (task flow) ,
- диаграмма потока с вайрфреймами[6] (wire flow) ,
- диаграмма потока с экранами (screen flow) .

Эти диаграммы могут создаваться на этапе проектирования информационной архитектуры (диаграмма потоков задача), на этапе разработки низко детализированного макета (wireframe flow) и на этапе, когда макеты полностью готовы (screen flow). Последние два случая полезны не только для проектирования интерфеса, но и для его тетстирования.

Не существует четкого стандарта для построения таких диаграмм потока, в отличии от, например, диаграммы вариантов использования. Но можно ориентироваться на стандарт BPMN (Business Process Model and Notation, нотация и модель бизнес-процессов).

Чтобы составить последовательность, нужно ответить на три главных вопроса:

- Кто является пользователем?
- Какова его цель?
- Какие шаги он должен предпринять для достижения этой цели?

Диаграммы потоков задач (task flow) описывают маршруты или последовательности операций, выбираемые пользователями (а иногда системой) в ходе работы с продуктом [31]. Существуют разные способы использования диаграмм потоков задач. В сочетании с картами сайтов они могут показывать, как пользователь попадает на страницу, где выводится определенная информация, то есть посмотреть не на статичную структуру продукта, а на процесс его использования.

[6]вайрфреим (wireframe) – макет интерфейса с низким количеством деталей

Символы	Наименование	Функция
	Начало/Конец	Для представления начальной или конечной точки
	**Стрелка**	Для представления связи между фигурами
	Ввод/Вывод	Для представления ввода/вывода информации
	Процесс	Для представления процессов
	Условие	Указывает на условие принятия решения

Рис. 4.7. Основные элементы диаграммы потоков задач

Иногда диаграммы потоков задач показывают, как пользователь конкретного типа (персонаж) будет перемещаться по сайту и что именно, исходя из его личной ментальной модели, он рассчитывает увидеть.



Рис. 4.8. Диаграмма потока задачи (task flow)

Диаграммы мы потоков задач полезны для идентификации сложных процессов, в которых необходимо досконально разобраться до передачи проекта команде разработчиков.

Чтобы построить диаграмму потоков задач, необходимо понимать цель пользователя. В одних случаях получится документ, описывающий требования, в других – типичный сценарий использования. Хотя сценарий использования состоит всего из несколько фраз, кратко суммирующих задачи и цели, он поможет преобразовать

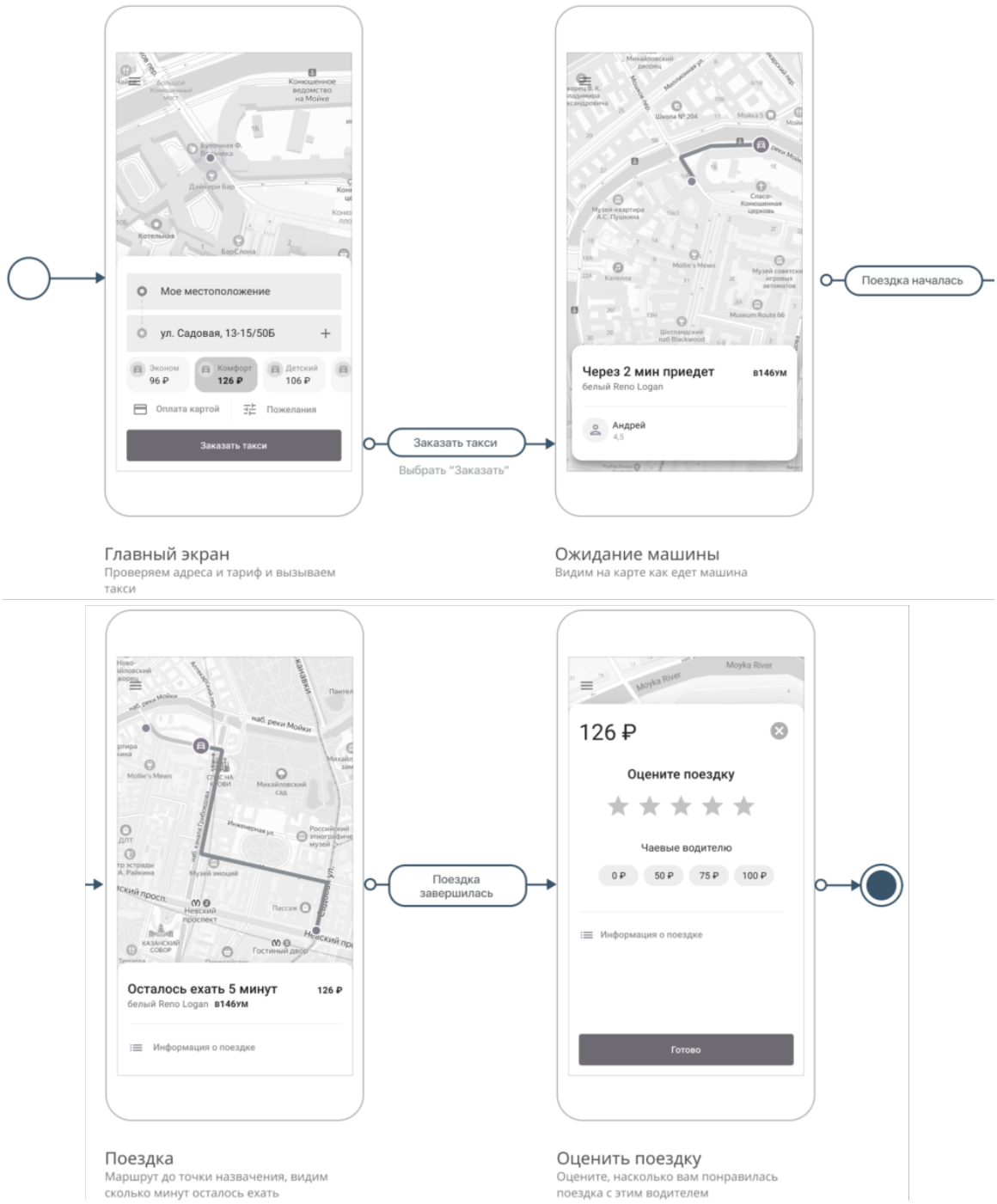


Рис. 4.9. Диаграмма потока (screen flow) заказа такси в мобильном приложение представленная в виде последовательности экранов и действий.

пользовательские представления о системе в реальный опыт взаимодействия.

Диаграммы потоков удобнее всего строить в программах или веб-сервисах ориентированных на создание UML диаграмм (например draw.io) или в программах для проектирования пользовательского

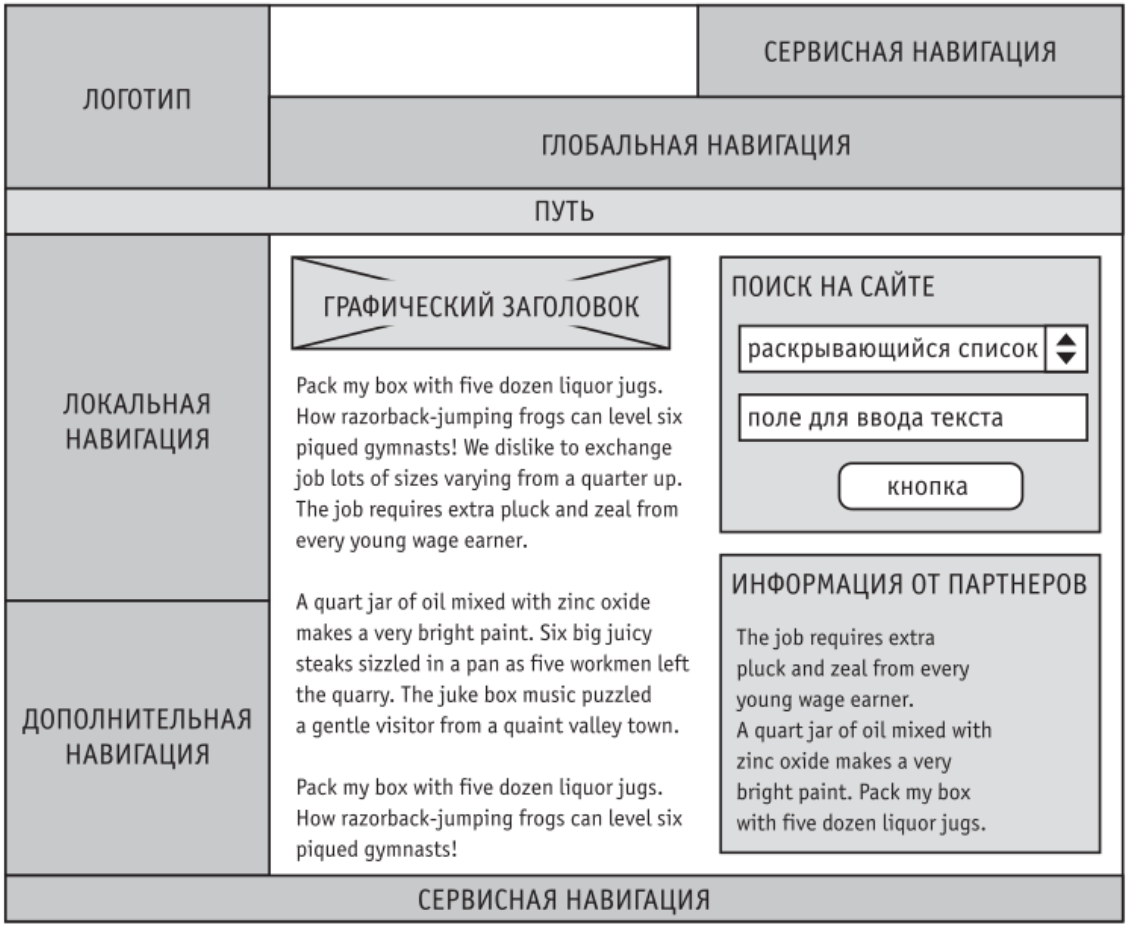


Рис. 4.10. Эскиз окна веб-страницы: видна компоновка элементов интерфейса, основные элементы подписаны, изображение приведено схематично, палитра не задана.

интерфейса. В Figma для диаграмм потока задач и диаграмм информационной архитектуры существует отдельный вид проектов – jamboard.

4.2.7 Уровень компоновки

На этом этапе определяется общий вид интерфейса интерфейс пользователя, примерное расположение основных элементов интерфейса пользователя. Компонуются экраны после составления диаграммы информационной архитектуры. Экраны представляются низко детализированными (low fidelity, lo-fi) макетами – вайрфреймами (от англ. wireframe), главная цель которых – показывать структуру макета, взаимное расположение элементов, но не их конечный вид (рис.4.10).

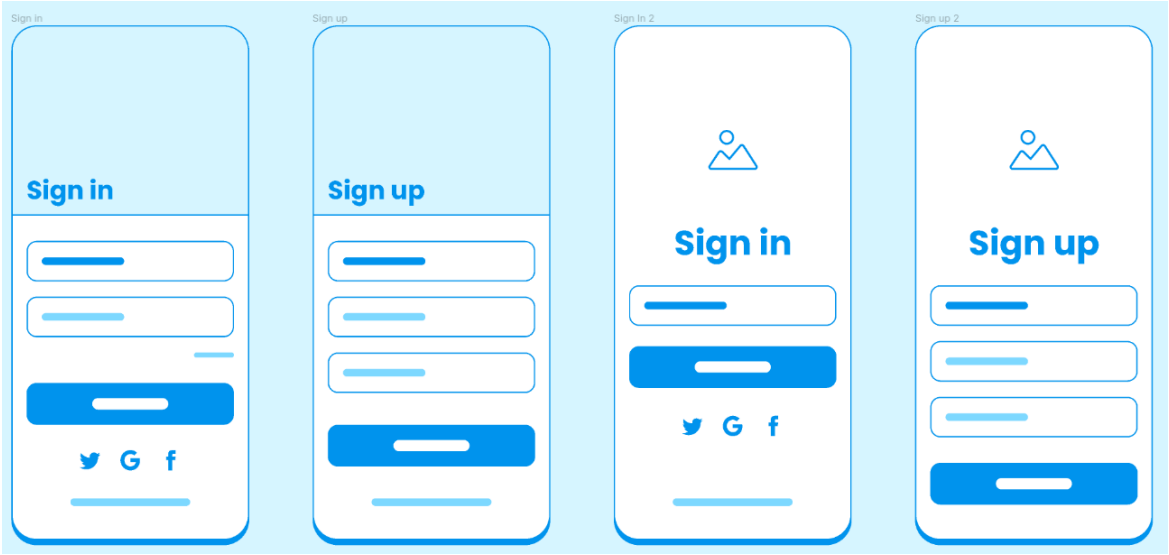


Рис. 4.11. Низко детализированные варианты экранов приложения. Выполнены в Figma, собраны из библиотеки готовых компонентов. Обозначены элементы интерфейса, расположение отдельных надписей, выбранная палитра играет техническую роль и не отражает конечный вид интерфейса.

Низко детализированные эскизы полезны для быстрого создания экранов. Легко создавать несколько вариантов одного экрана (рис. 4.11), менять их дизайн, не подгоняя элементы друг к другу. Дизайнер экономит время, откладывая выбор палитры, шрифтов, точных надписей и текста, иконок и т.д. для всех вариантов макетов.

На этом этапе полезно утверждать макеты у заказчика (если есть такая возможность) потому, что при более детальном проектировании интерфейса цена изменений возрастает.

Эскизы удобнее всего создавать в специальном ПО для дизайна интерфейсов, в графическом редакторе. Макеты можно нарисовать и от руки, затем легко вырезать из бумаги отдельные элементы интерфейса и экспериментировать с разными компоновками. Такое исполнение макета интерфейса для мобильных устройств с тачскринами даёт представление о физическом взаимодействии с интерфейсом (рис. 4.16).

Есть хороший способ убедиться, что визуальный дизайн эффективно задействует иерархию и отношения, – дизайнеры называют этот прием тестом с прищуриванием (squint test)[18]. Закройте один глаз

и посмотрите на экран прищуренным вторым глазом. Обратите внимание на то, какие элементы слишком выпирают, какие стали нечеткими, а какие объединились в группы. Эта процедура часто вскрывает не замеченные ранее проблемы в композиции интерфейса.

Сетка – один из самых мощных инструментов визуального дизайнера, стремительно набравший популярность в годы после Второй мировой войны благодаря швейцарским печатникам. Сетка помогает обеспечить однородность и последовательность структуры композиции, что особенно важно при проектировании интерфейса с несколькими уровнями визуальной или функциональной сложности. После того как проектировщики взаимодействия определили общую структуру продукта и элементов его пользовательского интерфейса, дизайнеры интерфейса должны организовать композицию в структуру в виде сетки, которая будет должным образом подчеркивать важные элементы и структуры и оставлять жизненное пространство для менее важных элементов и элементов более низкого уровня.

В основе композиционной сетки лежат прямоугольники (рис. 4.12). Они размечают пространство макета и чаще всего располагаются с отступами. Элементы интерфейса могут занимать пространство равное одному или нескольким прямоугольникам (рис. 4.13, 4.14). На практике для сайтов широко используется сетка из 12 колонок. Многие веб-фреимворки помогают верстать страницы сайта используя колонки, например Bootstrap 5 (рис. 4.15).

Часто композиционная сетка строится на основе модуля – прямоугольника с заданной шириной и высотой. Модуль лежит в основе композиции, определяя пространство для содержимого. Такая сетка является достаточно гибкой, чтобы учесть все необходимые вариации, при этом сохранив согласованность структуры везде, где это возможно.

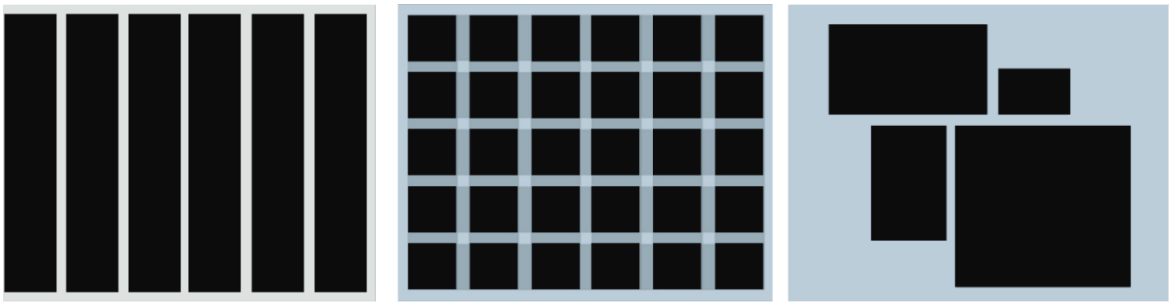


Рис. 4.12. Композиционные сетки: колоночная, модульная, блочная. Прямоугольники, размечающие пространство макета, всего располагаются с отступом друг от друга. Элементы интерфейса могут занимать пространство равное одному или нескольким прямоугольникам.

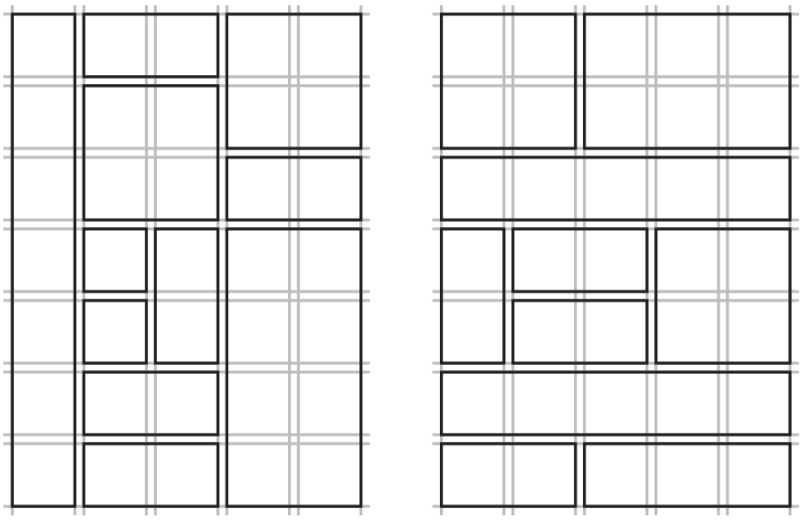


Рис. 4.13. Варианты расположения блоков в модульной сетке.

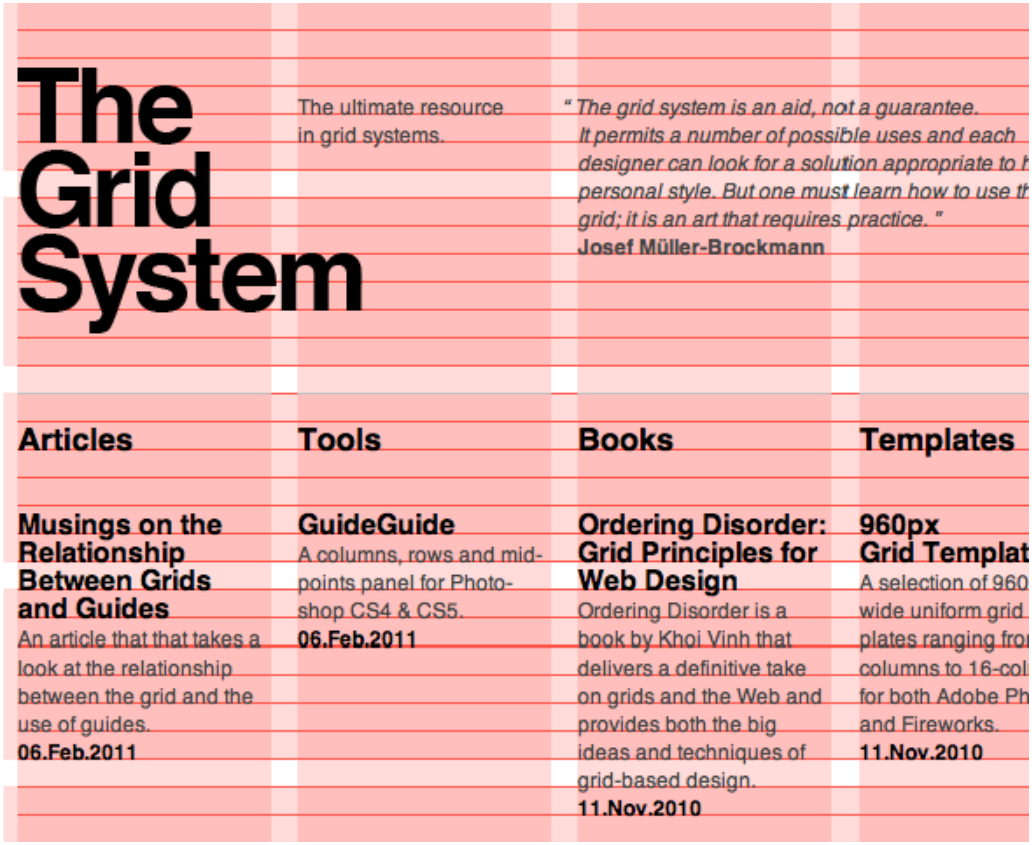


Рис. 4.14. Блоки с текстом занимают один или несколько модулей. Модули изображены розовыми прямоугольниками.

Композиция должна не только в точности следовать сетке, но и структурировать эффективный ***логический маршрут*** через интерфейс для пользователей, принимая во внимание тот факт, что (в случае западных языков) взгляд движется сверху вниз и слева направо (рис. 4.17).

4.2.8 Уровень поверхности

На этом этапе интерфейс получает свой окончательный вид. Должна быть выбрана цветовая палитра, нарисованы иконки и созданы изображения, окна или страницы наполнены содержанием.

Всё вышеописанное представляется в окончательном макете (рис. 4.19) или прототипе. Макет, демонстрирующий облик интерфейса пользователя обычно называют мокапом (mockup). Прототип, в отличие от макета, предполагает имитацию взаимодействия с поль-

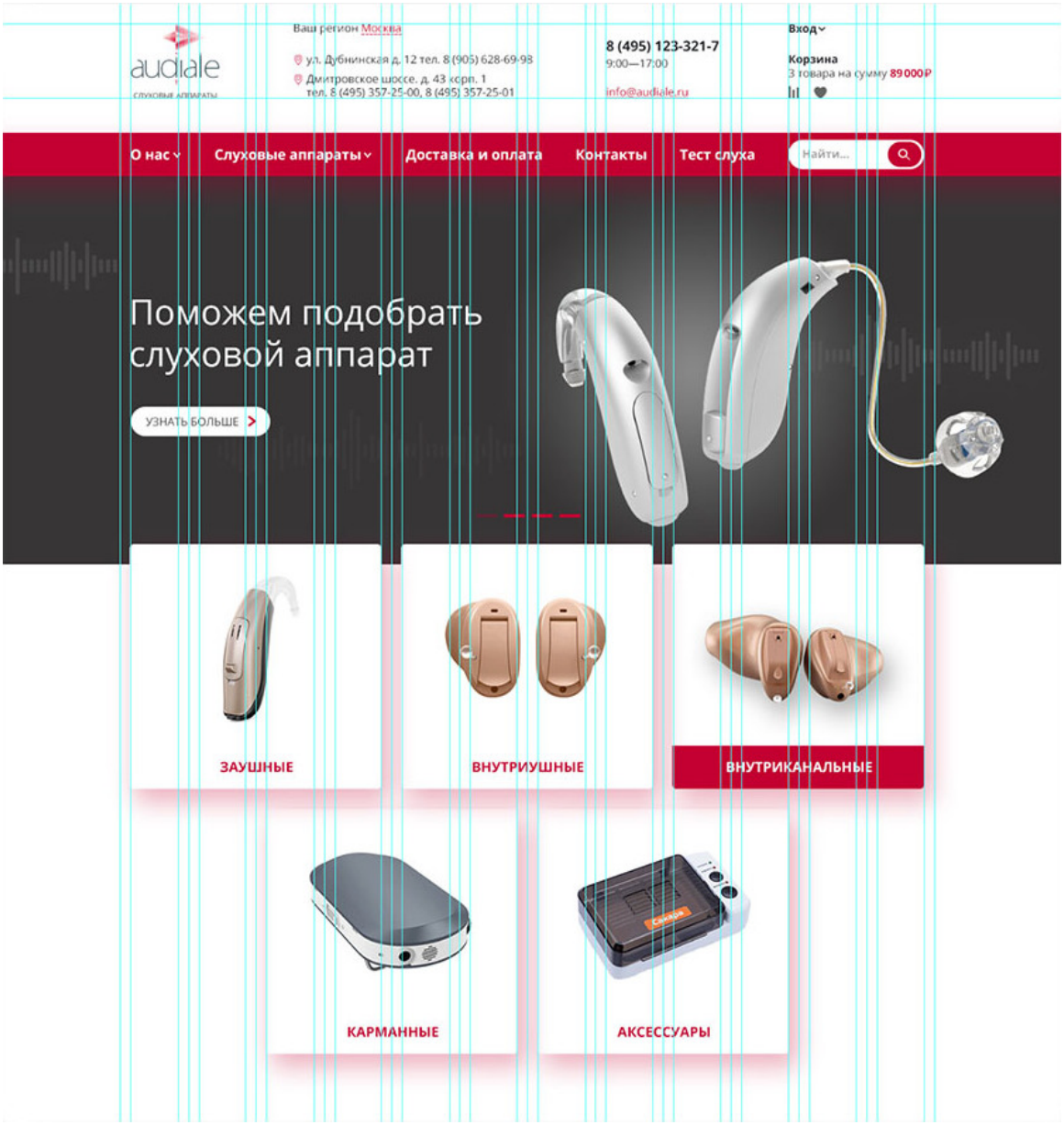


Рис. 4.15. Сетка из 12 колонок, используется в Bootstrap – свободном наборе инструментов для создания сайтов и веб-приложений

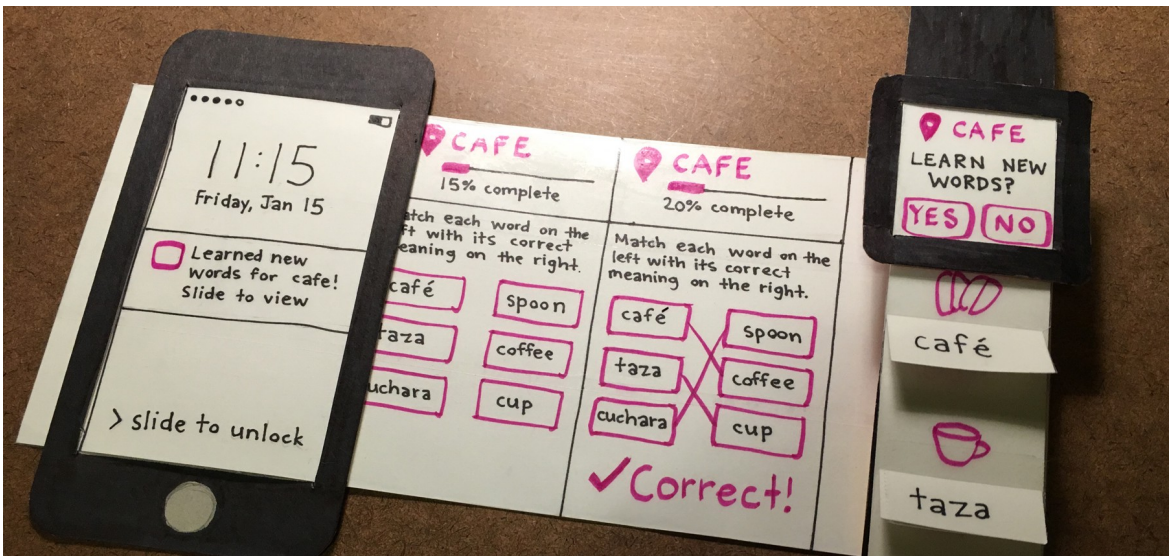


Рис. 4.16. Бумажный эскиз пользовательского интерфейса

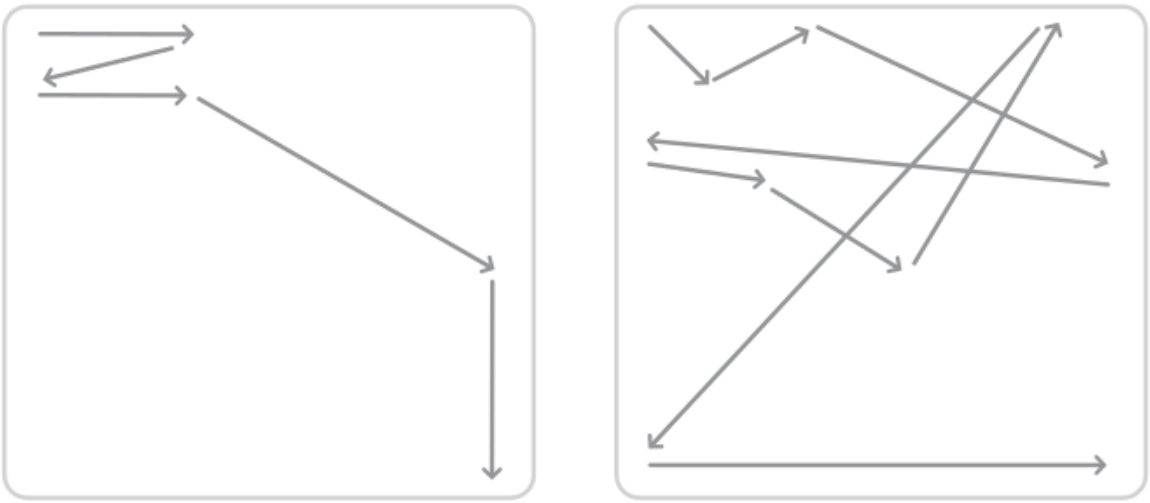


Рис. 4.17. Хороший, логический маршрут (слева) взаимодействия с элементами интерфейса одного окна или страницы направлен слева на право, сверху вниз, не требует от пользователя беспорядочного перемещения взгляда (справа)

зователем. Например, интерактивные переходы со страницы на страницу, прокрутку внутри окон, раскрытие меню и т.п.

Готовый макет используется для реализации интерфейса программы или сайта.

Макеты пользовательского интерфейса нужны и полезны разработчикам (программистам), руководителям проекта и заказчику. Причём, для них последних имеет смысл создать аннотированный

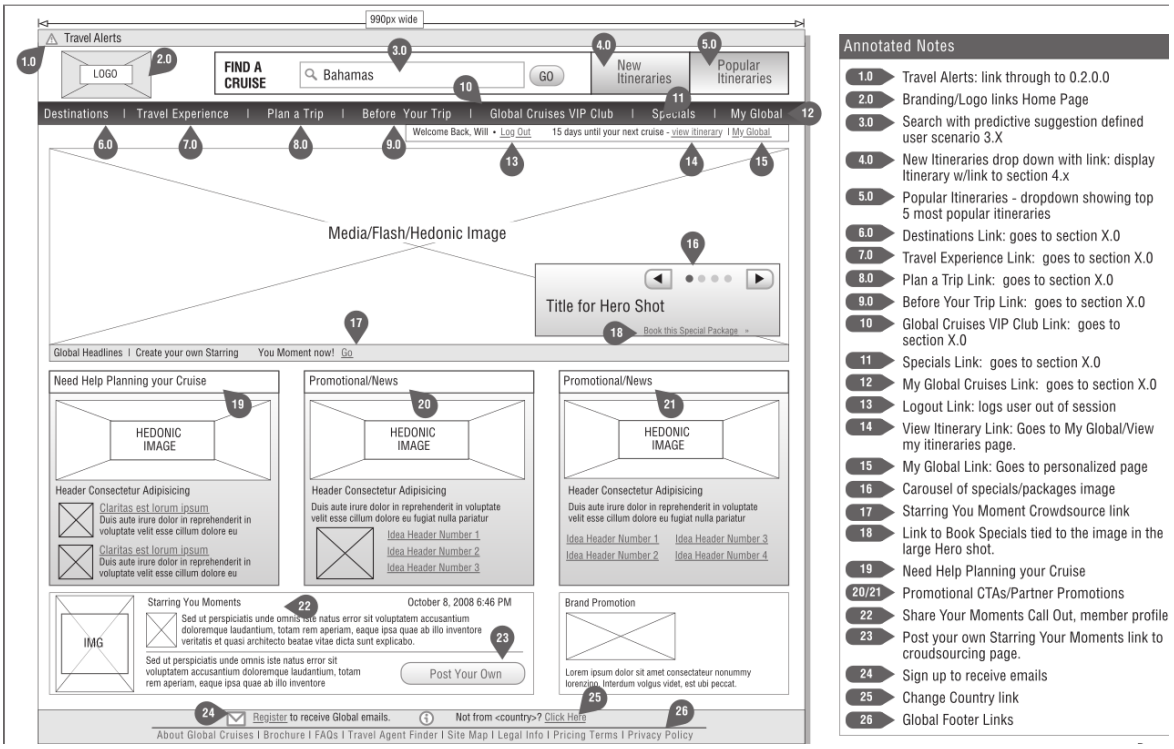


Рис. 4.18. Аннотированный макет. Аннотации необходимы для статичных макетов, где нельзя понять назначение и принцип работы элементов интерфейса т.к. взаимодействие с ними ещё не реализовано.

макет, в котором описано назначение и поведение элементов интерфейса (рис. 4.18).

4.2.9 Основы работы в Figma

Частая ошибка – приступать сразу к реализации крупного интерфейса (например на HTML и CSS, или в дизайнере форм IDE), пропуская этап проектирования. Из-за большого числа вариантов исполнения и гибкости процесса проектирования ПИ приходится часто вносить изменения, создавать множества вариантов ПИ для сравнения, так как сложно дать абсолютную оценку качества интерфейса.

Поэтому работа по проектированию интерфейса, в первую очередь для веба, где у дизайнера особенно много возможностей, делится на два этапа – создание макета и реализация макета (в области веб-дизайна это называется верстkой). Исторически для создания макетов широко используются графические редакторы, например Photoshop. В 2010-х начали появляться редакторы векторной гра-

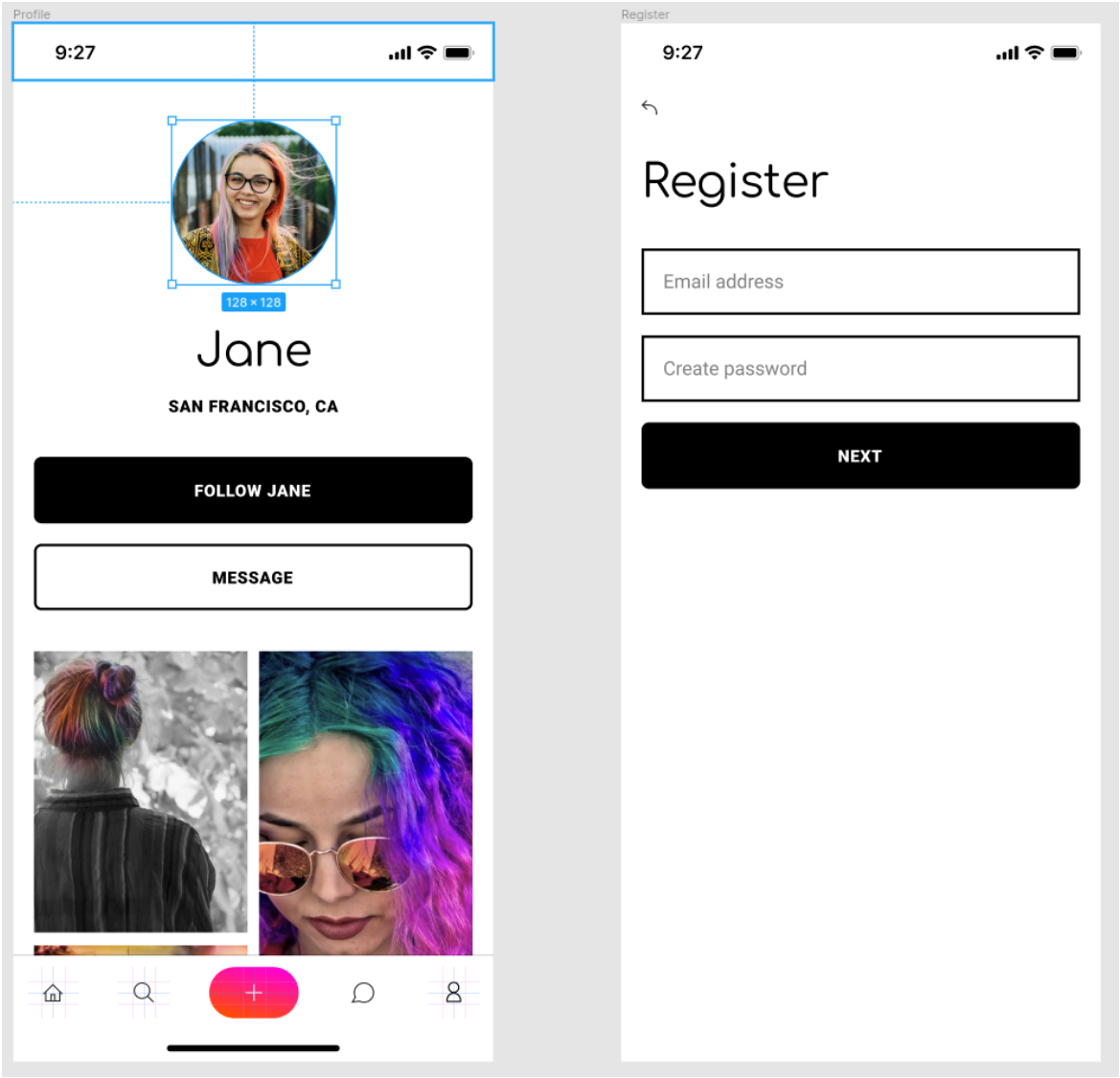


Рис. 4.19. Макеты пользовательского интерфейса в Figma. Отдельнай страница – отдельный фрйем.

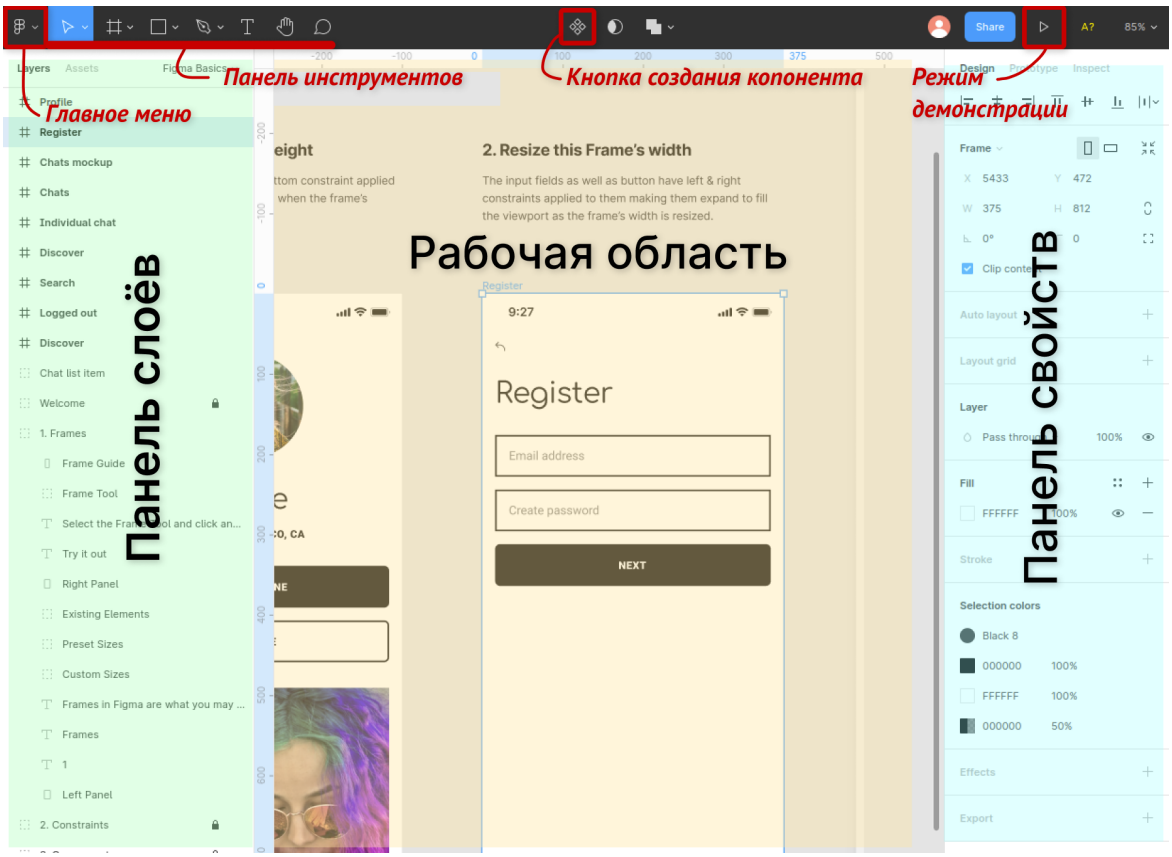


Рис. 4.20. Интерфейс Figma. Сверху (тёмная панель) – панель инструментов, настройки доступа к макетам. Слева – панель слоёв и страницы. В центре – рабочая область. Справа – панель свойств элемента из рабочей области.

фики, ориентированные на проектировании интерфейсов. Sketch и Figma – наиболее популярные из них.

Далее будут коротко рассмотрены основы работы в Figma (рис. 4.20). Полное руководство по Figma можно найти в официальной документации, или, например, в книге Руководство по Figma [29].

Figma — онлайн-сервис для разработки макетов и прототипов пользовательских интерфейсов. Сервис поддерживает групповую разработку, режим демонстрации макета незарегистрированным пользователям, содержит большую библиотеку плагинов, готовых шаблонов и элементов интерфейса. В Фигме можно создавать проекты двух типов: дизайн ПИ (кнопка New Design File на стронаце проектов) и файлы диаграмм (кнопка New FigJam File). Все изменения проектов автоматически сохраняются в облаке.

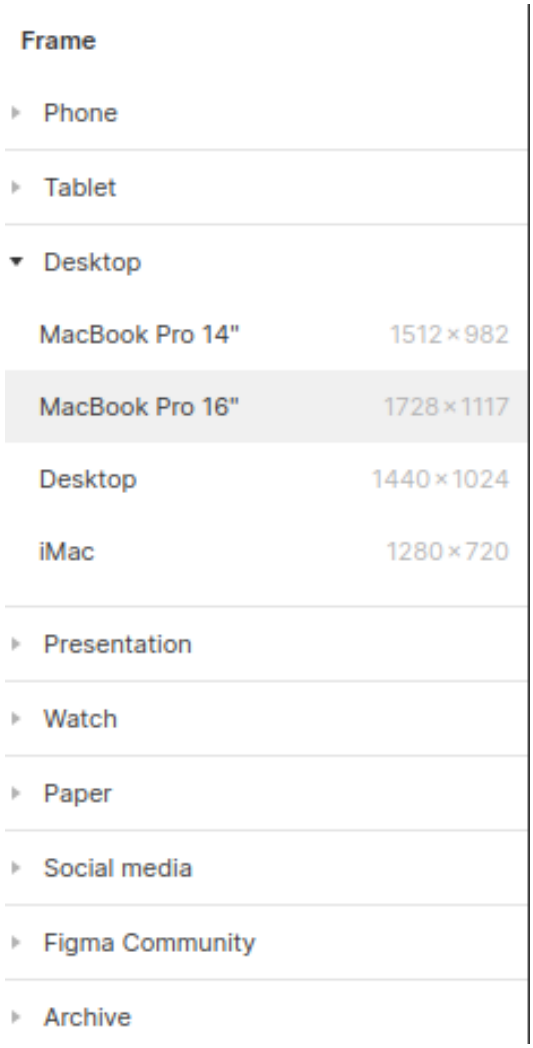


Рис. 4.21. Шаблоны для фреймов (frame). Как правило ширина фрейма сохраняется во время разработки макета, а высота изменяется в зависимости от контента

Фреймы

Фреймы (frame) – главные строительные элементы макета (рис. 4.20). На панели слоёв обозначаются знаком #. Фреймы, в первую очередь, используются для того чтобы представлять отдельную страницу или окно приложения. Доступны шаблоны фреймов (рис. 4.21). Именовать фреймы и все вложенные в них элементы рекомендуется так, чтобы по названию можно было судить о их назначении. Допустимы названия с пробелами и на русском языке.

Хорошая практика – выстраивать структуру фрейма по сетке. Сетки помогают соблюсти принципы восприятия, описанные в параграфе 3.2, сделать его аккуратным и основанным на правилах. Правила

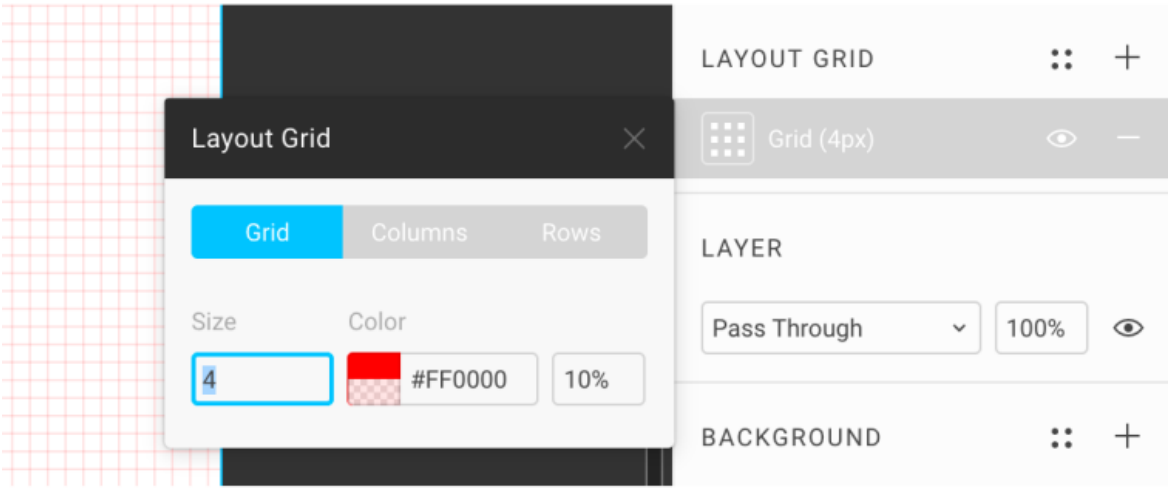


Рис. 4.22. Свойство фрейма: Layout grid, регулярная сетка – grid

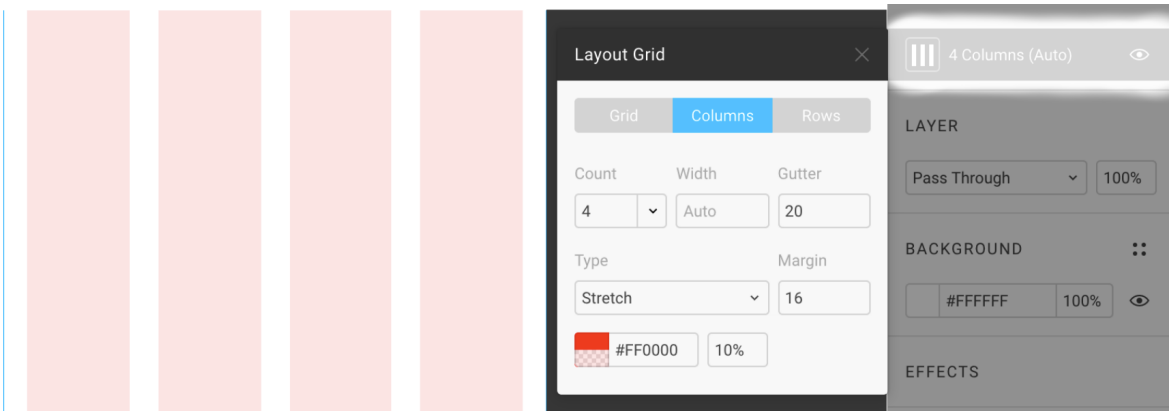


Рис. 4.23. Свойство фрейма: Layout grid, колоночная сетка – columns. Элемента интерфейса в такой сетке занимают ширину (width) одной или несколько колонок (выделены розовым). Count – количество колонок, gutter – расстояние между колонками, margin – поля слева и справа соответственно от первой и последней колонки.

расположения элементов по сетке помогают сократить[7] количество вариантов компоновки элементов, ускорить проектирование интерфейса. Если шаг сетки составляет четыре пикселя, то все расстояния между элементами и размеры элементов должны быть кратны четырем пикселям.

Сетка с маленьким шагом, от нескольких пикселей до первых десятков пикселей воспринимается плохо. Мелкие отличия в положении

[7] сокращение количества возможных вариантов компоновки элементов хоть и уменьшает творческую свободу дизайнера (чаще всего незначительно), но помогает ускорить его работу, позволяя тратить меньше времени на точное расположение блоков

и размерах могут вызвать у пользователя ощущение беспорядка и шаткости. Не все пользователи осознают эти ощущения, но для большинства они способны испортить впечатление о дизайне и тем самым свести на нет преимущества сетки.

В Figma свойства сетки находятся на панели свойств фрейма, в разделе Layout grid. Существует три вида сеток. Колоночная сетка - columns. Элементы интерфейса в такой сетке занимают ширину одной или несколько колонок (на рис. 4.23 выделены розовым). Для большинства веб-страниц хорошо подходит сетка из двенадцати колонок с полями слева и справа. Поля слева и справа нужны потому, что человеку чаще всего трудно воспринимать содержимое веб-страницы, заполняющее всё ширину экрана широкоформатного монитора[8].

❖ Компоненты

Фреймы – это контейнеры для элементов интерфейса. Сами элементы пользовательского интерфейса – кнопки, меню, поля ввода и т.д. могут быть созданы самостоятельно с помощью простых инструментов (прямоугольник, надпись и т.д. рис. 4.24) или использованы из любой доступной библиотеки figma.com/community/ui_kits.

Ещё одна хорошая практика – повторное использование ранее созданных элементов. Например кнопок, меню, т.е. элементов с повторяющейся структурой: форм, карточек товара, диалоговых окон. Сохраняя общую структуру эти части интерфейса могут иметь разное содержимое. Например на рис. 4.28 это поля ввода, все кнопки с надписью, ***Компоненты*** в Figma – это своего рода аналог класса в программировании. Компонент определяет общую структуру элемента и, чаще всего, задаёт его основные свойства – формы, цвета, текст и т.п. При этом можно создавать экземпляры компонента, которые будут повторять такую же структуру но могут отличаться отдельными свойствами, например цветом.

[8]см. также параграф «Выравнивание текста и длина строки» в главе «Типографика текст»

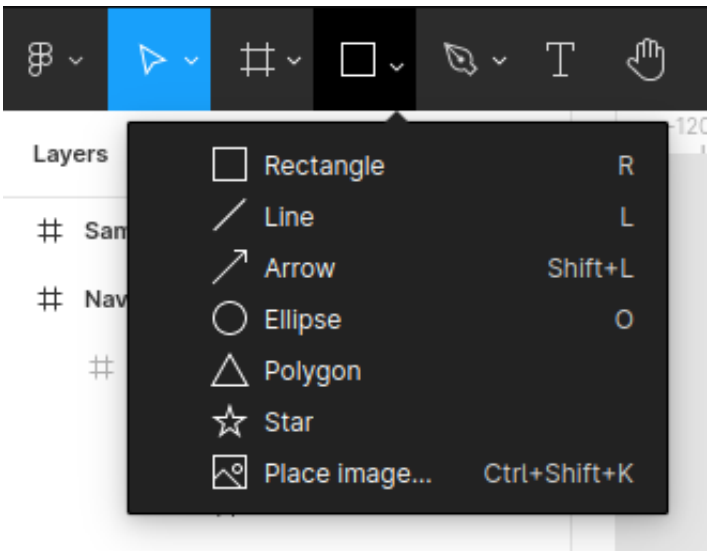


Рис. 4.24. Простые фигуры для рисования элементов интерфейса

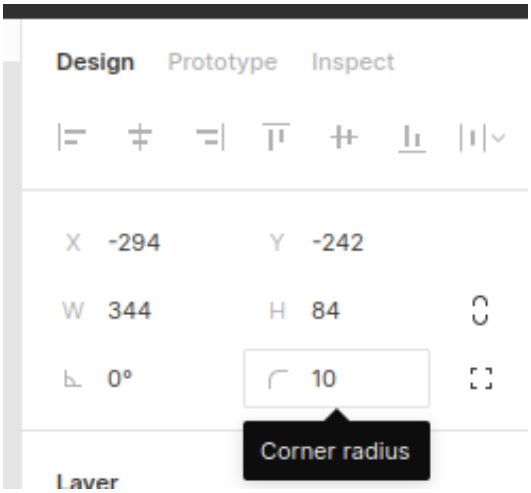


Рис. 4.25. Задание радиуса скругления (corner radius) углов прямоугольника

Рассмотрим создание компонента для кнопки. Создадим прямоугольник (инструмент Rectangle, горячая клавиша R). Зададим цвет через свойство Fill и закругление углов (рис. 4.25). Добавим текст (горячая клавиша T), задав размер, цвет с достаточным контрастом с фоном. Зададим выравнивание элементов по центру друг друга: выбрать все элементы, выровнять по горизонтальному и вертикальному центру 4.26. Не снимая выделения создадим компонент (alt + ctrl + K) или нажав значёк в виде составного ромба на панели инструментов (рис. 4.20).

Текст надписи на кнопке должен быть привязан к центру компонента (рис 4.26). Это избавит от необходимости передвигать текст вручную при изменении размеров компонента.

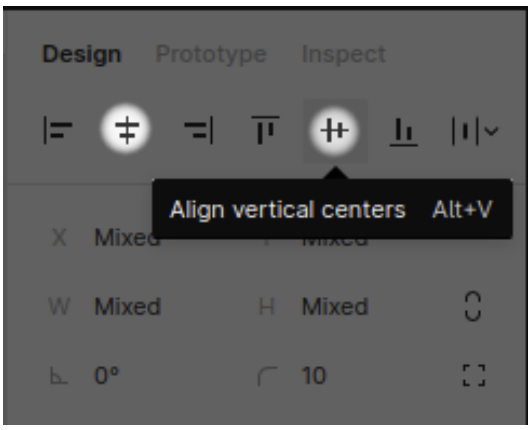


Рис. 4.26. Выравнивание выбранных элементов относительно общего центра

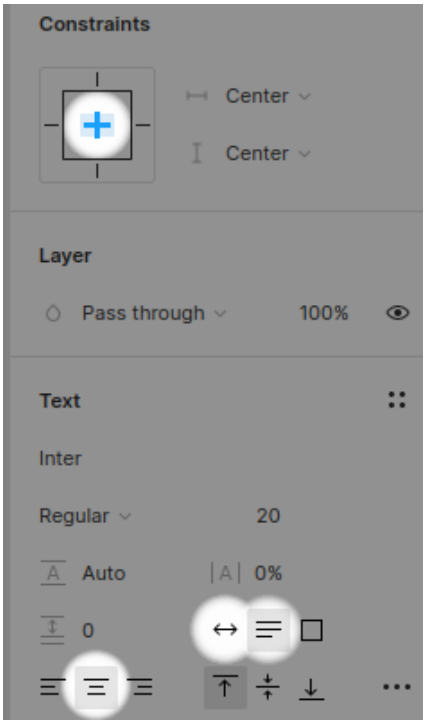


Рис. 4.27. **Constraints**: привязка (+) элемента к центру компонента, при изменении размеров компонента текст останется в его центре. **Text**: задать автоматический (минимально возможный) размер текстового блока и расположение текста по центру этого блока.

Если есть вероятность, что в процессе дизайна компонент изменит свой размер, то полезно привязать положение вложенных элементов к его границам или центру (рис. 4.27, раздел Constraints). Чтобы привязка текстового элемента выполнялась по его фактическому размеру, стоит задать у него минимально возможный размер (рис. 4.27, раздел Text), выровнять текст по центру элемента.

Создадим три копии компонента, изменив во второй копии текст и размер, в третей – текст и цвет (рис. 4.28). Изменение размеров компонента повлияет на первый экземпляр (“Войти“) и третий (“Удалить“). Так как у второго компонента собственный размер, то это свойство больше не связано с компонентом. Изменение текста компонента повлияет только на первый экземпляр (“Войти“), а

изменение цвета фона на второй и третий. Изменение остальных свойств, совпадающих у компонента и экземпляра будет синхронным. Например изменение радиуса скругления прямоугольника и шрифта текста.

Компоненты стоит хранить вне фреймов, чтобы их было легко найти. А изменения компонента оценивать по его экземплярам, расположенным на фреймах среди других элементов интерфейса, чтобы учитывать контекст. Если экземпляр компонента нужно значительно изменить, его можно отвязать от самого компонента, выбрав в контекстном меню экземпляра пункт Detach instance.

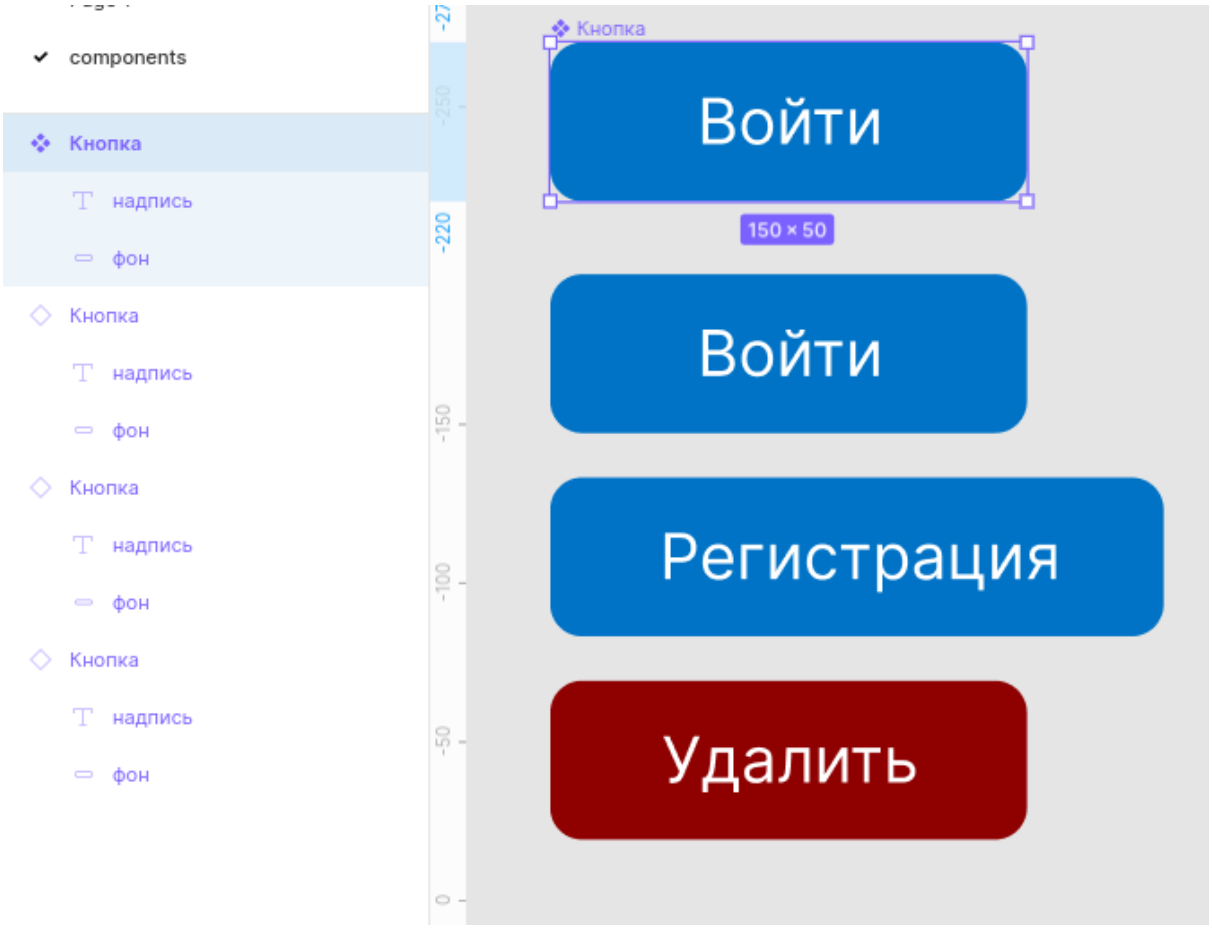


Рис. 4.28. Компонент ❖ **Кнопка** и его ◇ экземпляры (копии)

В процессе дизайна интерфейса приходится рассматривать разные варианты внешнего вида многих элементов интерфейса, или использовать несколько похожих друг на друга вариантов одного и того же элемента. В Figma для этих целей создаются ◆ ***варианты*** (variants) компонентов (рис 4.29): выделить компонент, в свойствах

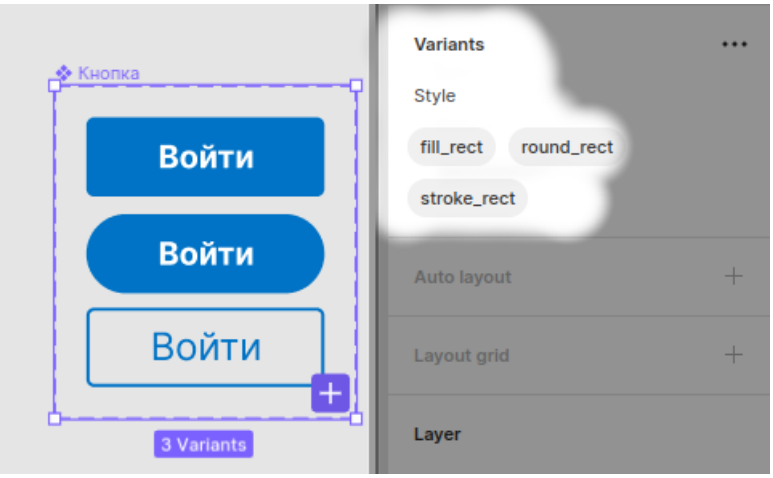


Рис. 4.29. Варианты компонента “Кнопка“. Каждый вариант формально отличается от другого значением свойства Style (название задано в свойствах компонента): fill_rect, round_rect, stroke_rect.

нажать + напротив Variants. На рисунке каждый вариант формально отличается от другого значением свойства Style (название задано в свойствах компонента): fill_rect, round_rect, stroke_rect. Экземпляры одного компонента легко заменять на его варианты: контекстное меню экземпляра > change variant. При этом изменятся только свойства экземпляра совпадающие с его первоначальным вариантом, но не будут изменены оригинальные свойства экземпляра.

Figma позволяет использовать сетку внутри компонентов или их экземпляров (свойство Layout Grid). Как и в случае с фреймами сетка может быть комбинированной, например состоять из вертикальной и горизонтальной разметки (рис. 4.30).

Изображения

Изображения тоже могут быть частью макета. Удобнее всего вставлять в макет изображения из буфера обмена. Для заданий свойств и изменения изображения нужно отредактировать раздел Fill на панели свойств (рис. 4.31). Этот же раздел отвечает за заполнение других объектов цветом. Существуют несколько способов заполнения изображением ограничивающего прямоугольника: Fill – заполнение, Fit – заполнение с машстабированием, Crop – заполнение с обрезкой, Tile – заполнение с повторением (плиточное заполнение).

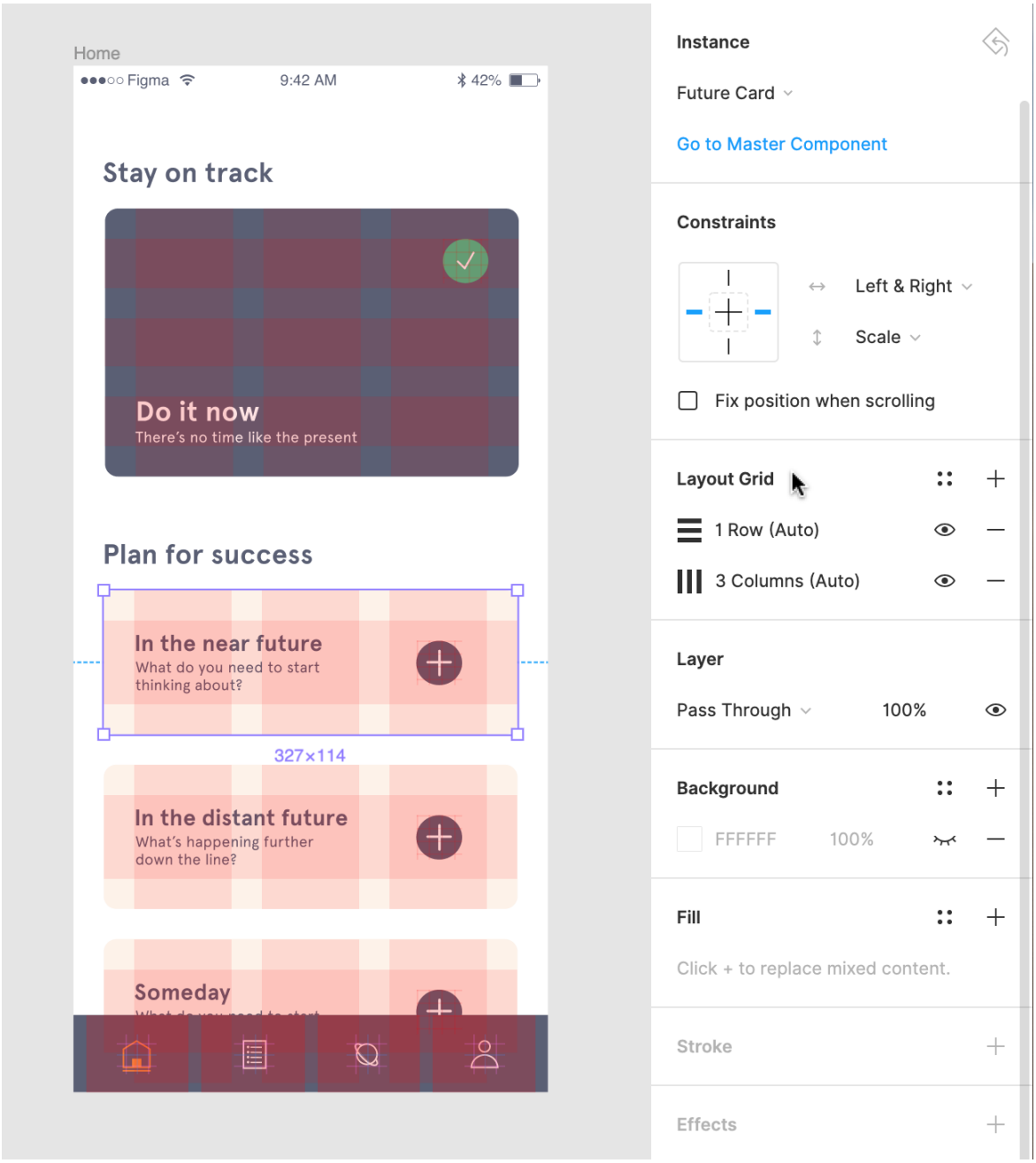


Рис. 4.30. Комбинированная (вертикальная и горизонтальная) сетка (layout grid) для компонентов

Изображения легко заменять: копировать в буфер обмена новое изображение, выделить заменяемое, “Paste to replace (Ctr + Shift + R)“ в контекстном меню. Внутри компонента можно заменить одно изображение другим: выбрать изображение, на панели свойств, в разделе Fill (рис. 4.31, справа) нажать на миниатюру изображения (image), в появившемся окне при наведении на картинку появится кнопка “Choose Image“). Вставленные из буфера обмена изображения можно сохранять в отдельные файлы на локальный диск через раздел свойств Export.

⁚⁚ Стили

Отдельные элементы макета должны иметь одинаковые свойства, для них нецелесообразно создавать отдельный компонент. Например надписи на всех макетах должны должны иметь один цвет и шрифт, или разнообразные панели и кнопки должны иметь одинаковый цвет фона.

В Figma возможно повторное использование стилей текста, цвета, эффектов (тень, размытие и т.д). Стили создаются из существующих свойств объектов. Например создание стиля цвета происходит так: выбрать объект, на вкладке свойств, в разделе Color нажать на кнопку ⁚⁚, в появившемся окне нажать + и в новом окне задать название для цвета (рис. 4.32). Все стили отображаются на панели свойств проекта, которая отображается если не выбран ни один объект (рис. 4.33).

Если в Figma создаётся макет сайта, то можно экспортировать свойства объектов в CSS. Описание стилей объектов содержится на панели свойств, на вкладке Inspect (рис. 4.34).

⟜ Интерактивность

В отличии от многих векторных редакторов Figma позволяет создавать прототипы интерфейса, имитирующие работу продукта. Например, открытие окна или переход в новое окно по клику на

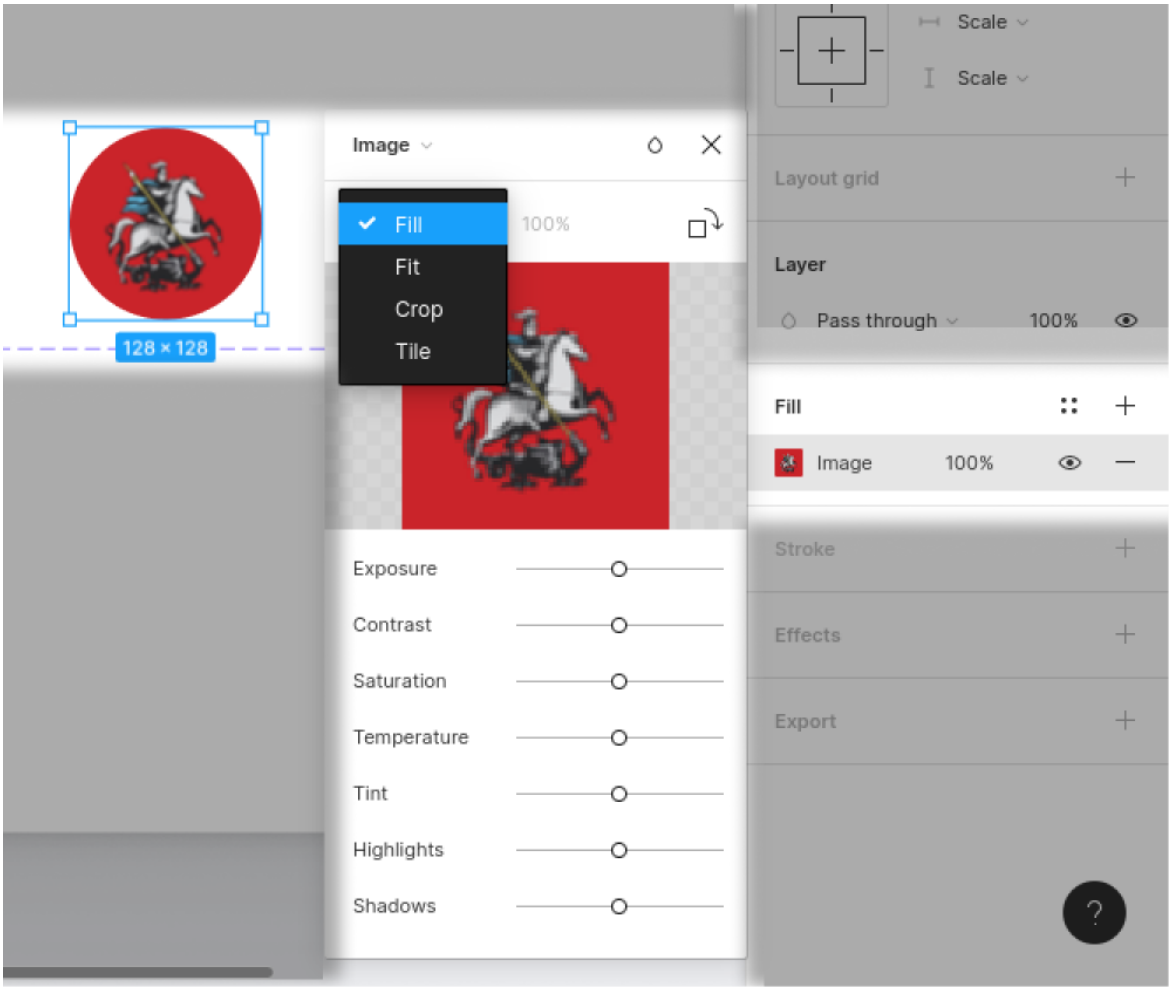


Рис. 4.31. Изображения в Figma настраиваются через свойство Fill, которое также отвечает за заполнение других объектов цветом. Виды расположения изображения: Fill – заполнение, Fit – заполнение с машстабированием, Crop – заполнение с обрезкой, Tile – заполнение с повторением (плиточное заполнение).

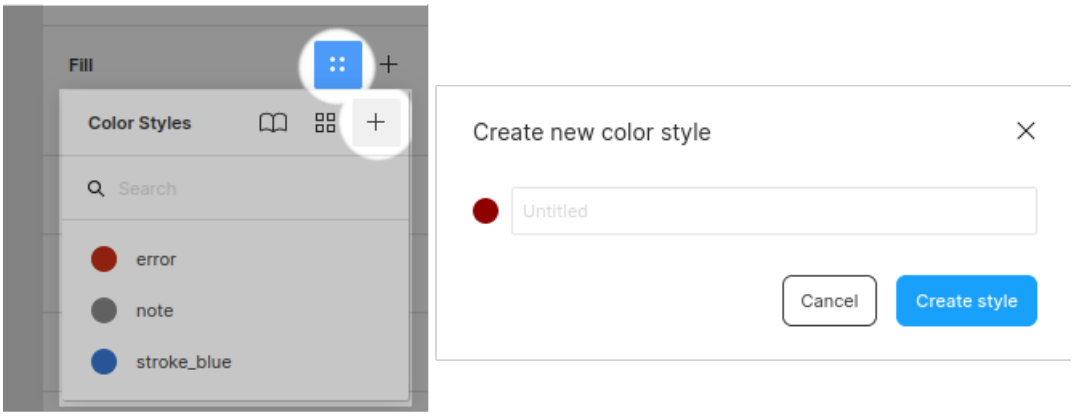


Рис. 4.32. Панель свойств, создание стиля для цвета

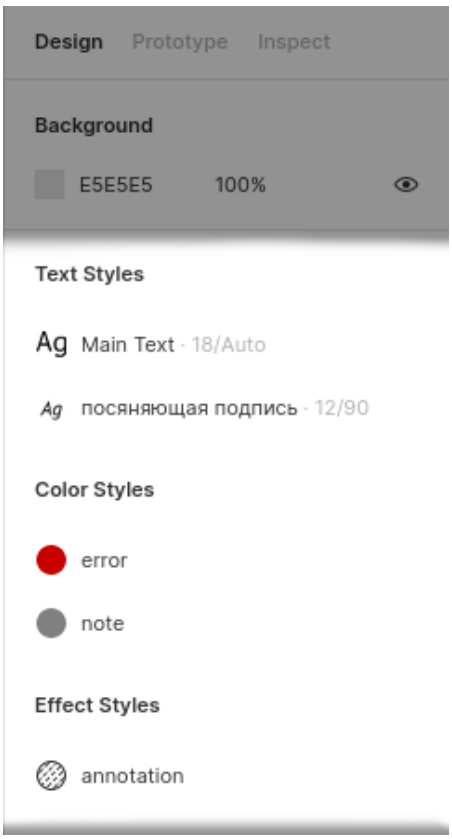


Рис. 4.33. Перечень существующих стилей на панели свойств проекта (показывается когда не выделено ни одного объекта)

кнопку. Интерактивный функционал доступен в режиме демонстрации макета (рис. 4.20) Рассмотрим создание перехода: выделим кнопку, на панели свойств переключимся на вкладку Prototype (шаг 1 на рис. 4.36), включится режим редактирования прототипа. У каждого элемента будет нарисована окружность (рис. 4.35), потянув за которую можно указать на любой фрейм. По умолчанию фреим будет показан после клика на выбранный элемент. В дополнении можно настроить друге действие и анимацию (шаг 3 и 4 на рис. 4.36), которые будет выполнены при взаимодействии с элементом.

Помимо режима демонстрации макета (кнопка share, рис. 4.20) Figma даёт возможность экспортировать фреймы или наборы объектов в растровые или векторные форматы изображений, pdf. При экспорте можно уменьшить или увеличить разрешение. Это может пригодится для того, чтобы поделиться отдельными макетами или чтобы обратно вставить начальные версии макета обратно в Figma для сравнения с новыми версиями и наглядного преставления изменений.

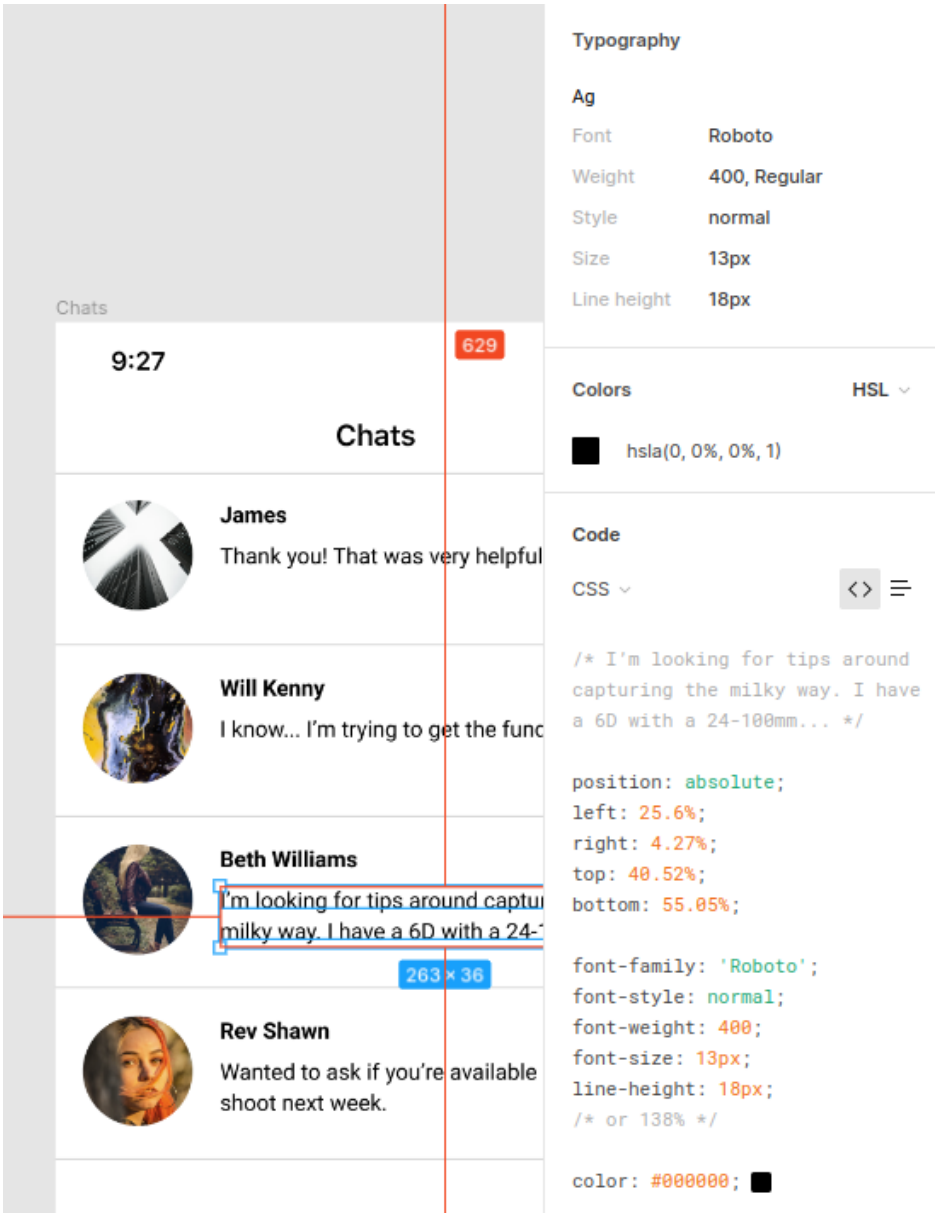


Рис. 4.34. Figma автоматически создаёт CSS код для описания свойств каждого объекта

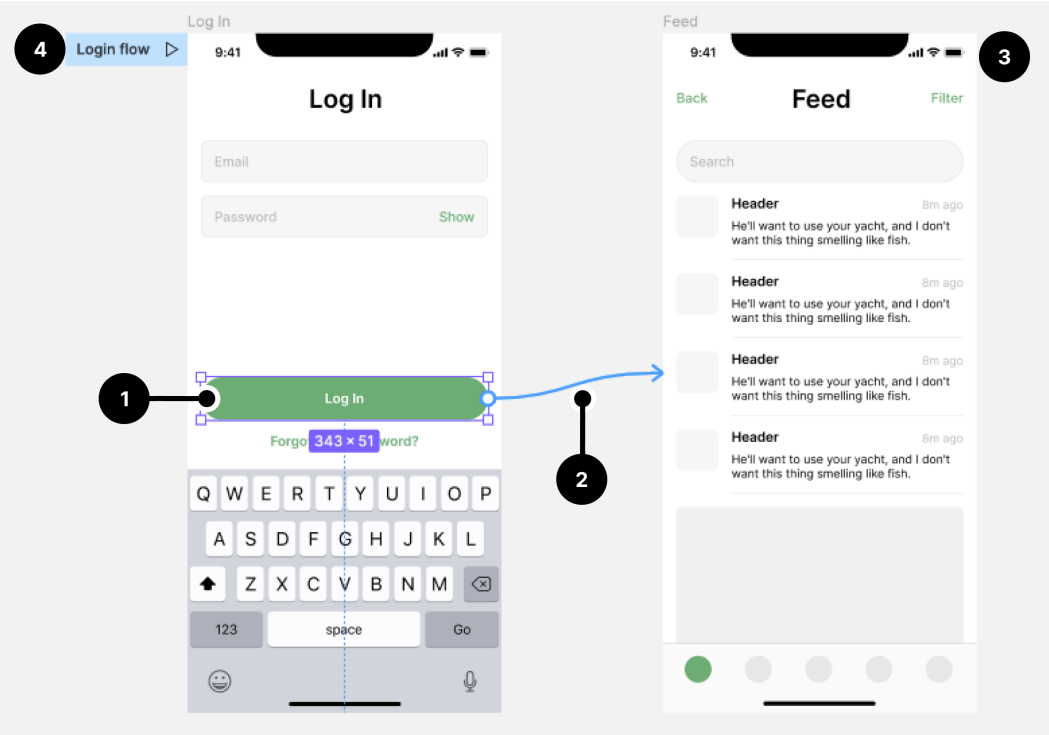


Рис. 4.35. Режим редактирования прототипа: создание перехода на новый фрейм по клику на кнопку

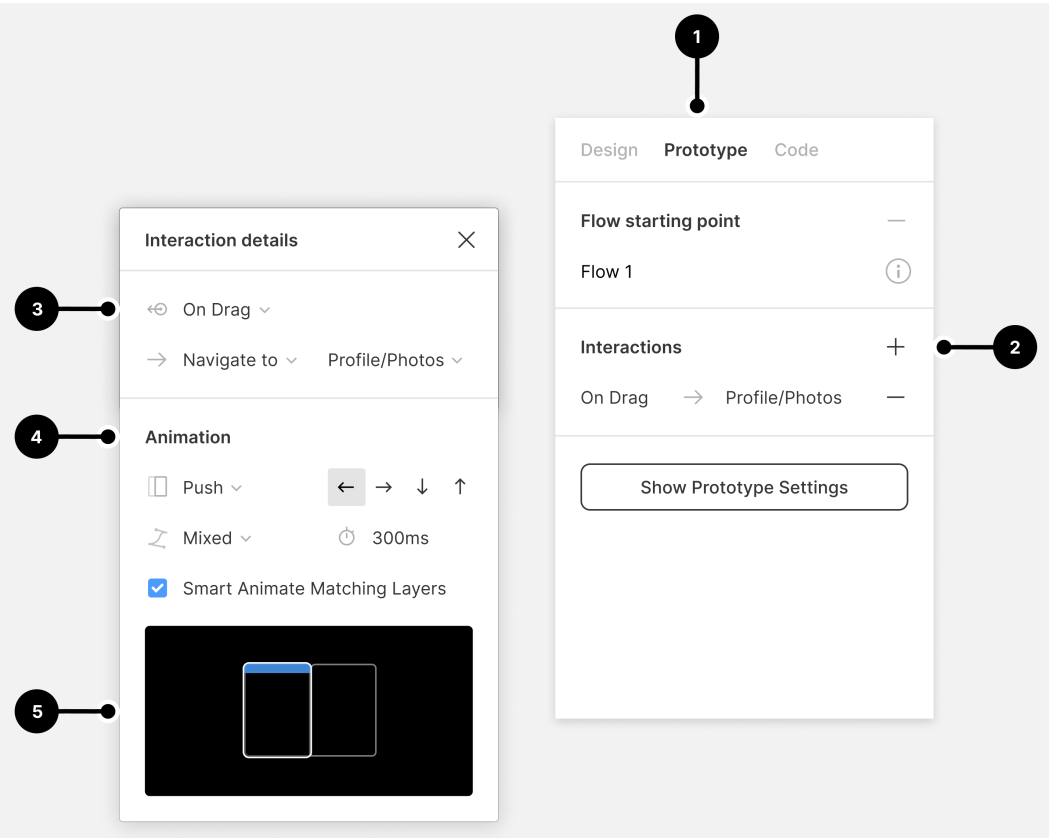


Рис. 4.36. Режим редактирования прототипа: окно свойств с настройками анимации и целевого действия для перехода на фрейм

Приведём рекомендации по работе в Figma:

- Макеты отдельных страниц, окон и экранов продукта должны быть представлены отдельными фреймами.
- Повторяющиеся паттерны должны быть представлены компонентами.
- Любой компонент в дизайн-системе должен ложиться в сетку.
- Объекты, из которых состоит макет, должны быть сгруппированы, группам должны быть заданы понятные имена.
- Нежелательно, чтобы в компонентах был какой-либо осмысленный контент. Их нужно делать максимально обезличенными, подходящими под все случаи использования. Плохо, когда в компоненте кнопки написано «Сохранить» и хорошо, когда «Кнопка».
- Не следует использовать компонент в качестве экземпляра в макете, его нужно выносить отдельно.
- Не следует делать из компонентов всё, поскольку это увеличит время на создание макета и внесение изменений.
- В первую очередь нужно настраивать стили текста и цветов. Не нужно делать компоненты для хранения цветов.
- Используйте ограничители (constraints) для привязки расположения блоков внутри компонента.
- Оценивайте компонент учитывая контекст его использования, т.е. оценивайте экземпляры компонента в макете.
- Если компонент должен содержать произвольный текст, создайте его экземпляр с наиболее длинной строкой чтобы оценить компоновку текста.

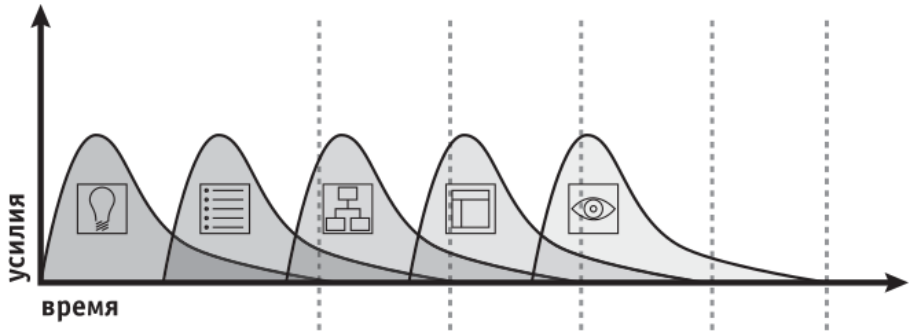


Рис. 4.37. Каждый следующий этап проектирования должен начинаться до полного завершения предыдущего.

4.3 Методология проектирования

Программы или сайты, решающие довольно простые задачи, можно создать пропуская многие этапы проектирования. Время, необходимое на исправление или изменение интерфейса таких продуктов может быть сильно меньше, чем время затраченное на проектирование интерфейса. Но для относительно крупных продуктов своевременное проектирование (определение перечня решаемых задач, информационной архитектуры и пользовательского интерфейса) позволяет избежать ошибок в дальнейшем.

Проектирование сверху вниз не обязательно должно иметь чётко очерченные этапы, где каждый следующий этап зависит только от предыдущих. Возможна корректировка решений, принятых на предыдущих этапах. Поэтому стоит начинать работать над новым этапом проектирования, когда предыдущий ещё полностью не завершился (рис. 4.37).

Например, создавая макет страницы (уровень компоновки) с описанием направлений подготовки (специальностей) вуза, разработчик может выяснить, что направлений слишком много. Пользователям будет проще ориентироваться если создать отдельные страницы факультетов или кафедр. Значит, нужно изменить диаграмму, описывающую информационную архитектуру.

Методология проектирования преложенная Джесси Гаретом – не единственно верная. Но почти у всех методологий проектирование

общая идея – проектирования идёт сверху вниз, от проектирования наиболее абстрактных аспектов (например, набора прецедентов) к наиболее конкретным (отдельным элементам пользовательского интерфейса). Более детальная схема (рис. 4.38) всех этапов проектирования дана, например, Аланом Купером в книге Основы проектирования взаимодействия [18].

Вопросы

1. Что такое UX? Что входит в это понятие?
2. Назовите этапы проектирования (по Джессу Гаретту), опишите каждый этап.
3. Зачем нужно моделировать потенциальных пользователей?
4. Из чего должно состоять описание персонажа?
5. На основе какой информации описывается персонаж?
6. Что такое диаграмма прецедентов? Зачем она нужна?
7. Какие бывают модели пользователя?
8. Из каких элементов строится диаграмма прецедентов?
9. Опишите виды отношений между прецедентами и ролями.
10. Как описать информационную архитектуру?
11. Зачем создавать низко-детализированные макеты пользовательского интерфейса?
12. Какие средства существуют для создания макетов?
13. Что что такое компонент (Figma)? Как его создать? Зачем их использовать?
14. Как влияет изменение свойств компонента на его экземпляры? Что если в экземпляры отличаются от компонента текстом, размерами, цветом?

15. В чём состоит преимущество использование сетки для выравнивания элементов интерфейса?
16. Как задать сетку для фрейма в Figma?
17. Как создавать стили для цветов, текста и эффектов в Figma? Как посмотреть перечень стилей?
18. Как должно распределятся время между этапами проектирования опыта взаимодействия?

Литература

- Алан Купер, Рейманн Роберт М., Основы проектирования взаимодействия. – Пер. с англ. – СПб.: Символ-Плюс, 2018, 720 с.
- Гарретт Дж. Веб-дизайн: Элементы опыта взаимодействия». – Пер. с англ. – СПб.: Символ-Плюс, 2008. – 192 с.

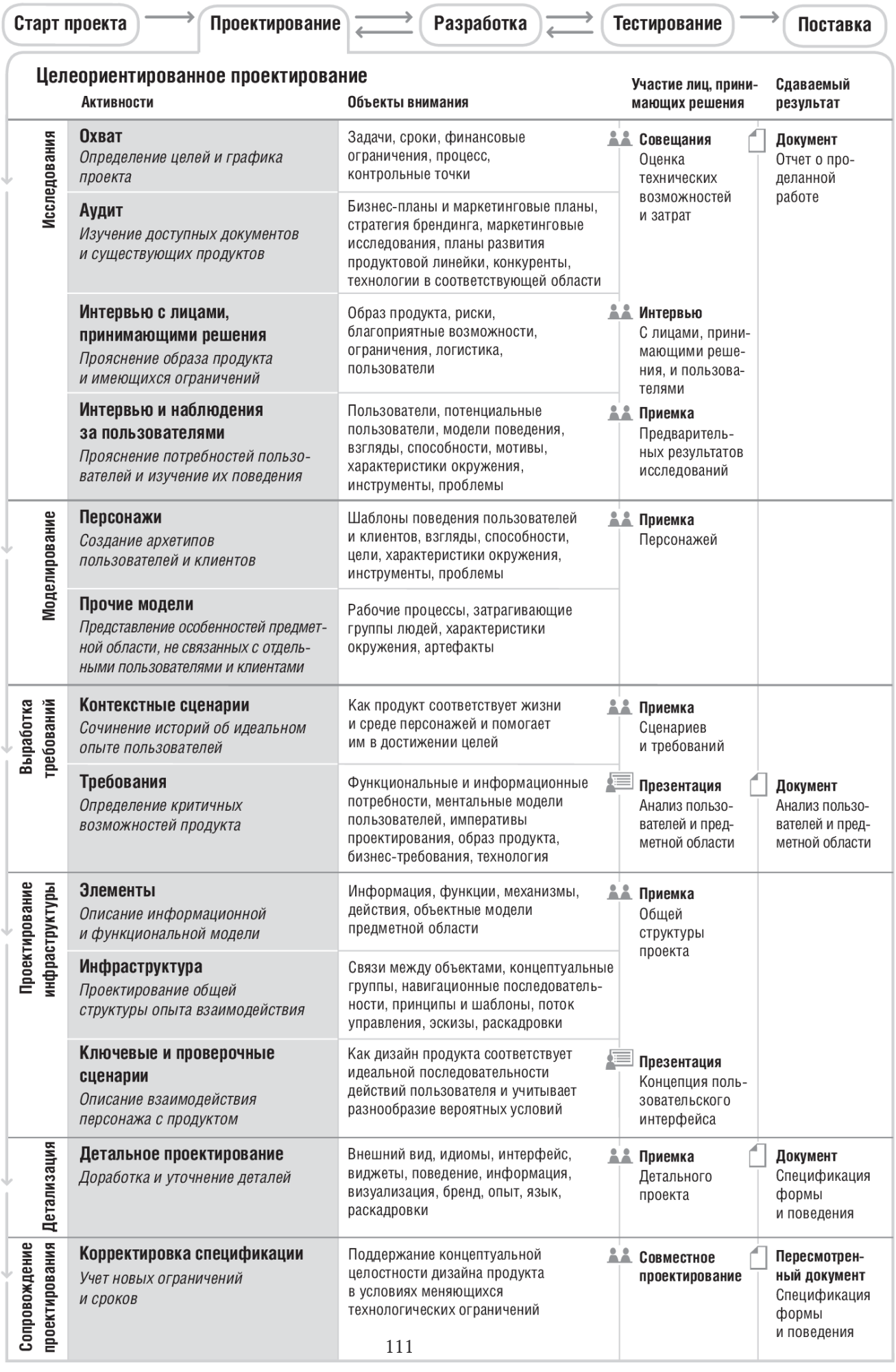


Рис. 4.38. Этапы проектирования продукта [18]

5 Юзабилити и тестирование

5.1 Юзабилити

Юзабилити (usability, удобство использования, эргономичность) — способность продукта быть понимаемым, изучаемым, используемым и привлекательным для пользователя в заданных условиях.

Существует большое число критериев, которые помогают оценить юзабилити продукта. Приведём показатели Шнейдермана [6], согласно которым интерфейс характеризуются:

- скоростью работы пользователя,
- количеством человеческих ошибок,
- субъективной удовлетворенностью,
- скоростью обучения навыкам оперирования интерфейсом,
- степенью сохраняемости этих навыков при неиспользовании продукта.

В предыдущей главе, были описаны этапы проектирования опыта взаимодействия. Первые два – изучение пользователей и описание возможностей продукта помогают сделать продукт привлекательным для пользователя. Качественное исполнение остальных аспектов юзабилити требует многостороннего подхода, который трудно формализовать и обосновать теоретически.

Рассмотрим 10 правил (эвристик[1]) Якоба Нильсона, помогающих добиться хорошего юзабилити.

[1]Эвристика – формально не обоснованное, но работающее на практике правило

1. ***Видимость статуса системы***. Пользователь должен всегда знать, что происходит, получая подходящую обратную связь в приемлемое время.

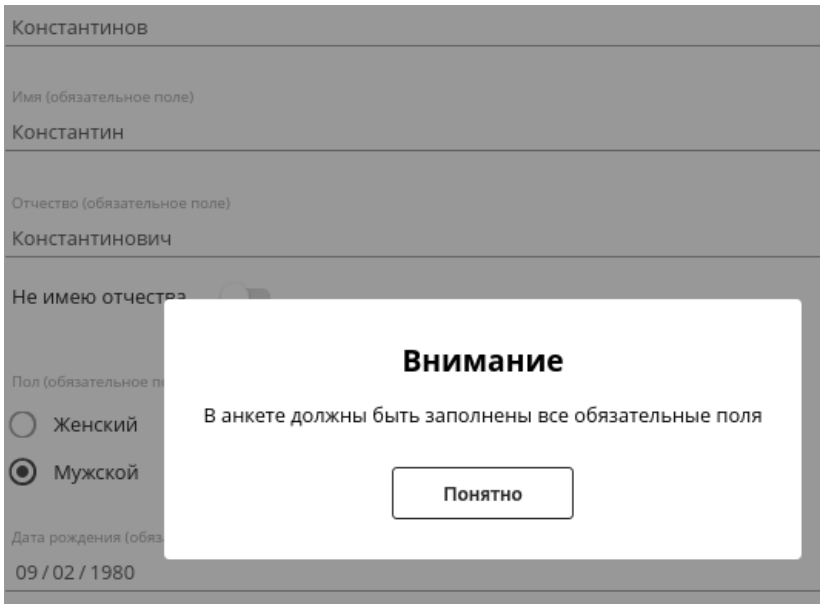


Рис. 5.1. Плохая видимость статуса системы. Анкета на сайте leadersofdigital.ru: обратная связь есть, но не полная. Непонятно какие поля не заполнены. Решение: вместо диалогового окна с сообщением, выделить (например, цветом) незаполненные поля, возможно добавить к каждому полю комментарий поясняющий формат (например: введите дату в формате: ДД / ММ / ГГГГ)

Антипримером может служить форма анкеты (рис. 5.1): обратная связь есть, но не полная. Непонятно какие поля не заполнены. Локус внимания, смещается не на путь решения проблемы, а на малополезное сообщение. Такое модальное окно, как обратная связь, само по себе требует обратной связи. Решение: вместо диалогового окна с сообщением, выделить (например, цветом) незаполненные поля, возможно добавить к каждому полю комментарий поясняющий формат (например: введите дату в формате: ДД / ММ / ГГГГ).

Пример сообщения на рисунке 5.2 не требует от пользователя дополнительной реакции на сообщение: клика на кнопку в модальном окне с сообщением об ошибке. При этом состояние формы хорошо заметно и точно указывает на проблему. При этом сообщение о статусе не должно быть навязчивым, обвинять пользователя в совершении ошибок, даже если они произошли (рис. 5.3). Сообщения

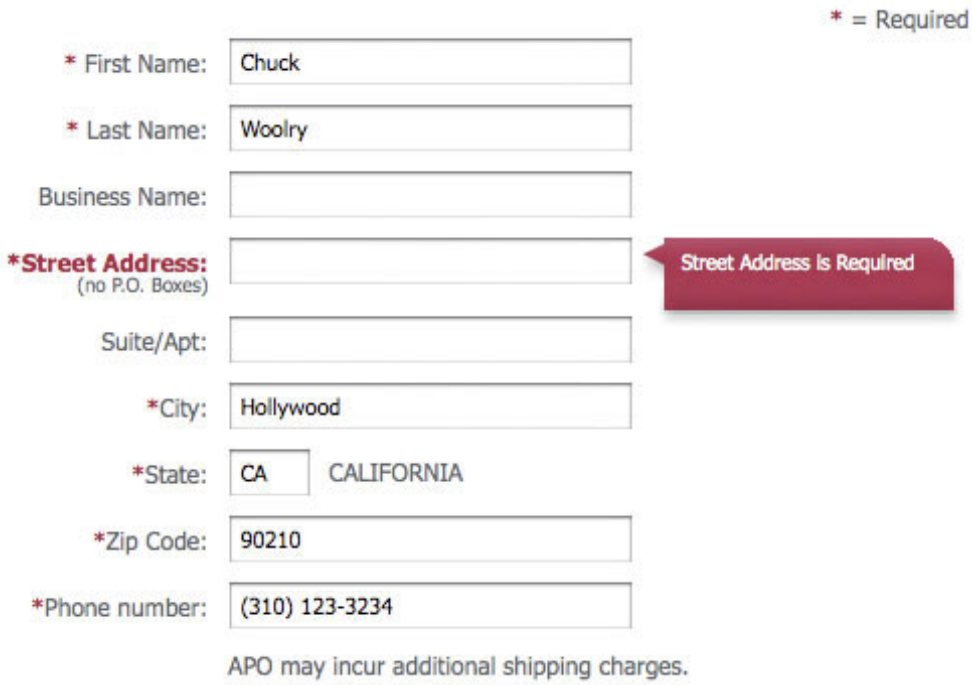


Рис. 5.2. Пример хорошего представления статуса системы. Название поля выделено. В подсказке справа, при необходимости, можно уточнить, что именно не так ввёл пользователь. На такие подсказки не нужно лишний раз кликать, в отличие от диалоговых окон. Они точно указывают на ошибку.

о исключительной ситуации, ошибке может быть написано в неформальной форме (рис. 5.4). В дополнении ко всему, форма объясняет зачем нужно вводить данные (“If we have order questions“). Однако для комфортного восприятия стоит увеличить размер шрифта.

Приведём отдельные рекомендации по отображению статуса системы:

- Если отклик программы более 200 мс покажите индикатор загрузки,смените форму курсора.
- Если программа выполняет действие дольше нескольких секунд, то покажите полосу прогресса.
- Если в программе предусмотрены режимы, то явно обозначьте текущий режим.
- Пример режимов: режим вставки или перезаписи в текстовом редакторе (вертикальный или горизонтальный курсов); режим кисти и резинки в графическом редакторе.

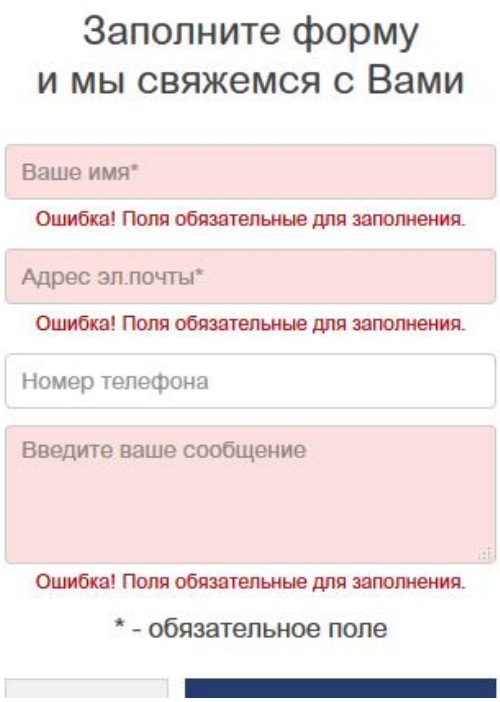


Рис. 5.3. Пример отображения статуса незаполненной формы. Плюсы: отмечены все незаполненные поля. Минусы: форма “кричит“ на пользователя: Ошибка! Ошибка! Ошибка! Решение: тактично сообщить об ошибке, например "Пожалуйста заполните поле"; выбрать менее насыщенный цвет для фона всех полей.

2. ***Соответствие между системой и реальным миром***. Система должна «говорить на языке пользователя», используя понятную ему терминологию и концепции.

- Не используйте специальные термины из IT: авторизация (вход), транзакция (операция, запрос), лейбл (надпись), …
- Старайтесь не использовать специальные термины, из предметной области без необходимости. Например, в мобильном приложении банка для широкого круга пользователей: транзакция, дебиторы, кредиторы, контрагенты, …

3. ***Управляемость и свобода для пользователя.*** Пользователи неизбежно ошибаются. Нужно дать понятную возможность исправить ошибку (рис. 5.5), например, функции отмены (undo) и повтора (redo).

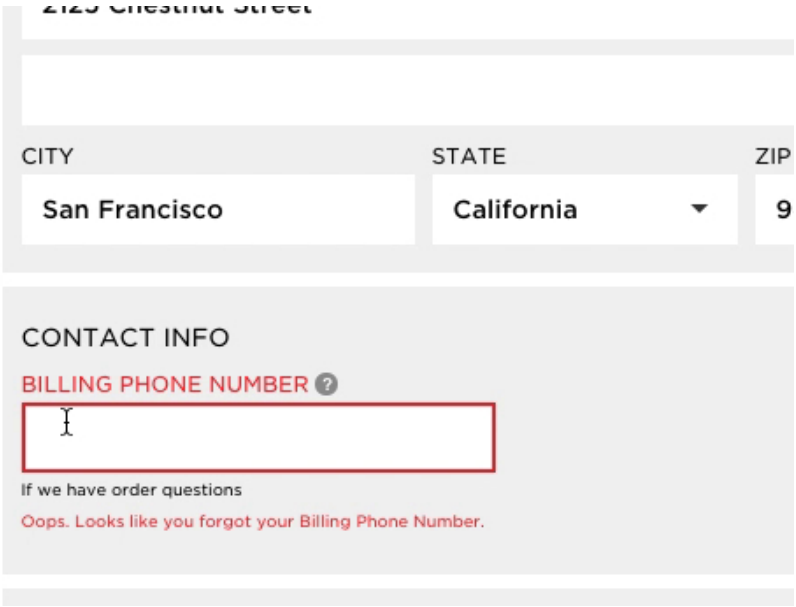


Рис. 5.4. Хороший пример отображения статуса незаполненной формы. Незаполненное поле выделено. Подсказка дана снизу. Нужно немного увеличить размер текста подсказки.

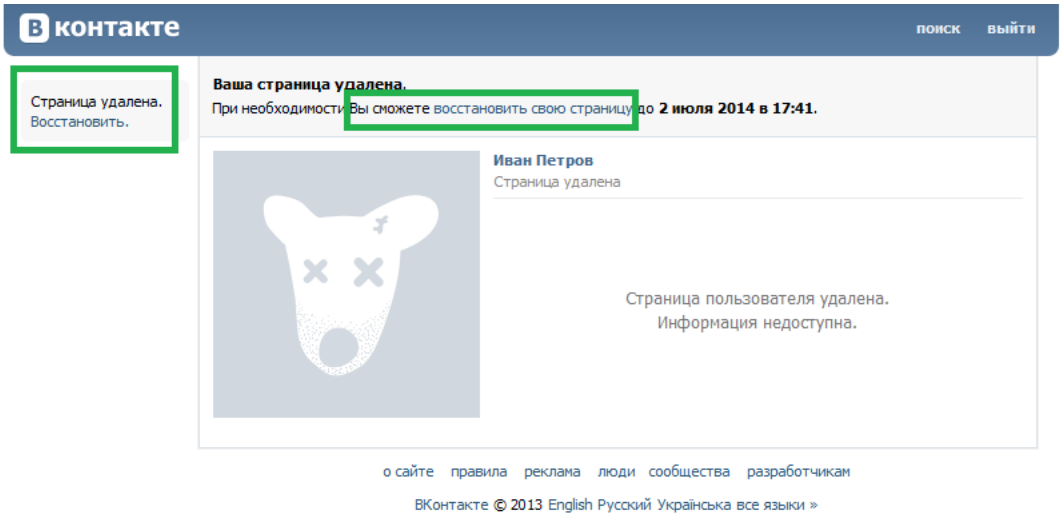


Рис. 5.5. Вид персональной страницы после запроса на удаления аккаунта

4. ***Согласованность и стандарты.*** Пользователи не должны гадать, значат ли одно и то же разные слова, ситуации или операции. Также нужно следовать соглашениям, принятым для данной платформы.

- Используйте одни и те же слова и обозначения для одних и тех же действий.

- Следуйте дизайн-системе[2]
- Сделайте интерфейс предсказуемым.

Это касается одинаковых обозначений одних и тех же действий и элементов как внутри среди разных окон одной программы, так и среди разных версий программы для разных операционных систем (на сколько это позволяют сами ОС).

5. ***Предотвращение ошибок.*** Продуманный дизайн, который не позволяет какой-то проблеме даже возникнуть, лучше, чем самые хорошие сообщения об ошибках. Следует устранять сами условия возникновения ошибок, либо выявлять их и предупреждать пользователя о предстоящей проблеме.

Дональд Норман предлагает делить всевозможные неточности в действиях пользователя на две категории: промахи и ошибки. Промах возникает в ситуациях когда пользователь понимает как нужно взаимодействовать с программой, но ошибается из-за недостатка внимания (рис. 5.6), например вводит в поле ввода логина свой пароль или не обращает внимание на раскладку клавиатуры [3]. Другой источник промахов – неточные движения мыши и случайные нажатия клавиш. Например пользователь случайно нажимает на кнопку “Ок“, вместо расположенной рядом[4] кнопки “Отмена“ или нажимает на кнопку О вместо кнопки 0 на клавиатуре.

Уменьшить число промахов можно минимизируя вероятность их возникновения, проводя пользователя только через безопасные области. Ограничения помогают пользователю задать неправильное значение, например, в числовое поле ввода нельзя ввести буквы. Предложение наиболее распространенных вариантов, облегчает выбор для пользователей, например, при поиске. Диалоги подтверждения помогают уменьшить вероятность деструктивных действий, например удаления файла.

---

[2]Дизайн-система — набор компонентов, правил, предписаний по дизайну
[3]см. также параграф 2.4 о режимах
[4]см. параграф о 6.2 прицеливании и законе Фиттса

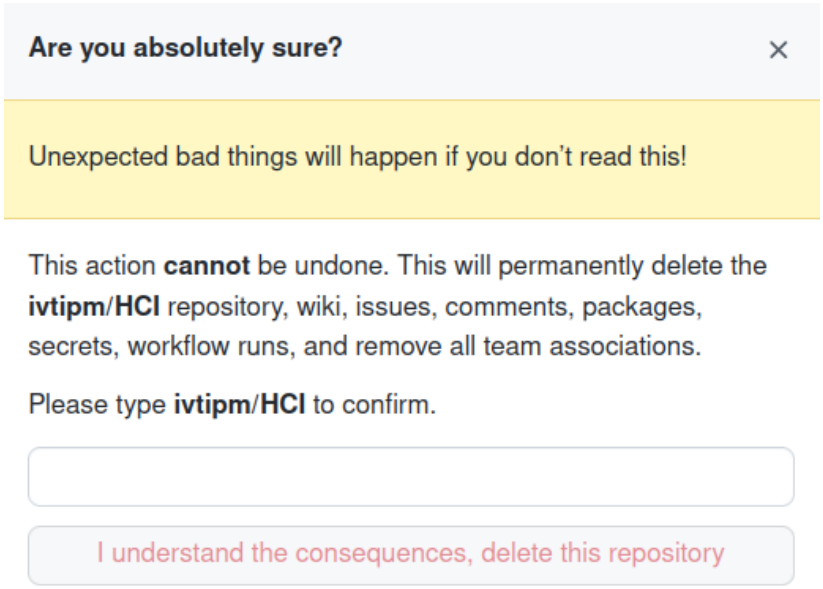


Рис. 5.6. Окно подтверждения опасного действия на GitHub. Нужно ввести название репозитория перед удалением. Это даёт пользователю возможность сделать имя репозитория локусом внимания и шанс остановится перед удалением не того репозитория.

Ошибки же часто вызваны неверной ментальной моделью в представлении пользователя о том, как работает система. В таких ситуациях пользователь неправильно понимает смысл коммуникации и сознательно выполняет действие, которое приводит не к тому результату, на который он рассчитывал. Такие ошибки зачастую не так просто исправить, и их следует выявлять на этапе пользовательского тестирования. Для предотвращения ошибок нужно делать модель представления как можно более ясной и понятной большинству пользователей.

6. ***Распознавать лучше, чем вспоминать.*** Минимизируйте нагрузку на память пользователя, явно показывая ему объекты, действия и варианты выбора. Например на рисунке 5.7) значки в нижней части программы не подписаны. Пользователю придётся так или иначе вспоминать их назначение. Пользователь не должен в одной части диалога запоминать информацию, которая потребуется ему в другой. Антипример приведён на рисунке 1.7. Инструкции по использованию системы должны быть видимы или легко доступны везде, где возможно.

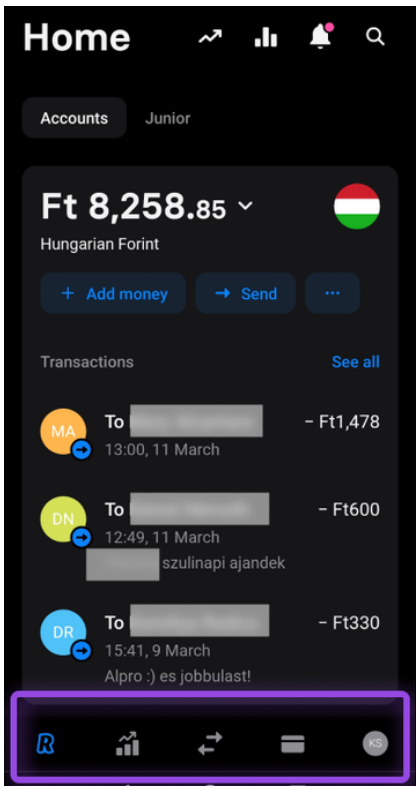


Рис. 5.7. Значки в нижней части программы не подписаны: пользователю придётся так или иначе вспоминать их назначение. Подписи решили бы эту проблему.

7. ***Гибкость и эффективность использования.*** Программа должна быть понятной для новичков и эффективной для профессионалов. Акселераторы (средства быстрого выполнения команд, например горячие клавиши), которые новичок даже не видит, для опытного пользователя часто могут ускорить взаимодействие. Поэтому система должна удовлетворять как неопытных, так и опытных пользователей. Следует давать возможность настраивать под себя часто используемые операции. Про анализ времени, которое пользователь тратит на взаимодействие с программой говорится в параграфе 6.4 о методе GOMS.

8. ***Эстетический и минималистический дизайн.*** В интерфейсе не должно быть информации, которая не нужна пользователю или которая может понадобиться ему в редких случаях. Каждый избыточный элемент диалога отнимает внимание от нужных элементов, затрудняет визуальный поиск и осмысление отображаемой информации. При некоторые способы упрощения интерфейса приведено

в параграфе 6.5.2. Эстетичность же повышает субъективную удовлетворённость пользователем от взаимодействия с продуктом.

9. ***Помочь пользователю понять и исправить ошибку.*** Сообщения об ошибках следует писать простым языком, без кодов (см. антипример на рис. 1.4), чётко формулируя проблему и предлагая конструктивное решение (рис. 7.28).

10. ***Справка и документация.*** Хотя было бы лучше, если бы система была пригодна к использованию без документации, всё же необходимо предоставлять справку и документацию. Информация должна быть простой в поиске, соответствовать задаче пользователя, описывать конкретную последовательность действий, и не должна быть слишком большой.

Про написание интерфейсных текстов, в частности сообщений об ошибках говорится в параграфе 7.4.

5.2 Юзабилити-тестирование

5.2.1 Понятие юзабилити-тестирования

Юзабилити-тестирование, (usability testing) — исследование, выполняемое с целью определения, удобен ли некоторый искусственный объект (такой, как веб-страница, пользовательский интерфейс или устройство) для его предполагаемого применения.

Это тестирование продукта, но не исследование целевой аудитории и не тестирование пользователей. Так как продукт к этому времени должен быть завершён хотя бы частично, а значит целевая аудитория уже должна быть изучена, известны её потребности. В результате проведённого тестирования дорабатывается пользовательский интерфейс продукта, но не изменяется целевая аудитория.

Юзабилити-тестирование должно проводится при участии пользователей, а не вовлечённых в разработку людей.

У пользователей, в отличии от разработчиков, другой набор ментальных моделей, они не осведомлены о внутреннем устройстве продукта.

Провести тестирование можно большим набором способов. Начиная от неформального общения с непосредственными пользователями, до строгих методов юзабилити-тестирования с наблюдением за пользователем, решающим конкретную задачу. В последнем случае действия пользователя могут быть записаны, а потом проанализированы с составлением журнала действий и измерением времени этих действий.

Неформальные методы юзабилити-тестирования проще организовывать, можно проводить относительно часто. С другой стороны, разработчик продукта обычно осознанно или нет вмешивается в действия пользователя, что искажает результаты.

5.2.2 A/B-тестирование

Идея A/B-тестирования [5] заключается в следующем:

- Пользователей разделяют на две (или более) группы.
- Каждой из групп показывай свой вариант (A или B) интерфейса.
- Варианты интерфейса различаются чаще всего в деталях, например расположением отдельных элементов, цветами и т.п.
- Пользователи часто не подозревают об участии в тестировании и о существовании разных вариантов интерфейса.
- После сбора достаточной большой статистики сравниваются показатели (например, конверсия[6], время проведённое на сайте и др.).

Как правило варианты интерфейса в A/B тестировании имеют небольшие отличия. Например на втором варианте страницы предвари-

[5] Термин изначально появился в маркетинге

[6] Конверсия — это отношение (в процентах) числа посетителей сайта, выполнивших на нём какие-либо целевые действия, к общему числу посетителей сайта

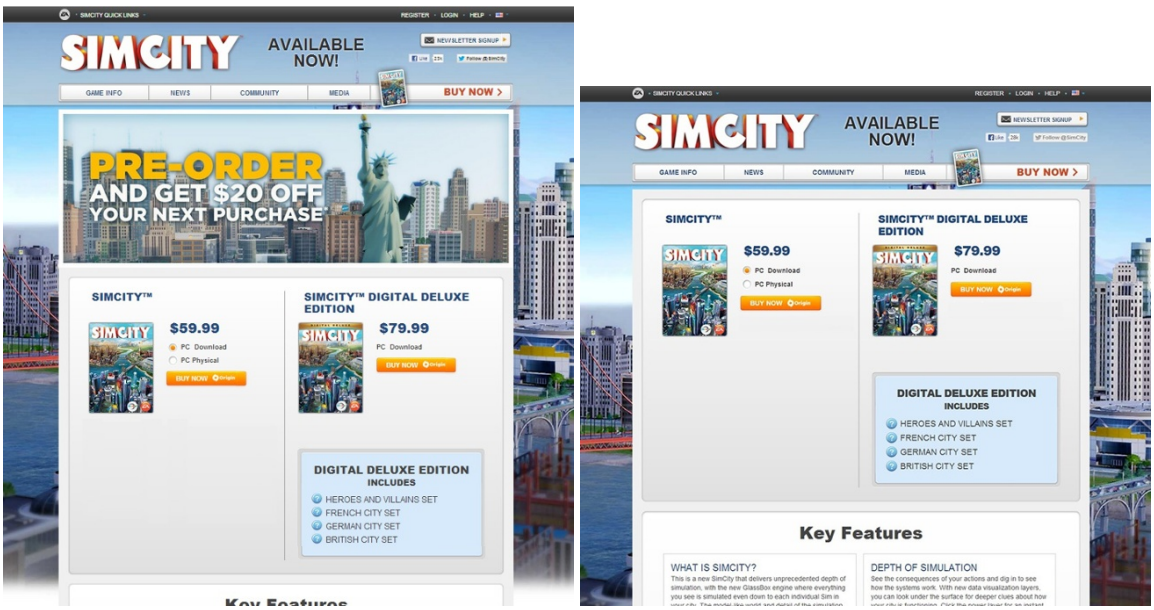


Рис. 5.8. A/B тестирование: в течение некоторого времени для разных пользователей веб-страница показывалась в одном из двух вариантов; результат эксперимента: пользователи, которым показывали второй вариант страницы, чаще делали заказ.

тельного заказа игры SimCity 4 (рис. 5.8) отсутствовал рекламный баннер. В результате эксперимента выяснилось, что пользователи, которым показывали второй вариант страницы, чаще делали заказ. Даже незначительные, на первый взгляд, изменения интерфейса могут требовать проверки. Например на втором варианте форме регистрации на сайте букмекерской фирмы (рис. 5.9, справа) добавлено: “Мы даём 100% гарантию безопасности ваших данных. Они никому не будут переданы.“ Результат эксперимента: увеличение количества регистраций на 20% по сравнению с оригинальной формой. При объёме выборки в 20257 пользователей и 380 регистраций.

5.2.3 Тестирование с наблюдением

Перед тестированием составляется список задач, для которых создавался продукт и которые предлагаются пользователям для решения. Процесс тестирования фиксируется в протоколе тестирования. Где отмечается трудности, с которыми столкнулся пользователь: непонимание инструкций, ответов системы и т.д. Протокол тестирования может включать видеозапись экрана.

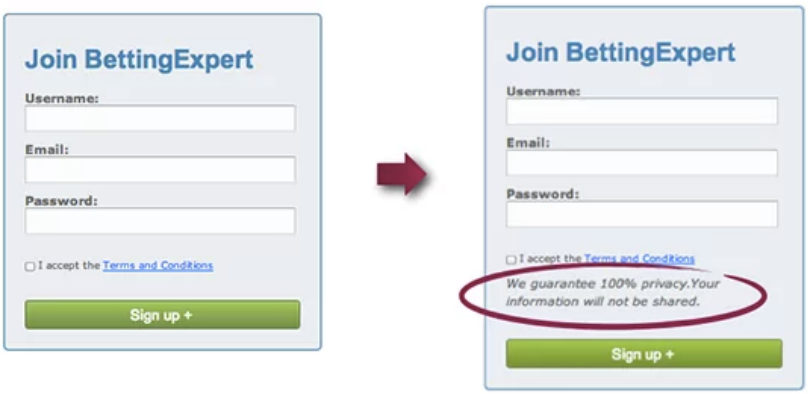


Рис. 5.9. Два варианта формы регистрации на сайте букмекерской компании. На форме справа добавлено: Мы даём 100% гарантию безопасности ваших данных. Они никому не будут переданы. Результат эксперимента: увеличение количества регистраций на 20% по сравнению с оригинальной формой. Объём выборки: 20257 пользователей, 380 регистраций.

Вариация метода – оценка мест, где пользователь при решении задачи отклонился от идеального сценария, описанного разработчиками продукта.

Сначала стоит тестировать самые частотные сценарии, т.е. предлагать пользователю решить самые распространённые задачи. Лучше всего заранее составить список задач, которые продукт помогает решить, и ранжировать их по убыванию частоты использования. Другой критерий ранжирования, который можно использовать совместно с частотностью, – важность или стоимость ошибки.

Наблюдение за тем, как пользователи взаимодействуют с продуктом, нередко помогает дизайнеру найти более оптимальные решения. Если при тестировании используется модератор, то его задача — держать респондента сфокусированным на задачах (но при этом не «помогать» ему решать эти задачи).

В качестве пользователей обычно выбираются люди, соответствующие разработанным ранее пользовательским моделям, то есть персонажам.

Коридорное тестирование. На практике далеко не всегда есть возможность пригласить потенциальных пользователей для тестиро-

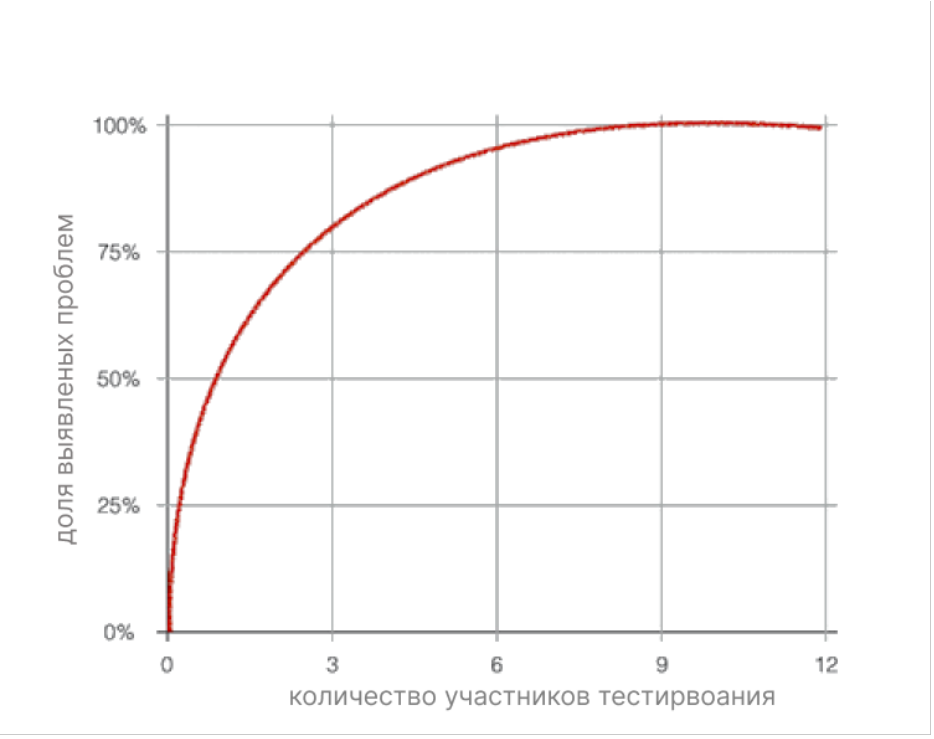


Рис. 5.10. Небольшое количество участников юзабилити тестирования позволяют выявить значительную часть проблем [44].

вания. Тем более, если это не сотрудники заказчика, для которых продукт и разрабатывается, и которые могут быть заинтересованы в тестировании. При этом важно начать проводить тестирование как можно раньше. В этом случае прибегает к так называемому коридорному тестированию. Когда к тестированию привлекается первый попавшийся человек “из коридора.“ Свежий взгляд любого человека на интерфейс лучше, чем взгляд дизайнера потому, что он лишён проклятья знания. Такое тестирование, как правило, не занимает много времени, его можно делать часто.

Стив Круг рекомендует привлекать для тестирования до трёх человек [17]. Они позволят выявить как минимум все значительные проблемы (рис. 5.10). При этом важно провести больше циклов тестирования, чем собрать максимум информации из каждого цикла. С небольшим количеством участников организовывать тестирование проще.

5.2.4 Метод карточной сортировки

Этот узконаправленный метод тестирования позволяет выявить то, как пользователь относит понятия к разным категориям. Создаётся список параметров или подкатегорий, которые пользователь должен классифицировать. Каждый параметр записывается на отдельной карточке. Пользователи группируют карточки наиболее логичным, по их мнению, образом давая группам названия. Наиболее часто используемый способ группировки реализуется в интерфейсе.

Метод обратной карточной сортировки переворачивает задачу. Пользователи по уже созданным группам ищут конкретные карточки. Смотрите также параграф 4.2.6 о разработке навигации.

Вопросы

1. Что такое юзабилити?
2. Что такое UX?
3. Как соотносятся эти понятия?
4. Какими характеристиками можно описать юзабилити?
5. Перечислите и охарактеризуйте эвристики Якоба Нильсона. Приведите примеры и антипримеры.
6. Зачем нужно проводить юзабилити-тестирование?
7. Что такое проклятье знания?
8. Что такое A/B тестирование?
9. Как можно измерять разницу между вариантами в A/B тестировании?
10. Что такое коридорное тестирование? В чем его приемущество?
11. Сколько стоит привлекать людей для проведения тестирования?
12. Опишите метод карточной сортировки и метод обратной карточной сортировки.

Литература

- Якоб Нильсен, Хоа Лоранжер. Web-дизайн: удобство использования Web-сайтов = Prioritizing Web Usability. — М.: «Вильямс», 2007. — 368 с.
- Унгер, Р. UX-дизайн. Практическое руководство по проектированию опыта взаимодействия / Р. Унгер, К. Чендлер. — Символ-Плюс, 2011. — 336 с.

6 Анализ интерфейса

6.1 Количественная оценка пользовательского интерфейса

Эффективность – один из внешних критериев качества ПО. Можно измерить требуемый объём оперативной памяти и количество процессорного времени при решении программой конкретных задач.

Однако не существует чётких метрик оценки эффективности пользовательского интерфейса. Но существуют модели, которые позволяют оценить отдельные аспекты ПИ с известными упрощениями.

“В сущности, все модели неправильны, но некоторые полезны“ – это высказывание Джорджа Бокса[1] применимо и к приведённым ниже моделям. Сами по себе они не отражают в точности отдельные аспекты поведения пользователей, но позволяют сделать полезные выводы о проектировании интерфейса или получить сравнительную оценку качества интерфейса.

6.2 Закон Фиттса

Закон Фиттса – математическая модель для определения времени указания на цель на экране. Среднее время, затрачиваемое на указание на цель определяется как

$$T = a + b \cdot log_2 \left(\frac{D}{W} + 1 \right), \tag{6.1}$$

[1] Джордж Бокс – статистик, внёсший заметный вклад в такие области, как контроль качества, планирование эксперимента, анализ временных рядов и Байесовский вывод.

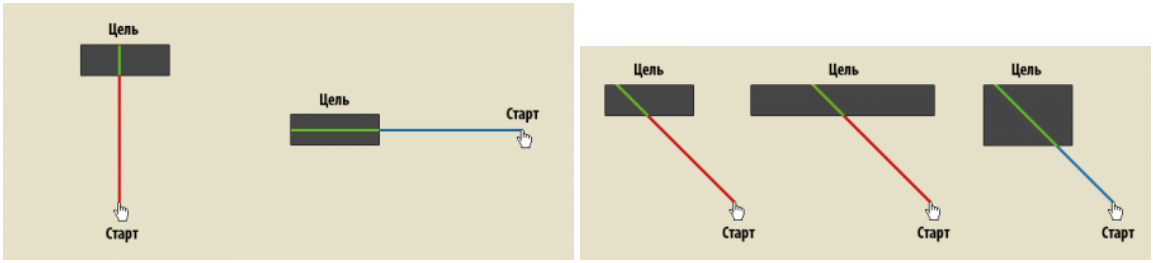


Рис. 6.1. Закон Фитса: время указания на цель зависит от дистанции до цели (красная и синяя линии) и размера цели вдоль оси движения (зелёная линия). Увеличение ширины цели не всегда даёт выигрыша во времени. Нужно учитывать либо все направления движений, либо самые типичные [66].

где a — среднее время запуска/остановки движения, b — величина, зависящая от типичной скорости движения указателя, D — дистанция от точки старта до центра цели, W — ширина цели, измеренная вдоль оси движения (рис. 6.1). Стоит учитывать, что увеличение ширины цели не всегда даёт выигрыша во времени, так как это может не изменить параметр W. Нужно учитывать либо все направления движений, либо самые типичные [66].

Значение коэффициентов a и b можно оценить экспериментально, измерив время наведения для разных целей, находящихся на разных расстояниях. Но с точки проектирования интерфейса, важны выводы, следующие из закона, а не конкретное количество времени, которое пользователь затратит на наведение на цель. Тем более что закон был описан Полом Фиттсом в 1954 году в эксперименте, где требовалось указывать стилусом на параллельные друг другу металлические полоски разной ширины (рис. 6.2).

Закон имеет допущения. Пользователь знает где находится указатель, знает где находится цель, на которую он должен указать. Пользователь не промахивается мимо цели. Направление движения курсора не влияет на время указания.

Логарифмическая зависимость позволяет сделать два вывода. Небольшое увеличение небольшой цели делает её простой (быстрой) для указания (наведения). Такое же небольшое увеличение большой цели не даст заметного выигрыша.

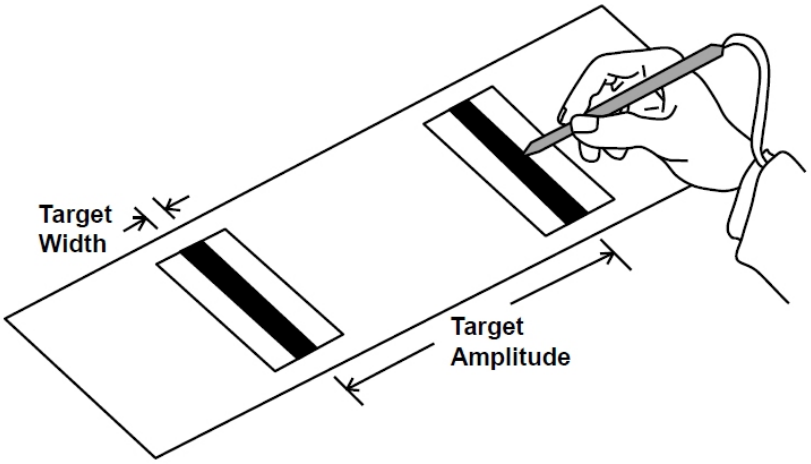


Рис. 6.2. Оригинальная постановка эксперимента, по результатам которого был описан закон Фиттса. Участник эксперимента должен был как можно быстрее касаться стилусом двух металлических пластин. Эксперимент повторялся для пластин разной ширины и с разным расстоянием между ними [48].

Делать все элементы интерфейса большими неразумно. Тем более, если элементы будут находиться слишком близко друг к другу, то пользователи будут чаще ошибаться указывая не на тот элемент, который нужно.

Правило размера цели. Нужно увеличивать размер только часто используемых целей. То есть тех кнопок, ссылок, элементов меню и т.д. на которые пользователь во время работы кликает чаще всего.

Правило бесконечной границы. Цели, расположенные у края экрана, имеют условно бесконечную высоту (ширину), так как двигаясь вниз до края курсор не сможет продвинуться дальше границы цели (рис. 6.3). В идеальном случае, как на рисунке, размер верхней цели бесконечен. Значит, логарифм в формуле 6.1 равен нулю. В остальных случаях время указания на цель, помимо расстояния, зависит либо от ширины либо высоты цели.

Цели, расположенные в углах экрана, имеют условно бесконечную ширину и высоту. Поэтому указание на них из любой точки экрана, согласно закону Фиттса, занимает минимально возможное количество времени.

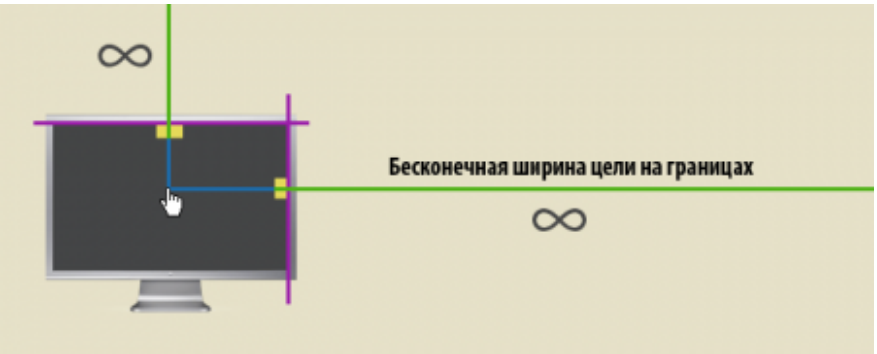


Рис. 6.3. Правило бесконечной границы. Так как нельзя промахнутся мимо края экрана, ширина или высота цели в направлении границы экрана считается бесконечной.



Рис. 6.4. Правило размера цели. Панель задач Windows 7 (сверху), Windows 10 (снизу). Кнопка "Пуск"и кнопка "свернуть все окна"(справа на панели задач) находится в углах экрана, что сокращает время наведения на них до минимально возможного. Круглая кнопка Пуск имеет квадратную область клика.

Правило бесконечной границы работает для панели задач операционной системы Windows (рис. 6.4). Кнопки закрытия окна расположены в правом углу экрана, когда окно развёрнуто. Вкладки в большинстве браузеров находятся прямо на заголовке окна, то есть возле верхнего края экрана, когда окно развёрнуто. Полоса прокрутки во многих приложениях расположена в правой части окна, что даже даёт преимущество, когда окно развёрнуто.

С точки зрения закона Фиттса выгодно располагать кнопки у края окна (если оно развёрнуто во весь экран). На рисунке (6.5) кнопки на левой панели инструментов имеют бесконечную ширину (если область нажатия доходит до левой границы окна).

В семействе операционных систем Mac OS в верхней части экрана традиционно располагается панель меню запущенной программы (рис. 6.6). Панель меню запущенной программы всегда находится в верхней части экрана. При наведении на панель программ в нижней

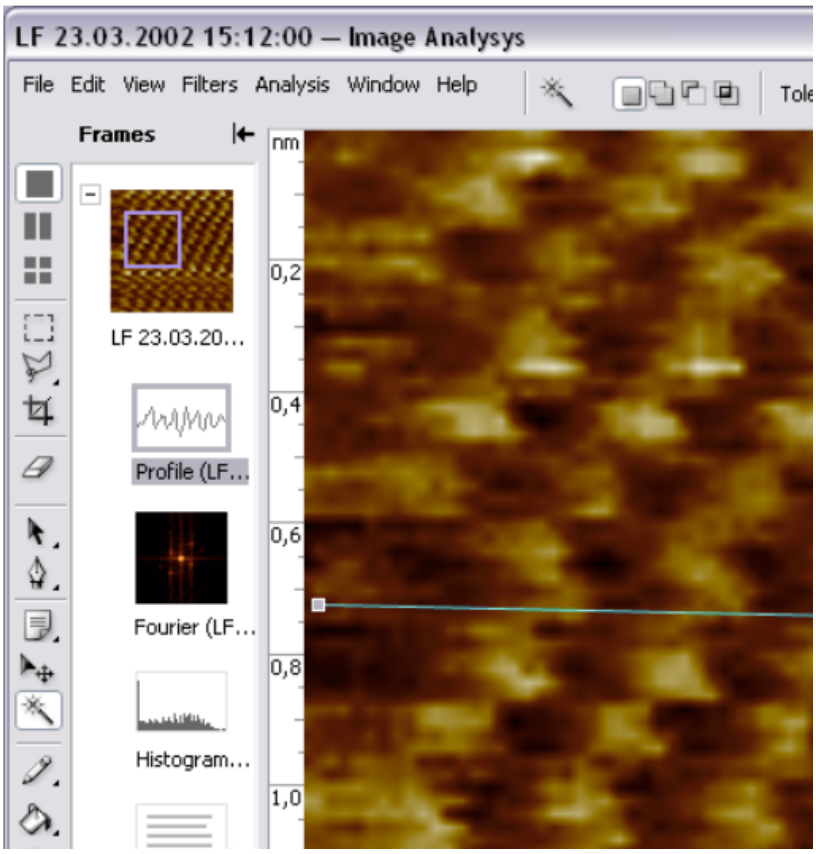


Рис. 6.5. Правило бесконечной границы: когда окно развёрнуто во весь экран, кнопки на левой панели инструментов имеют бесконечную ширину (если область нажатия доходит до левой границы окна) [1].



Рис. 6.6. MAC OS Leopard. Панель меню запущенной программы всегда находится в верхней части экрана. При наведении на панель программ в нижней части экрана ближайшие к курсору кнопки увеличиваются, облегчая наведение.

части экрана ближайшие к курсору кнопки увеличиваются, облегчая наведение.

Элементы, расположенные у границ экрана, удобны ещё и тем, что при указании на них пользователю не приходится замедлять движение курсора, чтобы точнее прицелиться. Что, однако, уже не является следствием закона Фиттса.

Кроме того, часто используемые элементы логично делать большими ещё и потому, что они становятся заметнее. Уменьшится время *поиска* цели.

Минимизировать время указания можно и минимизируя расстояния до цели, в формуле закона Фиттса (рис. 6.1). Контекстное меню всегда открывается в месте клика, так что не приходится слишком далеко вести курсор для выбора пункта. Самые часто используемые элементы меню стоит помещать в самый верх, ближе к начальной

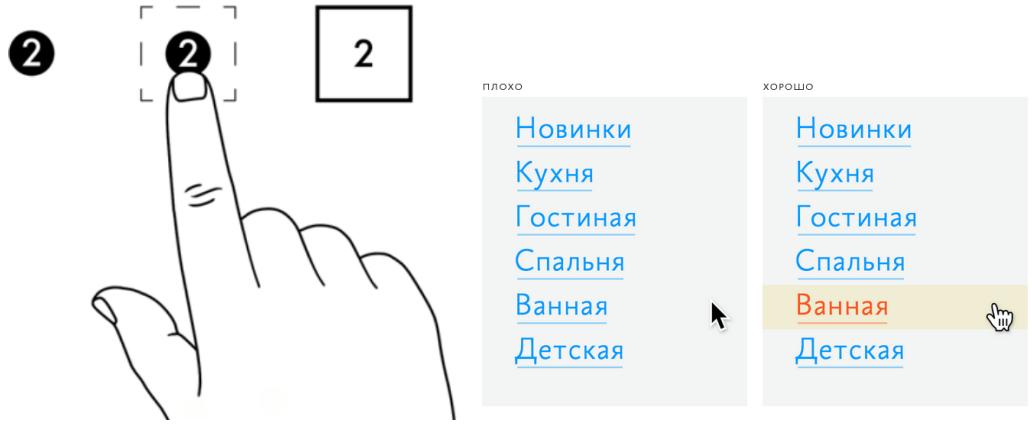


Рис. 6.7. Если объект нельзя увеличить в размерах, то можно попробовать увеличить невидимую область нажатия вокруг него. Примеры из bureau.ru/books/ui/demo/4

позиции курсора. В круговом контекстном меню (рис. 6.8) оптимизируется и расстояние до всех элементов их размер.

Расстояние до цели можно сократить почти до нуля. На рисунке 6.10, в приложении «Камера» кнопка для появляется в месте тапа.

Если элементы интерфейса могут быть выбраны последовательно, то, возможно, стоит расположить их ближе друг к другу (рис. 6.9). С другой стороны, делайте достаточное *расстояние* до кнопок, чтобы защитить пользователя от случайного нажатия.

Закон Фиттса подразумевает, что пользователь знает, где находится элемент интерфейса, на который он хочет навести курсор. Пользователь не знает, где искать элемент, то ко времени позиционирования добавляется время поиска этого элемента на экране. Это тоже нудно минимизировать.

Типичные элементы управления лучше располагать там, где пользователи ожидают их увидеть. Например, кнопка входа на сайт, как правило, располагается в верхней правой части сайта, меню “файл“ стоит в начале панели меню, а пункт “копировать“ – самый первый в контекстном меню для текста.

Скорее всего у пользователей сформированы привычки, для взаимодействия с типичными элементами интерфейса. Ко времени

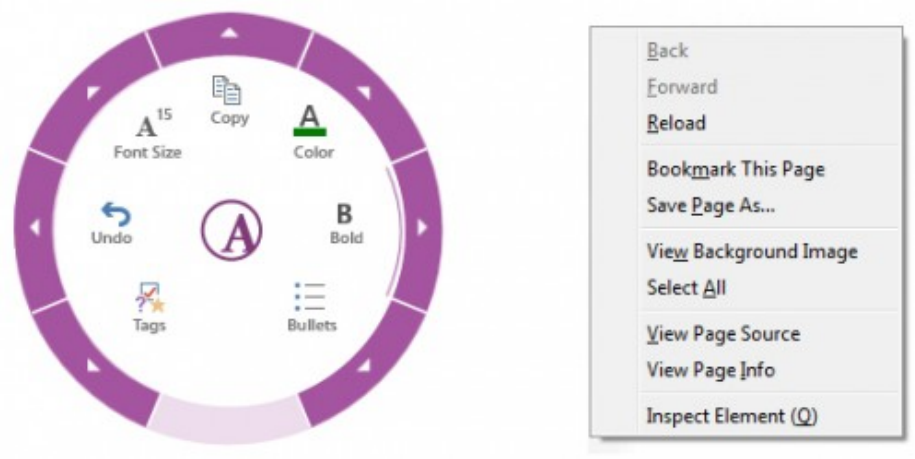


Рис. 6.8. Круговое контекстное меню. Время указание на все элементы, в отличие от классического контекстного меню, одинаково, а сами элементы больше. Классическое контекстное меню. Самые часто используемые пункты должны быть расположены ближе к указателю мыши (выше)

.

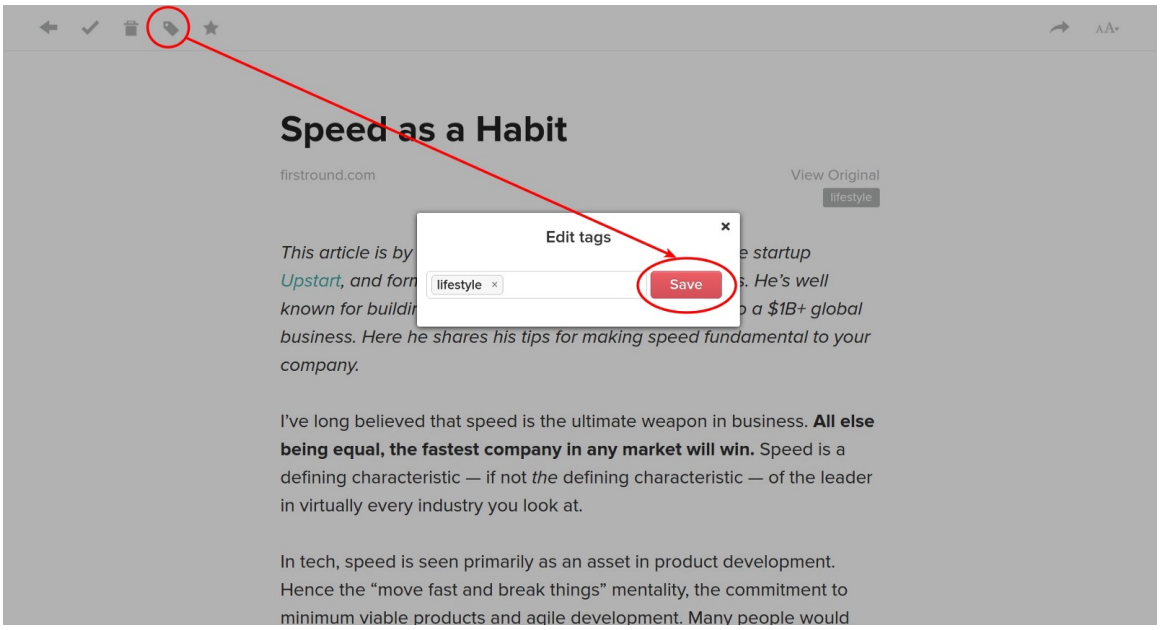


Рис. 6.9. Ожидается, что кнопки должны быть нажаты последовательно, однако они разнесены на большое расстояние.

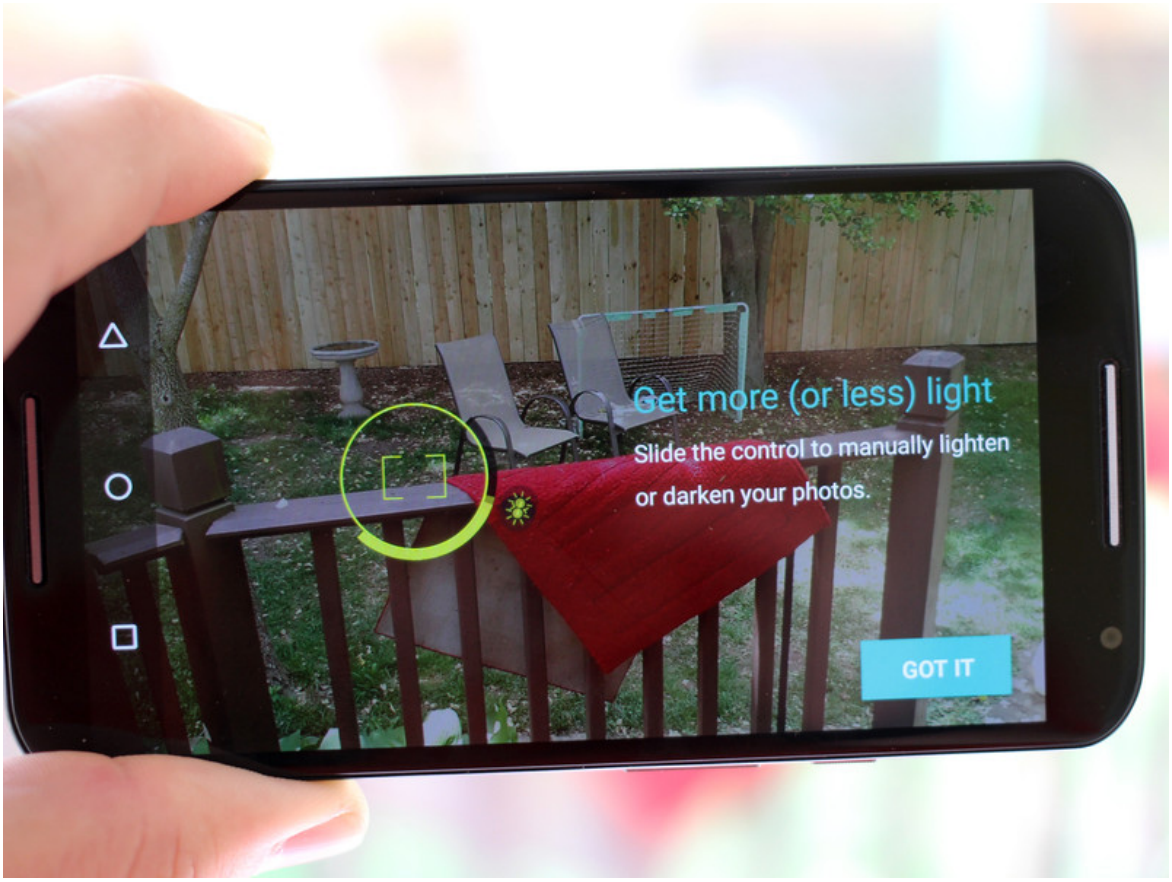


Рис. 6.10. Расстояние до цели равно нулю. Кнопка появляется в месте тапа.

наведения не будет добавлено время поиска элемента, а само взаимодействие с ним с высокой долей уверенности будет автоматичным.

6.3 Закон Хика и проблемы выбора

Время, затрачиваемое на выбор, является функцией числа альтернатив

$$T = a + b \cdot log_2 (n + 1), \tag{6.2}$$

где T — значение времени реакции, усреднённое по всем альтернативам; a, b – константы; n — число равновероятных альтернативных альтернатив; $+1$ — дополнительная альтернатива — случай отказа от выбора.

Под альтернативами можно понимать различные варианты действия пользователя. Например, выбор кнопки на панели инструментов, элемента меню или вкладки.

Коэффициенты a и b, используемые в выражении закона Хика, в большой степени зависят от многих условий, включая то, как представлены возможные варианты, и то, насколько хорошо пользователь знаком с системой. Наличие навыков и привычек в использовании системы снижает значение b.

Как и для закона Фиттса логарифмическая зависимость позволяет сделать вывод: немного уменьшая небольшое количество альтернатив мы существенно снижаем время выбора, такое же уменьшение большого числа альтернатив не даёт заметного выигрыша во времени.

Закон имеет допущения и ограничения. Пользователь знает обо всех доступных действиях, среди которых должен сделать выбор, а не узнаёт о них просматривая, например, контекстное меню. Закон не применяется, если выбор связан с принятием сложного решения, изучением каждого из вариантов и т.д. Например, выбор из 5

автомобилей для покупки и выбор из 5 супов в ресторане, скорее всего, займут разное количество времени. Закон Хика не описывает время *поиска* элемента на экране, среди множества других.

Для случая, когда вероятности выбора альтернатив существенно отличаются, используется следующая формула

$$T = a + b \sum_{i}^{n} p_i \log_2(1/p_i + 1), \tag{6.3}$$

где p_i – вероятность выбора i-й альтернативы.

Из формулировок закона следуют простые выводы. Сокращайте число альтернатив. Если это сделать невозможно, учитывайте то, что некоторые варианты могут выбираться чаще чем другие. Спрячьте часть непопулярных вариантов в подменю или подкатегории[2]. Например, если категорий товаров очень много и часть из них весьма специфические, то показывайте только самые популярные категории, добавьте кнопку "Показать все категории на сайте интернет-магазина. При такой организации выборы в большинстве случаев, пользователи смогут сделать быстрый выбор популярных вариантов среди небольшого числа и только изредка им придётся выбирать из большого числа вариантов или делать это в два этапа.

Делайте выбор за пользователя, если это возможно[3]. Например, предлагайте ему популярные товары или рекомендуйте их, основываясь на предпочтениях конкретного пользователя.

Сам процесс выбора из большого числа альтернатив может вообще отталкивать пользователей от совершения выбора (+1 в формуле 6.2). Тогда, сократите сложность принимаемого решения разбив его на несколько этапов. А если же пользователь не может отказаться от выбора, это требует от него лишних ментальных усилий. Что может ухудшить пользовательский опыт.

[2]о том, как придумать перечень категорий или разделов, например, на сайте, читайте в параграфе 5.2.4 про метод карточной сортировки

[3]см. также параграф 6.5.2 о информационной эффективности интерфейса

Ко времени непосредственно выбора может добавиться время, необходимое на изучение перечня альтернатив и время поиска элемента интерфейса, отвечающего за альтернативу на экране.

Время поиска элемента можно сократить, если, опять же, сперва показать самые вероятные альтернативы. Например, при регистрации на сайте нужно указать страну проживания или гражданство, то можно показать самые популярные варианты в начале списка. Можно сделать выбор за пользователя, основываясь на географической привязке его IP адреса. Если эти варианты всё же не сработают, то пользователь должен понимать принцип организации списка. Например, список стран будет отсортирован пол алфавиту. Тогда пользователь будет иметь примерное представление где искать нудный элемент списка.

6.4 GOMS

Для оценки времени, которое требуется пользователю, для решения задачи с помощью программы используется модель GOMS.

Модель GOMS (the model of Goals, Objects, Methods, and Selection rules, правила для целей, объектов, методов и выделения) позволяет предсказать время, необходимое для выполнения задачи с помощью конкретного интерфейса.

Весь процесс взаимодействия пользователя устройствами ввода разбивается на элементарные жесты.

Время, требующееся для выполнения какой-то задачи системой «пользователь – компьютер», является суммой всех временных интервалов, которые потребовались системе на выполнение последовательности элементарных жестов, составляющих данную задачу.

Для большей части сравнительного анализа задач, включающих использование клавиатуры и графического устройства ввода, вме-

сто проведения измерений для каждого отдельного пользователя можно применить набор стандартных интервалов.

Эта модель, как и законы Фиттса и Хика, даёт *сравнительную* оценку интерфейсов.

Элементарные жесты:

- ***K*** = 0.2 с. Нажатие клавиши.
- ***P*** = 1.1 с. Указание. Время, необходимое для того, чтобы указать на позицию на мониторе.
- ***H*** = 0.4 с. Перемещение руки с ГУВ[4] на клавиатуру или обратно.
- ***M*** = 1.35 с. Ментальная подготовка. Время необходимое чтобы умственно подготовиться к следующему шагу.
- ***R***. Ответ. Ожидание ответа компьютера[5].

Ограничения модели GOMS:

- Рассматривается средний пользователь.
- Пользователь знаком с интерфейсом.
- Не учитываются размеры и положение элементов интерфейса.
- На учитывается сложность принятия решения в ментальной операции.
- Характеристики устройств ввода не влияют на время жестов.

Вычисление времени, необходимого на выполнение действия, с помощью модели GOMS начинаются с перечисления операций из списка жестов модели GOMS. Затем расставляются ментальные операторы. Наконец, ментальные операторы удаляются в отдельных местах.

- ***Правило 0.*** Начальная расстановка операторов M

  Операторы M следует устанавливать перед всеми K (нажатие клавиши) и P (указание с помощью ГУВ), предназначенными для выбора команд; но перед операторами P, предназначенными для указания на аргументы этих команд, ставить оператор M не следует.

[4]графическое устройство ввода данных, например, мышь.
[5]анализ времени отклика компьютеров: 1977-2017

- ***Правило 1.*** Удаление ожидаемых операторов М

  Если оператор, следующий за оператором М, является полностью ожидаемым, с точки зрения оператора, предшествующего М, то этот оператор М может быть удалён.

  Например, если вы перемещаете ГУВ с намерением нажать его кнопку по достижении цели движения, то в соответствии с этим правилом следует удалить оператор М, устанавливаемый по правилу 0. В этом случае последовательность РМК превращается в РК.

- ***Правило 2.*** Удаление операторов М внутри когнитивных единиц

  Если строка вида М К М К М К… принадлежит когнитивной единице, то следует удалить все операторы М, кроме первого.

  Когнитивной единицей является непрерывная последовательность вводимых символов, которые могут образовывать название команды или аргумент. Например, «Елена Троянская» или «-4564.23» являются примерами когнитивных единиц.

- ***Правило 3.*** Удаление операторов М перед последовательными разделителями

  Если оператор К означает лишний разделитель, стоящий в конце когнитивной единицы (например, разделитель команды, следующий сразу за разделителем аргумента этой команды), то следует удалить оператор М, стоящий перед ним.

- ***Правило 4.*** Удаление операторов М, которые являются прерывателями команд

  Если оператор К является разделителем, стоящим после постоянной строки (например, название команды или любая последовательность символов, которая каждый раз вводится в неизменном виде), то следует удалить оператор М, стоящий перед ним. (Добавление разделителя станет привычным действием, и поэтому разделитель станет частью строки и не будет требовать специального оператора М.) Но если оператор К является разде-

лителем для строки аргументов или любой другой изменяемой строки, то оператор M следует сохранить перед ним.

- ***Правило 5.*** Удаление перекрывающих операторов M

  Любую часть оператора M, которая перекрывает оператор R, означающий задержку, связанную с ожиданием ответа компьютера, учитывать не следует.

Пример оценки интерфейса

Задача пользователя: ввести логин (5 символов) и пароль (5 символов) и нажать OK. Взаимодействовать с формой авторизации пользователи могут по-разному. Переходить от одного поля ввода к другому с помощью клавиши tab или клика мыши. Здесь рассмотрим вариант, когда пользователь не знаком с горячими клавишами (рис. 6.11). Исходное положение: рука пользователя лежит на мыши (после запуска программы), поле ввода логина уже в фокусе ввода. Сравнить среднее время с другим вариантом взаимодействия с интерфейсом предлагается читателю.

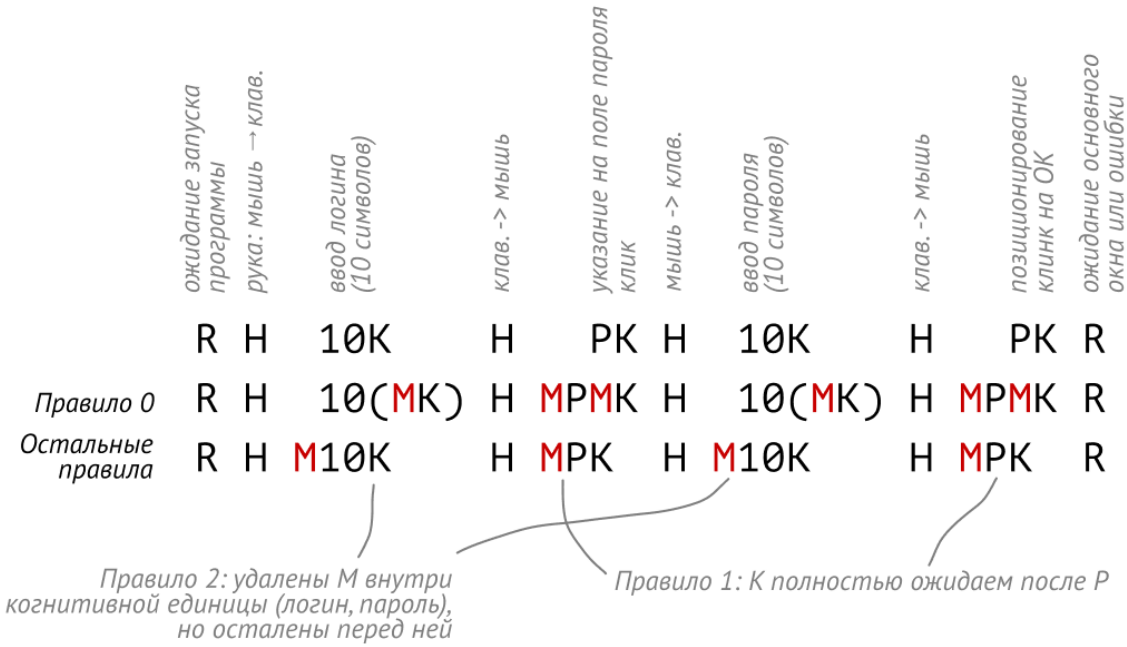


Рис. 6.11. Начальная запись жестов модели GOMS для окна ввода 6.12; применение правила 0 для расстановки когнитивных операторов; удаление когнитивных операторов по правилам 1-5.

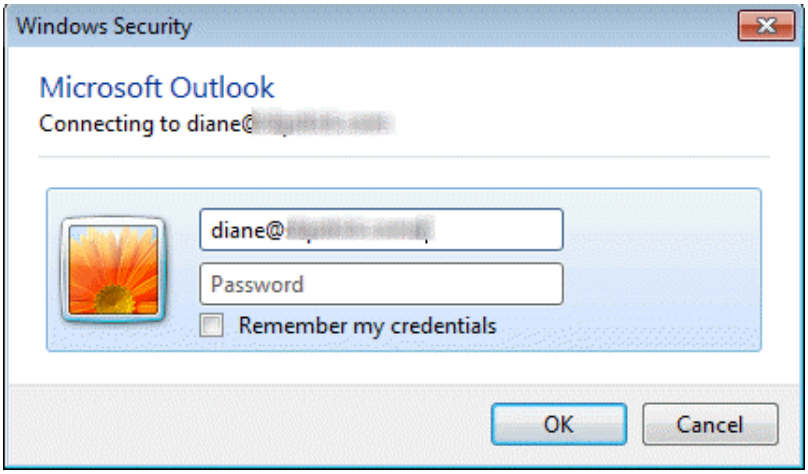


Рис. 6.12. Окно авторизации в Microsoft Outlook. Пример для оценки времени решения задачи пользователем по модели GOMS.

Некоторые способы ввода данных с точки зрения GOMS занимают примерно одинаковое время. Например, задание значения числового параметра с помощью ползунка займёт 2.4 секунды (РКРК): навести указатель, нажать клавишу мыши, навести на новое значение, отпустить клавишу. Ввод трёхзначного числа в поле клик по поле ввода займёт 2.3 секунды (РК-Н-ККК). Но оперировать небольшим ползунком, который перемещается вдоль одной оси проще чем, указывать на любую другую цель на экране. Фактически это будет занимать почти всегда меньше времени чем 1.1 с. С другой стороны, точно указать значение с помощью ползунка быстро вряд ли можно.

Иногда стоит комбинировать способы ввода данных, чтобы пользователь мог выбрать наиболее удобный для него способ взаимодействия (рис. 6.13). Пользователь может воспользоваться ползунком для быстрого выбора оттенка (Hue) и палитрой для выбора насыщенности и светлоты (saturation и value). Для точного задания этих параметров можно ввести значения в поля ввода. Список использованных ранее цветов (document colors) вообще избавляет от процедуры задания всех параметров цвета по отдельности.

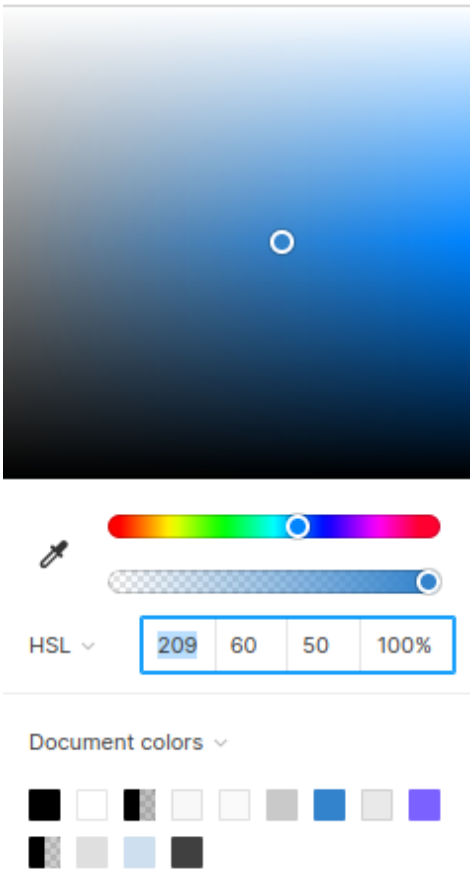


Рис. 6.13. Выбор цвета в Figma. Пользователь может воспользоваться ползунком для быстрого выбора оттенка (Hue) и палитрой для выбора насыщенности и светлоты (saturation и value). Для точного задания этих параметров можно ввести значения в поля ввода. Список использованных ранее цветов (document colors) вообще избавляет от процедуры задания всех параметров цвета по отдельности.

6.5 Информационная эффективность интерфейса

6.5.1 Информативность интерфейса

GOMS и закон Фиттса оценивают время на совершения операций. Другая характеристика пользовательского интерфейса – количество информации, которое вводит пользователь.

Информативность — это доля смысловой части в общей длине сообщения.

Интерфейс предполагает обмен информацией, а пропускная способность любого канала ограничена. Поэтому высокая информативность — признак хорошего интерфейса.

Повышать информативность можно двумя способами: убирая лишние или пряча редко используемые элементы интерфейса, создавая меньше "визуального мусора"и добавлять смысловую нагрузку существующим элементам интерфейса. Например в окне с сообщением

об обновлении (рис. 6.15) дано краткое описание изменений в сообщении о новой версии программы, а на кнопке «Купить» указана цена.

Интерфейс старых браузеров (рис. 6.14) содержал множество видимых элементов интерфейса (вверху, браузер NetScape Navigator). Современные браузеры, при возросшем разрешении дисплеев, (внизу, Safari, 2017) скрывают меню, многие кнопки скрыты или расположены компактно вместе с адресной строкой, полоса прокрутки и строка состояния появляются только когда это необходимо.

Один из приёмов повышения информативности – вынос за скобки. Повторяющуюся информацию (слова, пиктограммы и т.п.) в перечных приводит один раз, вынося на один уровень иерархии выше или опускают, если это понятно из контекста (рис. 6.16, 6.17). На рисунке 3.12 первая колонка таблицы содержит повторяющиеся значения. В улучшенной версии таблицы (рис. 3.13) каждое значение проводится один раз, одновременно являясь как бы заголовком к остальному содержимому.

6.5.2 Оценка информативности

Информационная производительность (эффективность) интерфейса ***E*** – это отношение минимального количества информации $I_{\text{мин}}$, необходимого для выполнения задачи, к количеству информации $I_{\text{введ.}}$, которое фактически требуется ввести от пользователя.

$$E = \frac{I_{\text{мин}}}{I_{\text{введ.}}} \tag{6.4}$$

Дополнительно рассмотрим пограничные случаи, не учитываемые формулой 6.4. Если никакой работы для выполнения задачи не требуется или работа просто не производится, то производительность составляет 1. Если действия не требуется, но оно производится, то производительность составляет 0 (рис. 6.18).

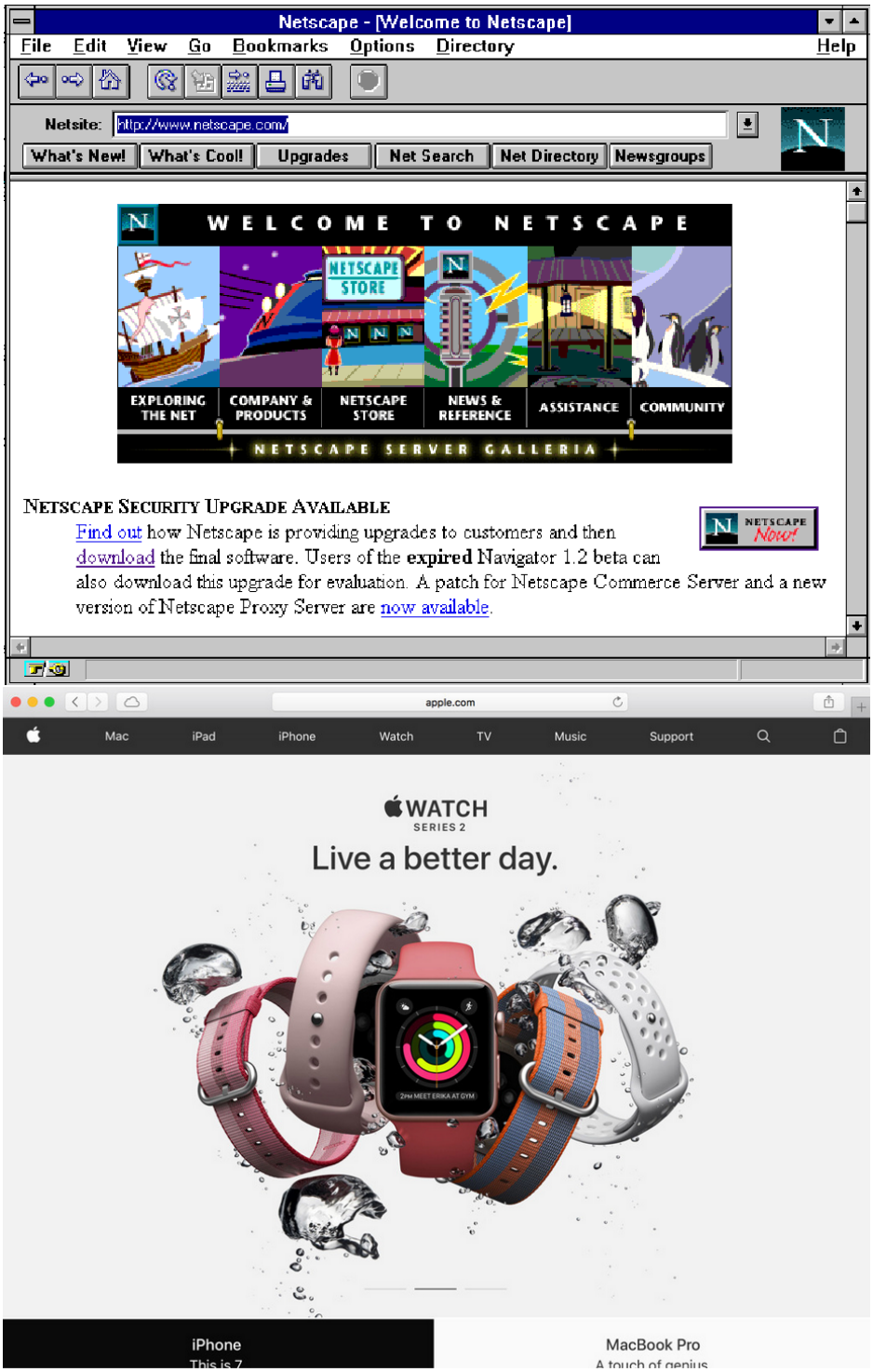


Рис. 6.14. Интерфейс старых браузеров содержал множество видимых элементов интерфейса (вверху, браузер NetScape Navigator). Современные браузеры, при возросшем разрешении дисплеев, (внизу, Safari, 2017) скрывают меню, многие кнопки скрыты или расположены компактно вместе с адресной строкой, полоса прокрутки и строка состояния появляются только когда это необходимо [?]



Рис. 6.15. Примеры повышение информативности интерфейса: краткое описание изменений в сообщении о новой версии программы. Объединение кнопки "Купить"и цены товара [**?**]

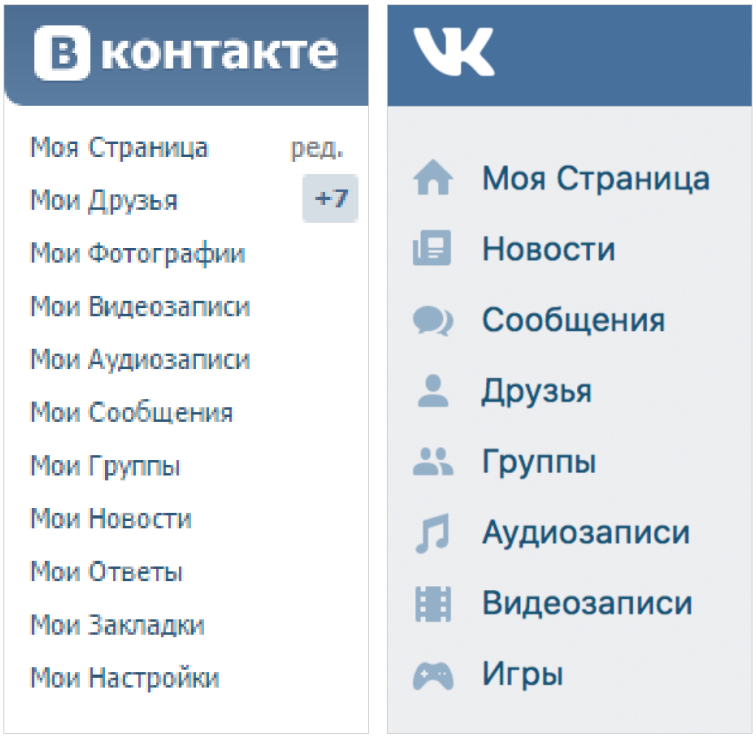


Рис. 6.16. Слово “Мои“ можно вынести за скобки [**?**]

Поискать в регионах: **СНГ**; **Россия** (Красноярск, Томск, Новосибирск, Кемерово)

Поискать в рубрике: **Культура и искусство** (Русская проза, Библиотеки, Живопись)
Общество и политика

Поискать то же самое на: AltaVista · Google · MSN · Yahoo

Поискать то же самое

в регионе: **СНГ**; **Россия** (Красноярск, Томск, Новосибирск, Кемерово)

в рубрике: **Культура и искусство** (Русская проза, Библиотеки, Живопись)
Общество и политика

в других поисковых системах: AltaVista · Google · MSN · Yahoo

«Пушкин»

в регионе: **СНГ**; **Россия** (Красноярск, Томск, Новосибирск, Кемерово)

в рубрике: **Культура и искусство** (Русская проза, Библиотеки, Живопись)
Общество и политика

в других поисковых системах: AltaVista · Google · MSN · Yahoo

Рис. 6.17. Улучшения предложения продолжить поиск (верхний вариант, такой блок приводился ниже результатов поиска на странице yadex.ru) в два этапа: вынос за скобки “Поискать“, замена обозначения данных "то же самое"на сам поисковый запрос [**?**].

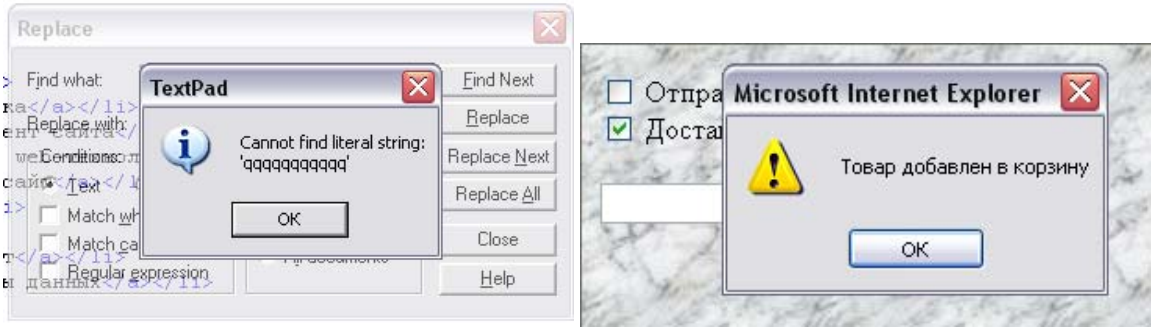


Рис. 6.18. Нулевая информационная эффективность: данных для ввода не требуется, но пользователь должен ввести один бит информации нажав на кнопку OK.

Количество вводимой информации, когда пользователь совершает выбор из n альтернатив, определяется выражением:

$$I = \log_2(n)$$

При этом отказ от выбора не рассматривается.

Если же альтернативы не равновероятны, информация, передаваемая i-й альтернативой, определяется по формуле

$$I = p_i \log_2(1/p_i)$$

где p_i - вероятность выбора i-й альтернативы.

Символьная эффективность определяется как минимальное количество символов, необходимое для выполнения задачи, отнесённое к количеству символов, которое в данном интерфейсе требуется ввести пользователем.

Вопросы

1. Сформулируйте закон Фиттса. Какие величины в него входят?
2. Какие условия применения закона?
3. Какие рекомендации следуют из закона Фиттса?
4. На какие 5 пикселей на экране можно кликнуть быстрее всего?
5. Как влияет расположение элемента возле краёв экрана на время указания на него?
6. Сформулируйте закон Хика. Какие выводы из него следуют?
7. Как уменьшить время поиска или выбора элемента в списке?
8. Для чего используется метод GOMS?
9. Какие элементарные жесты входят в модель GOMS?
10. Назовите правила записи жестов.

Литература

- Раскин Д., Интерфейс: новые направления в проектировании компьютерных систем. – Пер. с англ. – СПб: СимволПлюс, 2007. – 272 с.
- Бирман И. Б. Пользовательский интерфейс. — М.: Изд-во Бюро Горбунова, 2017. Ознакомительный фрагмент книги: https://bureau.ru/books/ui/demo/

7 Типографика текст

7.1 Понятие типографики

Около 90% всей визуальной информации человек получает через текст. Больш*ая* часть информации в пользовательских интерфейсах – тоже текст (рис. 7.1). Поэтому важно понимать, как доносить информацию через текст эффективно.

Представления о способах передачи текстовой информации менялись вместе с технологиями тиражирования и отображения текстовой информации. Начиная со времени когда стали широко доступны печатные машинки, а потом матричные принтеры в широкой практике сложились определённые привила структурирования текстовой информации. Часть правил, из-за скудных технических возможностей печатающих устройств, были актуальны для своего времени. Но некоторые из них, вроде частого выделения важных слов в тексте подчёркиванием, сохранились до сих пор. При этом широкий круг возможностей электронной типографики часто игнорируется. Для выделения важного часто используется подчёркивание, полужирное начертание, верхний регистр, яркий красный цвет, но реже просто увеличение размера, смена цвета на заметный, но гармонирующий с остальным текстом. Заголовки, пояснения и другие особые элементы текста иногда никак не выделяются, создавая эффект стены текста. С другой стороны, многие возможности используются во вред. Часто случается неуместное использование многих узконаправленных или специализированных шрифтов (Comis Sans, готические и рукописные шрифты), неудач-

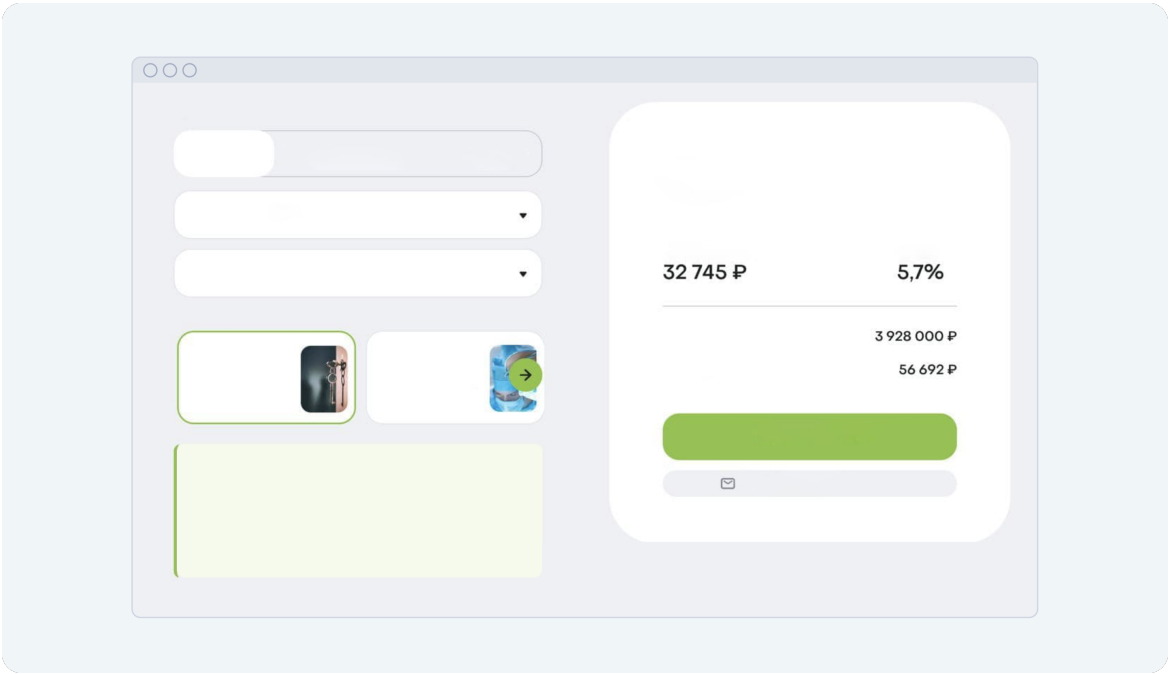


Рис. 7.1. Графический пользовательский интерфейс без слов чаще всего невозможен.

ная комбинация большого количества шрифтов, концентрация на эстетической составляющий в ущерб читаемости.

Типографика – это технология и искусство делать текст различимым (legibility), удобочитаемым (readability) и эстетичным (aesthetic).

Различимость (legibility) – свойство символов быть отличными друг от друга (рис. 7.2). Читаемость (readability) – свойства текста, характеризующее простоту восприятия читателем, зависящую в частности от свойств шрифта.

Типографика, с одной стороны, представляет собой одну из отраслей графического дизайна, с другой – содержит свод строгих правил, определяющих использование шрифтов для создания наиболее понятного для восприятия читателя текста.

Рассмотрим основные понятия типографики и приведём классификацию шрифтов.

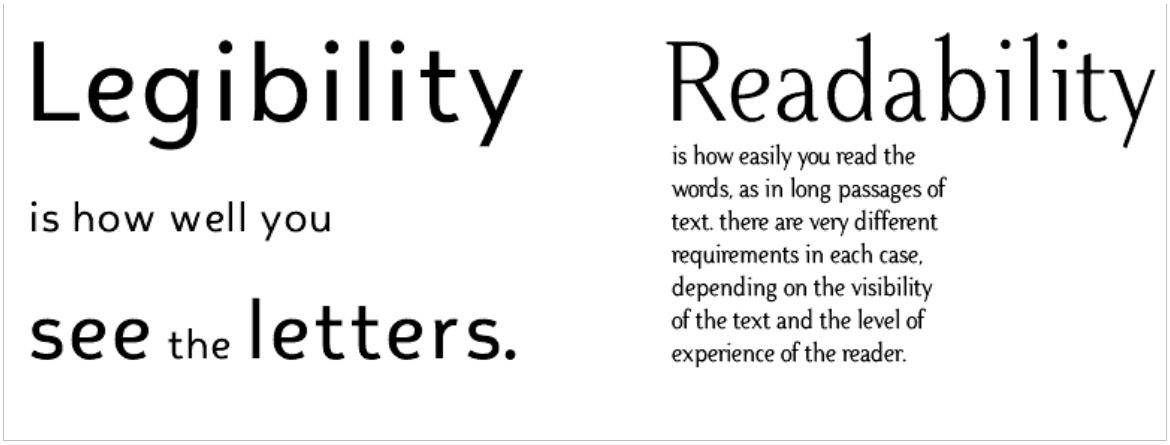


Рис. 7.2. Различимость (legibility) – свойство символов быть отличными друг от друга; читаемость (readability) – свойства текста, характеризующее простоту восприятия читателем, зависящую в частности от свойств шрифта.

7.2 Классификация шрифтов

В первую очередь шрифты можно разделить (см. рис 7.3) на две категории – шрифты с засечками (serif, антиква) и шрифты без засечек (sans serif, гротеск).

Нет убедительных доказательств того, что антиква или гротеск должны использоваться в отдельных ситуациях. Часто рекомендуют использовать шрифт с засечками для больших текстов. Например, в оформлении документов, книг, текстовых статей на сайтах. Шрифт без засечек рекомендуют для отдельных, коротких надписей. Иногда рекомендуют использовать антикву для печатных текстов, гротеск – для электронной типографики. Эти рекомендации имеют много исключений и чаще описывают традицию, не основываясь на исследованиях.

Брусковый шрифт — шрифт с мощными засечками прямоугольной формы без скруглений или с небольшим скруглением в местах присоединения к основным штрихам (см. рис 7.5). В таких шрифтах толщина основных и дополнительных штрихов отличается меньше чем в антикве.

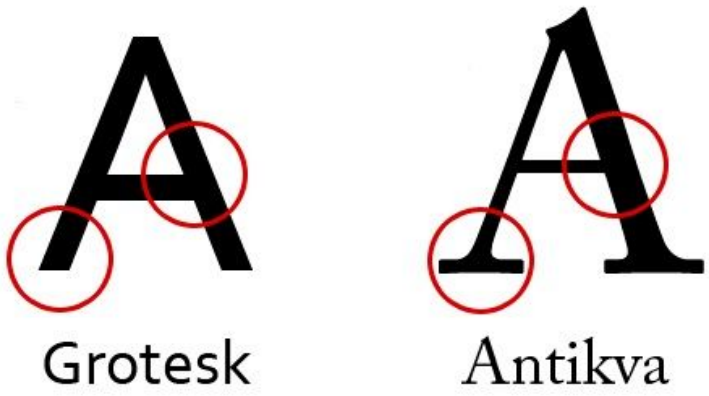


Рис. 7.3. Гротеск – sans serif – шрифт без засечек. Антиква – serif – шрифт с засечками. Антиквенные шрифты, как правило, имеют разную толщину основного и дополнительно штриха, в отличие от гротеска.

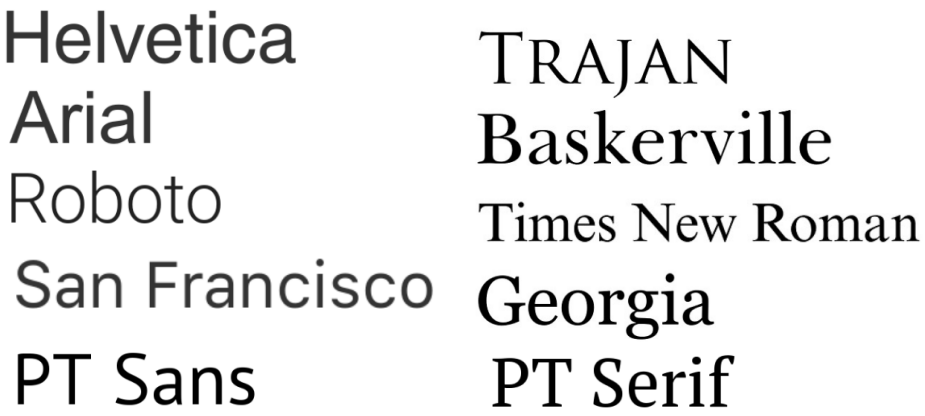


Рис. 7.4. Шрифты с засечками (слева) и без засечек (справа)

Handgloves

Handgloves

Handgloves

Рис. 7.5. Антиква (вверху) и её подвид – брусковый шрифт (внизу), в середине – шрифт имеющий черты антиквы и брускового.



Рис. 7.6. Акцентные шрифты своим хара*к*терным видом часто призваны соответствовать общей эстетике дизайна, и не обязательно имеют высокую различимость и читаемость

Шрифты этого типа занимают ведущее место по шкале читабельности и идеально подходят для набора длинных текстов, так как имеют очень слабый контраст. Однако страница, набранная этим шрифтом, выглядит значительно «темнее» страницы, набранной обычной гарнитурой, поскольку штрихи брускового типа плотнее и более единообразны по толщине. Брусковый шрифт часто используется при наборе детских книг.

Акцидентные (декоративные) шрифты – самая большая группа шрифтов, непохожих друг на друга. Такие шрифты не предназначены для набора основного текста и используются для заголовков, небольших отрывков текста с целью привлечения и акцентирования внимания или служат исключительно эстетическим целям (рис. 7.6). Эти шрифты обязательно содержат начертания для всех символов и нередко создаются прямо во время работы над общим дизайном продукта, например, визитки или логотипа.



Рис. 7.7. Пропорциональный (proportional) и моноширинные (monospaced) шрифты.

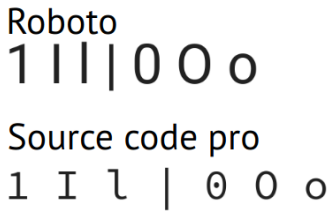


Рис. 7.8. В моноширинных шрифтах (например, Source Code Pro), часто уделено много внимания различимости похожих символов. В пропорциональных шрифтах (например, Roboto) бывают очень похожие символы.

Более формальный подход к классификации шрифтов – учитывать различия в ширине символов (рис. 7.7):

- Пропорциональные;
- Моноширинные.

Символы пропорциональных шрифтов имеют ширину, пропорциональную их конструкции. Например, буква ж шире чем к. В моноширинных шрифтах все символы имеют одинаковую ширину.

Используйте моноширинные шрифты когда: важно выравнивание символов (см. рис: 7.9) и различимость (см. рис 7.8) символов. Некоторые шрифты имеют режим отображения цифр с подстройкой одинаковой ширины (OpenType feature – Tabular Figures). Последнее важно, например, для отображения числовых данных в колонке таблицы так, чтобы разряды находились друг под другом (см. рис 3.13). Некоторые моноширинные шрифты имеют лигатуры (слитное начертание нескольких символов) для обозначений операторов в программном коде (см. рис: 7.9).

Наконец приведём наименее формальную классификацию шрифтов: нейтральные и хар*а*ктерные. Нейтральные шрифты не выделяются, у них нет заметных особенностей. Часто они выглядят похожими друг на друга. Эти шрифты часто служат утилитарной цели – доносить информацию, не привлекая внимания к особенностям начертания символов. Характерные шрифты непохожи на другие,



Рис. 7.9. Моноширинные шрифты используются в редакторах кода, где важно выравнивание кода. Некоторые моноширинные шрифты имеют лигатуры – отдельные начертания, обозначающие популярные сочетания символов, например для ->.
Слева – шрифт Consolas, справа – шрифт Fira Code.

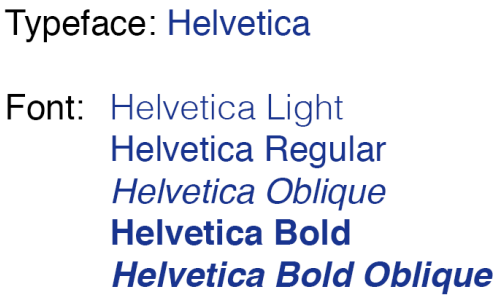


Рис. 7.10. Шрифт (font) и гарнитура (typeface).

их легче различать. Для них важно узнавание, поэтому их часто разрабатывают как часть экосистемы бренда.

7.3 Основные понятия из типографики

Шрифт (font) — графический рисунок начертаний букв и знаков, составляющих единую стилистическую и композиционную систему, набор символов определённого размера и рисунка. Гарнитура (typeface) — набор из одного или нескольких шрифтов в одном или нескольких размерах и начертаниях, имеющих стилевое единство рисунка и состоящих из определённого набора типографских знаков (см. рис 7.10).



Рис. 7.11. Гарнитура Roboto, начертания с толщинами от 100 до 900

Одна гарнитура может содержать несколько начертаний отличающихся:

- по толщине (м. рис 7.11);
- по стилю: прямое начертание (roman), наклонное (oblique), курсив (italic), см. рисунок 7.12;
- по плотности начертания: обычная (normal), плотное (condensed), разреженное (extended) и др., см. рис 7.13.

Наклонное начертание часто представляет собой символы прямого начертания, только с наклоном, в отличие от курсива, где символы могут иметь совершенно другое начертание. Это хорошо видно на примере буквы а, на рисунке 7.12.

Не все шрифты содержат все перечисленные виды начертаний. Если в шрифте нет курсивного начертания, то оно часто заменяется наклоном символов без изменения начертания.

Наконец шрифты отличаются набором символов. Например, многие латинские шрифты не содержат начертаний для кириллических алфавитов. Многие алфавиты для китайского языка содержат всего один вид начертания латинского алфавита.

Разработка шрифта – долгий процесс, требующий от дизайнера рисования большого числа символов. Поэтому, часто разработчики ограничиваются малым числом начертаний, исключают курсив. Например, шрифт Roboto, разработанный компанией Google, имеет 12

Normal - Sphinx of black quartz, judge my vow.

Italic - Sphinx of black quartz, judge my vow.

Oblique - Sphinx of black quartz, judge my vow.

Рис. 7.12. Прямое начертание (roman), наклонное (oblique), курсив (italic). Характерные различия между прямым и курсивным начертаниями видны на примере буквы a.

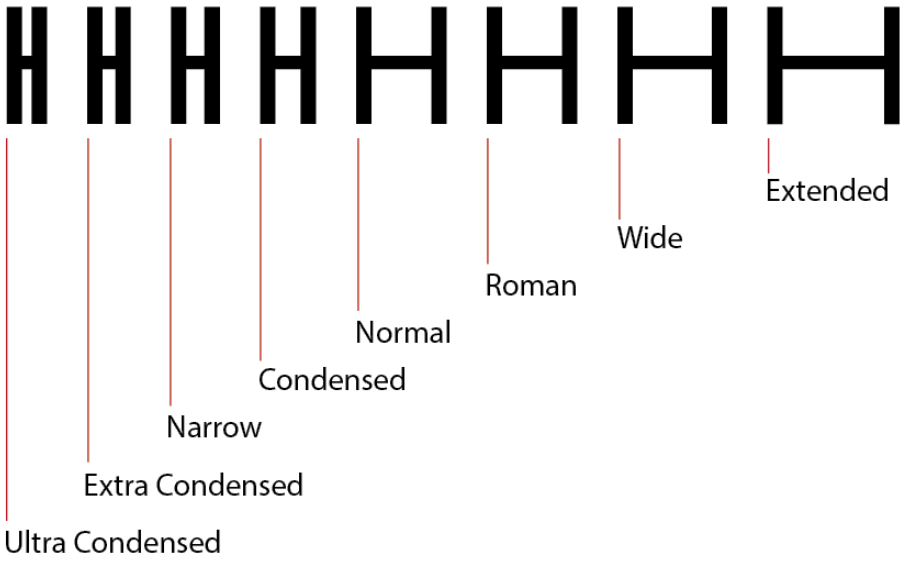


Рис. 7.13. Различные плотности начертания. Вариации плотности встречается чаще всего в интерфейсных и переменных шрифтах.



Рис. 7.14. Наличие равномерного расстояния между символами в слове не означает, что и визуальное (воспринимаемое человеком) расстояние будет таким же.



Рис. 7.15. Фрагмент веб-страницы (v-fonts.com) с демонстрацией переменного (Variable) шрифта Helvetica Now.

начертаний отличающихся толщиной (100, 300, 400, 500, 700, 900) и начертанием (regular и italic), а шрифт PT Sans, разработанный компанией ParaType имеет 4 начертания.

Разработанный шрифт помимо самих начертаний ещё содержит информацию о сочетаниях символов. Кернинг (kerning) – избирательное изменение интервала между буквами в зависимости от их формы.

Проблема создания большого числа начертаний может быть частично решена автоматической настройкой параметров шрифта (например, толщины и плотности) по заданным разработчиками шрифта правилам. Такие шрифты называются переменными (variable). Примеры можно увидеть на сайте: v-fonts.com (рис. 7.15)

7.3.1 Выбор шрифта

Подбор шрифта или сочетания шрифтов – это творческая задача, плохо поддающаяся формализации. Но можно выделить отдельные рекомендации.

Во-первых, шрифт должен хорошо читаться в предполагаемых условиях использования. В Некоторых случаях особенно важна хорошая различимость символов. Например, если шрифт будет использоваться для отображения логинов, паролей, исходного кода на языке программирования. Многие гарнитуры специально создаются для определённых целей и носителей.

Например, Roboto, San Francisco и Segoe созданы специально для отображения на дисплеях. Они содержат множество начертаний, поэтому всего одна гарнитура может быть использована для всех элементов пользовательского интерфейса (рис. 7.16). Шрифт Comic Sans был разработан для текста показываемого в «облачках слов» виртуальных помощников в Microsoft Bob (рис. 2.8). Используемый ранее Times New Roman по стилю плохо подходил для этих целей.

Во-вторых, шрифт должен соответствовать общему дизайну, бренду назначению текста. Нейтральные шрифты могут быть использованы практически везде. Особенно хорошо строгие и нейтральные шрифты (рис. 7.17) подходят для документов. Шрифт может быть описан в руководстве по дизайну (Human Interface Guidlines [45, 49, 40, 41, 42, 46]) для операционной системы или платформы, брендбуке компании или руководстве по фирменному стилю.

Наконец, самый субъективный способ: опишите бренд, сайт или текст набором прилагательных, подберите шрифт, которому также подходят эти прилагательные. Например ярлык с ценой для кофе (рис. 7.18) может существенно влиять на восприятие продукта, подчёркивать его свойства. Times New Roman можно встретить часто, например в документах и в небрежно сделанных объявлениях.

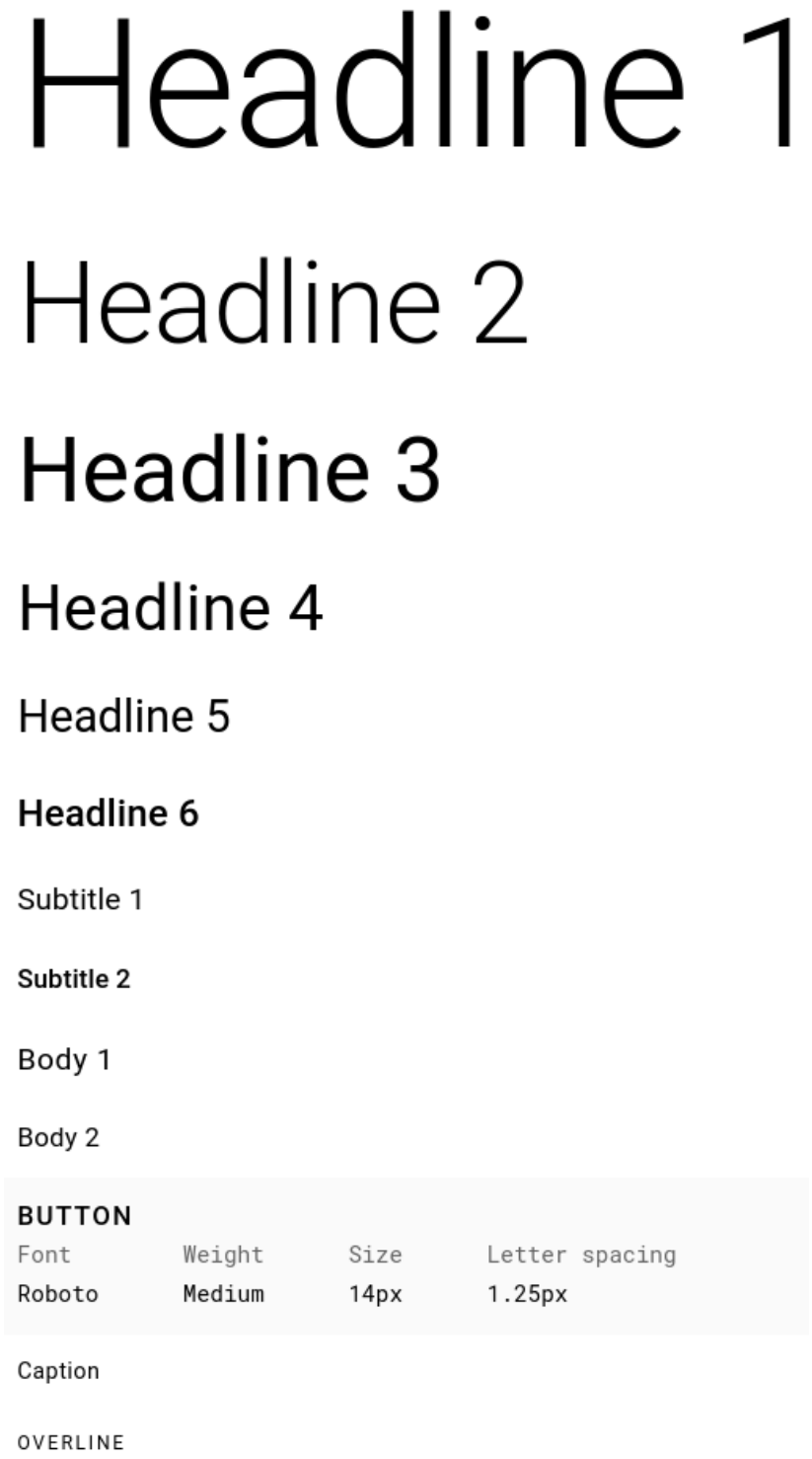


Рис. 7.16. Рекомендации Material Design по использованию гарнитуры Roboto для разных элементов интерфейса

Курсовая работа по программированию
«База данных библиотеки»

Курсовая работа по программированию
«База данных библиотеки»

Курсовая работа по программированию
«База данных библиотеки»

Курсовая работа по программированию
«База данных библиотеки»

Рис. 7.17. Шрифт Пушкин (внизу) – слишком хар*а*ктерный для документа, с плохой удобочитаемостью и различимостью. Courier New (второй снизу) наоборот обладает хорошей различимостью, но удобочитаемость моноширинных шрифтов для больших текстов сравнительно невысокая. Comic Sans (второй сверху) выглядит слегка небрежно для такого документы, буквы неровные, будто написаны от руки. PT Serif – шрифт с засечками, какие традиционно используются в документах, выглядит достаточно нейтрально для документа.



Рис. 7.18. Шрифт может существенно влиять на восприятие продукта, подчёркивать его свойства. Times New Roman – шрифт используемый по умолчанию во многих ситуациях, для документов и небрежно сделанных объявлений. В этом случае он, возможно, даст шанс сформировать невысокие ожидания.

Поэтому весьма вероятно, что он поможет пользователю сформировать невысокое мнение о дизайне в целом.

7.3.2 Шрифтовая иерархия

Иерархия в типографике – способ показать отношение между отдельными частями текста. Например, заголовки имеют больший по отношению к основному тексту размер.

Иерархия даёт понять[1]:

- структуру сообщения (заголовок, подзаголовок, краткое содержание, основное сообщение, второстепенная информация и т. д.);
- отношение между текстовыми блоками.

Исследования [35] показывают, что больше половины пользователей просматривают веб-страницы, но не читают их содержимое последовательно и целиком. Для таких пользователей важно быстро понять структуру страницы или сообщения, находить важную и интересную для них информацию. Правильно выстроенная шрифтовая иерархия облегчает сканирующее чтение (рис. 7.19, 7.20, 7.21).

Иерархию можно показывать используя:

- Размер шрифта;
- Отступы;
- Цвет текста;

[1]Элементы иерархия этого перечисления: заголовок и перечисляемые элементы. Отношение показано с помощью отступов и маркеров списка

- Толщину, начертание (*курсив*, *наклонный*, обычный);
- ПРОПИСНЫЕ, строчные буквы, капит*е*ль (маленькие заглавные, small caps)
- Т р е к и н г;
- Гарнитуру.

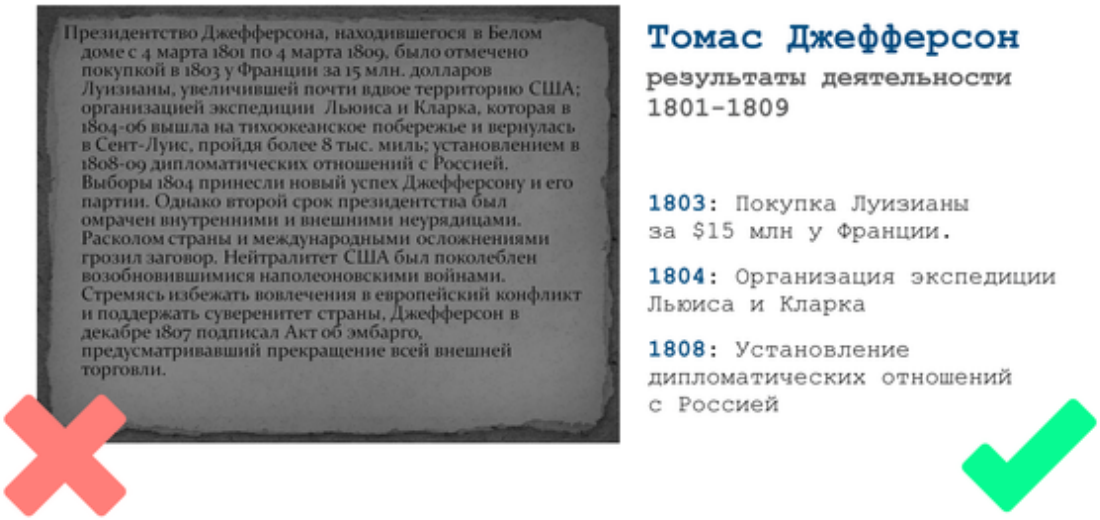


Рис. 7.19. Плохо структурированное сообщение читать не хочется, оно кажется скучным. В этом примере меньшее количество информации хорошо оформленного текста полезнее, чем много неструктурированного текста, который прочитает мало людей.

7.3.3 Выравнивание текста и длина строки.

При выравнивании текста по ширине лучше избегать появления непропорционально длинных пробелов. Выравнивание по левому краю лучше выравнивания по ширине, если возникают длинные пробелы. Перенос слов *может* помочь избавится от длинных пробелов. В некоторых программах для вёрстки текстов возможна тонкая подстройка параметров выравнивания текста: задание ограничений на увеличение пробела, трекинга и размеров букв на 1–3%. Такие изменения практически незаметны, но могут помочь добиться ровной правой границы текста.

Считается, что оптимальная длинна строки 40–70 знаков (рис. 7.24). Длина строки в статьях на wikipedia.org больше 200 символов (рис.



Рис. 7.20. Веб-страница без хорошо заметной визуальной структуры (слева) и с хорошо заметной структурой (справа). Структура страницы, шрифтовая иерархия должна облегчать сканирующее чтение.

Чем отличаются относительные и абсолютные риски

УЗНАТЬ ↓

Омикрон обладает меньшей природной вирулентностью почти для всех. Но не для детей

Первые оценки вирулентности для невакцинированных и неболевших, о которых уже писала «[redacted]», говорили о том, что омикрон, видимо, реже вызывает тяжелые заболевания, чем дельта. Для людей, которые никогда не сталкивались с SARS-CoV-2 и не были привиты, риск госпитализации оценивался примерно на 25–30% меньше. Новое исследование группы Фергюсона даже улучшило эту оценку — в среднем по популяции относительный риск попасть в больницу при омикроне по сравнению с дельтой уменьшился на 70% (но речь здесь только о тех, у кого нет никакого иммунитета).

Пример:

> Допустим, в некоем абстрактном городе, где еще нет вакцинированных, происходит распространение варианта дельта.

Рис. 7.21. Помимо заголовка, выделенной шрифтом без засечек аннотации, в статье выделены подзаголовки, некоторые абзацы скрыты, отдельно отмечены примеры и главные идеи. Такая структура страницы облегчает сканирующее чтение.

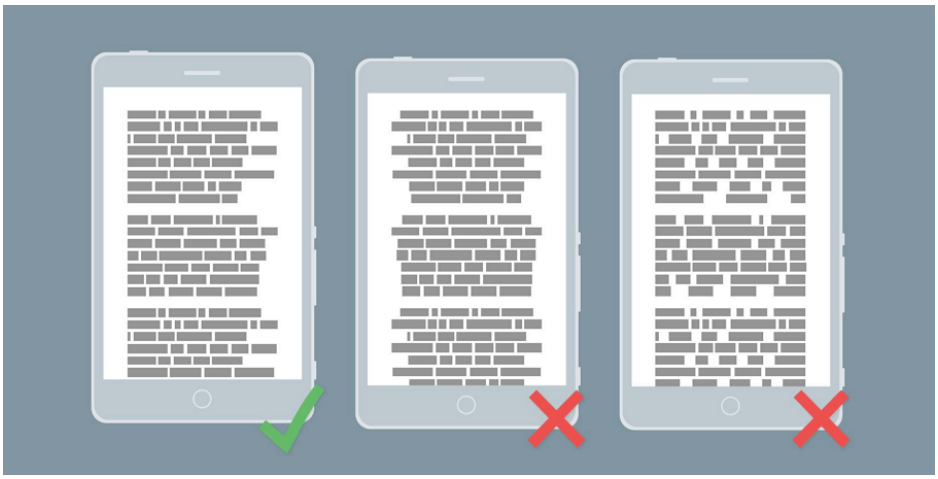


Рис. 7.22. На сайтах уместнее использовать выравнивание больших блоков текста по левому краю, избегая непропорциональных пробелов.

7.23). Дойдя до конца строки при чтении сложнее вернутся к началу следующий. При этом чем длиннее строка, тем больше должен быть интерлиньяж – вертикальное расстояние между соседними строками. При больших объёмах текста у пользователя может рябить в глазах. В этом случае помогает снижение контрастности между текстом и фоном. Чаще всего можно сменить чёрный цвет текста на тёмно-серый.

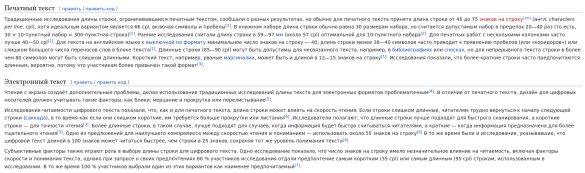


Рис. 7.23. Длина строки в статьях на wikipedia.org больше 200 символов. Дойдя до конца строки при чтении сложнее вернутся к началу следующий.

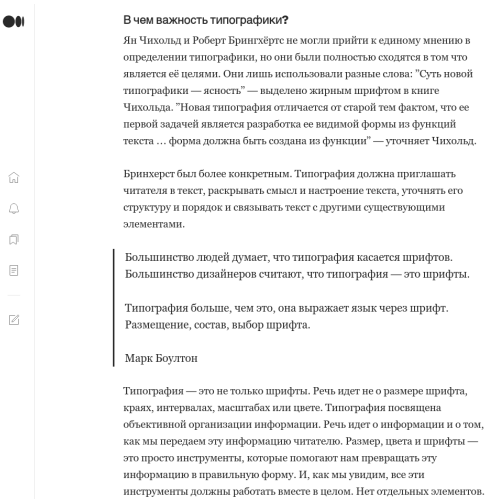


Рис. 7.24. Длина строки в статьях на medium.com около 60 символов.

7.4 Текст и синтаксис интерфейса

Как было отмечено, значительную часть информации человек получает в виде текста. Взаимодействия с пользователем редко обходится без текста.

Хорошая типографика не исправит ситуацию, если текст написан плохо: его трудно понять, он слишком длинный, там много “воды“, сложных речевых оборотов и незнакомых пользователю терминов.

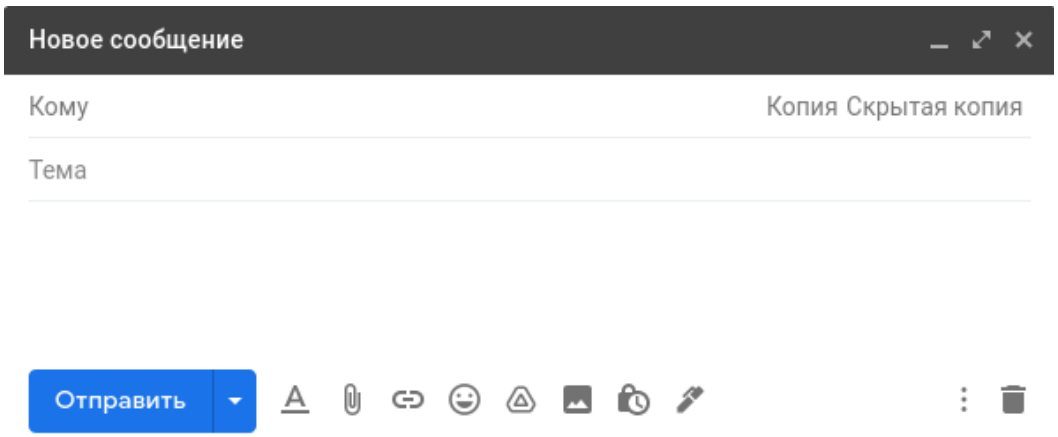


Рис. 7.25. Форма отправки сообщения на сайте Gmail

Поэтому важно ещё и уметь эффективно доносить информацию через тексты, надписи, пояснения и формы.

Многие диалоговые окна и формы можно рассматривать так, будто они образуют предложение или связный текст [1]. Например, форму отправки сообщения в Gmail (рис. 7.25) можно представить как отдельное предложение. Подлежащее – заголовок (существительное), сказуемое – кнопка отправить (глагол). Такой взгляд подражает привычной для человека подаче информации, с той лишь разницей, что из элементов интерфейса не всегда можно сложить текст механическим соединением. Элементы интерфейса имеют те же синтаксические отношения, что и слова в предложениях. Одни элементы интерфейса можно представлять главными членами предложениями – подлежащим[2] и сказуемом[3]. Не нужно добавлять к кнопке слово “нажмите“, к полю ввода слово “введите“ и т.д. (рис. 7.26) потому, что сам элемент интерфейса говорит за себя.

Кнопки и другие элементы интерфейса, отвечающие за совершения действий, стоит подписывать глаголами. Это особенно полезно в диалоговых окнах, где пользователь может понять смысл сообщения из подписей к кнопкам (рис. 7.27). Это помогает ускорить

[2]Подлежащее – главный член предложения, который обозначает предмет, действие которого выражается сказуемым

[3]Сказуемое — главный член предложения, которым бывает глагол, связанный с подлежащим и отвечающий на вопросы: что делает предмет? что с ним происходит? каков он? и др; обозначает действие или состояние.

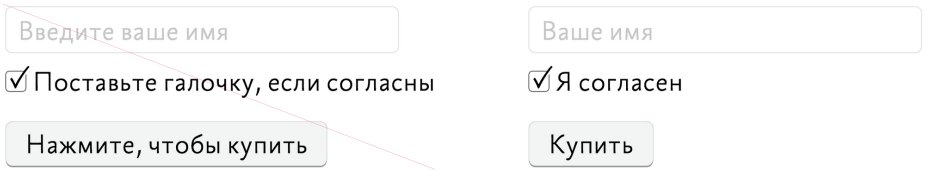


Рис. 7.26. Некорректные и корректные подписи элементов интерфейса [1]

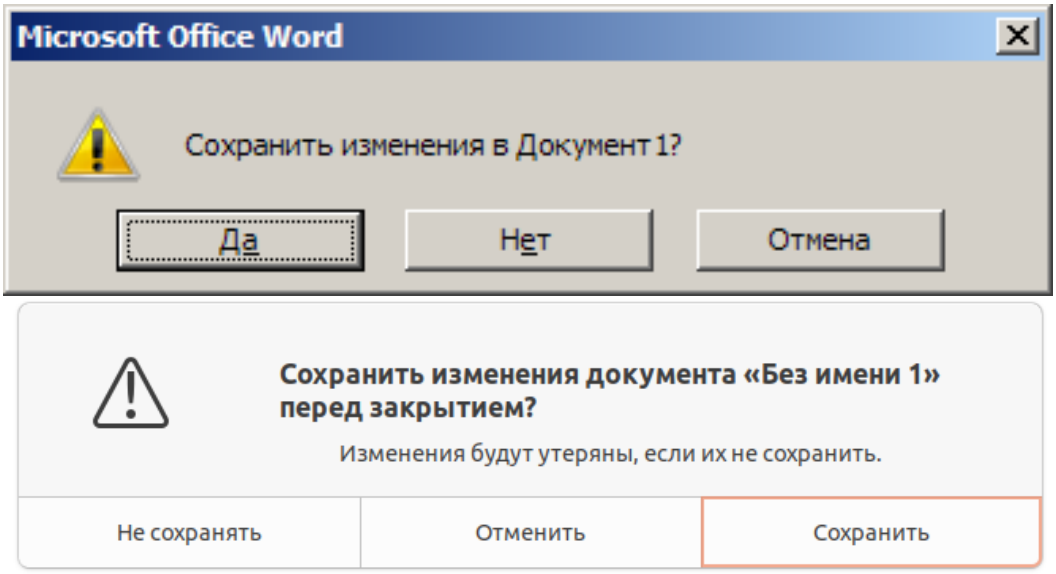


Рис. 7.27. Диалоговые окна Microsoft Word 2003 и Libre Offcie 5

работу пользователей знакомых с интерфейсом и даст повод обратить внимание на посыл сообщения тем пользователям, кто не часто обращает внимание на текст сообщения.

Кнопки и элементы меню могут ещё означать изменение статуса. Например “отметить письмо как Важное“ или “отметить письмо как Спам“. В таких случаях уместно выносить общий текст (“отметить письмо как“) за скобки и оставить на кнопках только статус: “Важное“ и “Спам“.

Большая часть действий пользователя – это применение некоторого *действия* к некоторому *объекту*. Например, *изменения шрифта* для *абзаца текста* в текстовом редакторе. В интерфейсе могут использоваться две последовательности: сначала выбор глагола (изменить шрифт), а затем выделение существительного (абзац); сначала существительное, а потом глагол.

Эксперименты [28] показывают, что в большинстве случаев модель существительное → глагол предпочтительнее. Это уменьшает количество ошибок. Последовательность глагол → существительное устанавливает *режим*. Это повышает скорость: пользователю не требуется переключать своё внимание с содержания (которое и вызвало необходимость выполнения операции) к самой команде и затем опять к содержанию. Помогает отменять действия: при использовании модели глагол → существительное должна быть предусмотрена возможность отмены команды. Если пользователь назначает команду и затем решает изменить её, следует учитывать, что он находится в этот момент в режиме, и система ожидает, что будет сделано выделение текста для применения уже назначенной команды.

Особено важно донести до пользователя информацию, когда он или программа совершили ошибку. Сообщения об ошибках должны соответствовать трём критериям (рис. 7.28):

- быть понятным (clear)
- быть лаконичным (concise)
- быть полезным (useful).

Сообщение об ошибке 404 на рисунке 7.30 понятно и лаконично, но не предлагает путей решения проблемы. Скорее всего ошибка появилась после поискового запроса, поэтому логично показать поисковую строку (рис. 7.30). Кроме того, Яндекс не упускает возможности показать ссылки на собственные сервисы.

Приведённые в первой части примеры сообщений об ошибках (рис 1.4) соответствуют только второму критерию.

Помимо интерфейсного текста, разработчик пишет справку, документацию, инструкции и иногда сопроводительный текст для сайта приложения или страницы в Google Play или App Store. Здесь требования лаконичности и ясности не менее важны.

Необходимость сжато и ясно передавать информацию текстом появилась задолго до пользовательских интерфейсов. Яркий пример

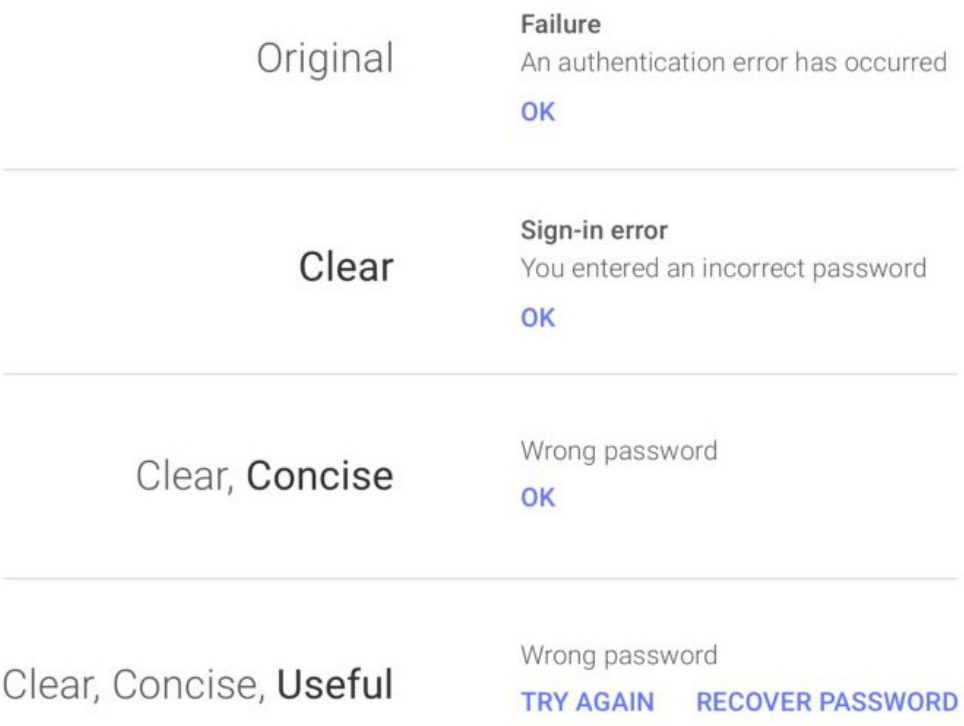


Рис. 7.28. 3 стадии улучшения сообщения (блок Original): Сбой. Произошла ошибка аутентификации. 1) Сделать сообщение понятным (clear): Ошибка входа. Вы ввели неправильный пароль. 2) Сделать лаконичным (consice): Неправильный пароль. 3) Сделать полезным (useful), добавить кнопки “попробовать ещё раз“ и “восстановить пароль“.

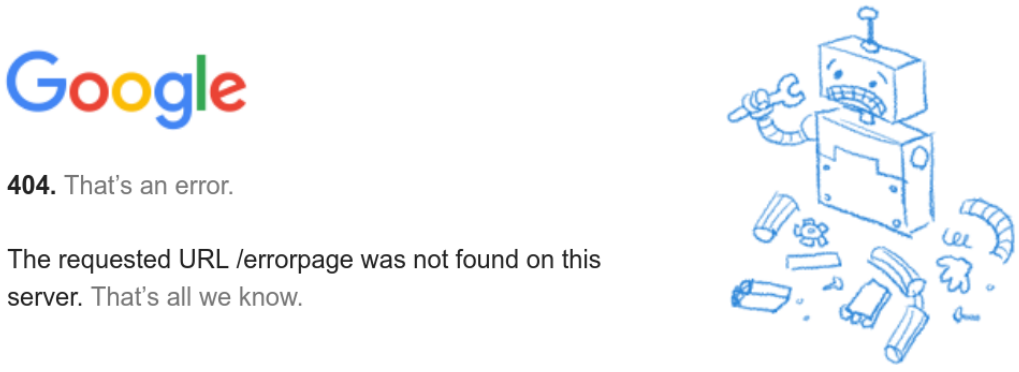


Рис. 7.29. Страница не предлагает пользователю полезных действий на странице с ошибкой 404.

Рис. 7.30. Страница 404 с сообщением об ошибке

сокращения текста демонстрирует Ольминский М. С.[4] на примере

[4]Ольминский Михаил Степанович – революционер, публицист, историк, литературный критик, литературовед и историк литературы

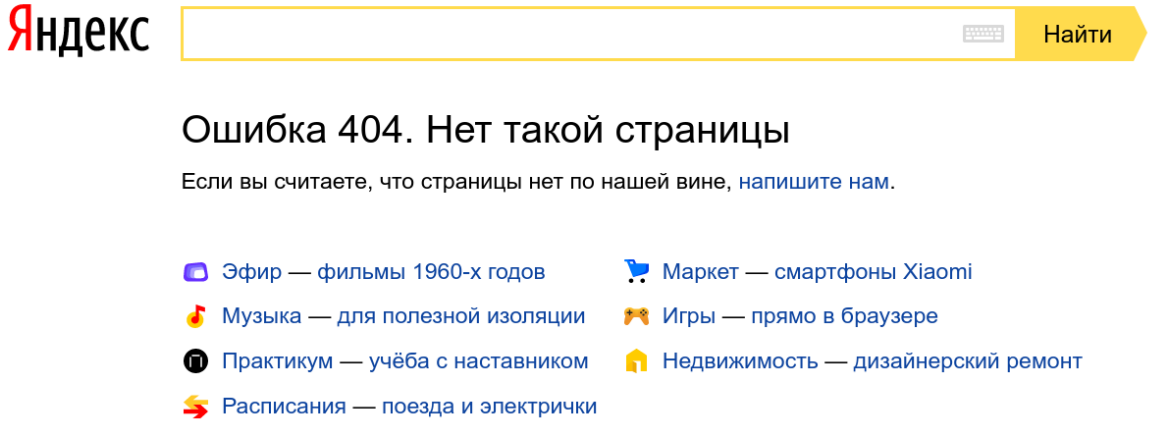


Рис. 7.31. Страница 404 с поисковой строкой

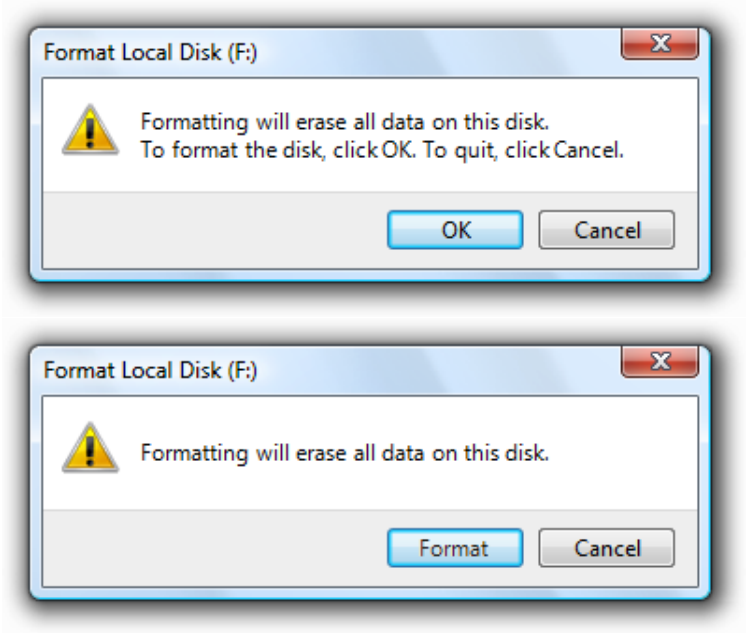


Рис. 7.32. Глаголы на кнопках предпочтительнее существительных там, где идёт речь о совершении действия

новостной заметки о демонстрации в Твери. Текст заканчивался словами: “Явившаяся на место происшествия местная полиция арестовала восемь человек демонстрантов“. Ольминский критикует:

– Местная – *разве в Твери могла явиться полиция не местная, а казанская?*

– Явившаяся на место происшествия – *разве могла она арестовать, не явившись?*

—Полиция – *кто же арестует, кроме полиции?*

– Человек демонстрантов – *конечно, не коров и не прохожих.*
Остаётся для печати: “Арестовано восемь“.

Упростить редактуру могут сервисы:

- orfogrammka.ru – проверка орфографии, пунктуации, стилистики и элементов типографики;
- glvrd.ru – Главред – оценка текста с точки зрения информационного стиля [13];
- hemingwayapp.com – оценка читаемости и понятности текстов на английском языке.

Сервис «Главред» уделяет особое внимание очистке текста от слов не несущих смысла. В интерфейсных текстам к ним относится указание модальности, то есть отношения к действию: можете авторизоваться, должны завершить заказ, нужно пройти процедуру. Такие слова почти всегда можно удалить, а предложение переформулировать. Например вместо “Вы *должны* зарегистрироваться чтобы продолжить“ написать “зарегистрируйтесь чтобы продолжить“. Не нужно указывать очевидное: данный сайт, этот документ, меню ниже, на этой странице, форма внизу страницы, нажмите на кнопку, кликните здесь.

Фразы в пассивном залоге воспринимаются в среднем немного хуже чем в активном. Например “Вам будет отправлено письмо со ссылкой для входа“ стоит заменить на “Мы отправим вам письмо со ссылкой для входа“.

На веб-страницах стоит разделять текст для пользователей и текст для поисковых роботов. Первый должен быть лаконичным и лёгким для понимания, а второй должен способствовать поисковой оптимизации.

Вопросы

1. Что такое типографика?
2. Что такое удобочитаемость и различимость? Чем отличаются?

3. Приведите классификации шрифтов.
4. Что такое антиква и гротеск? Брусковый шрифт?
5. Чем отличается моноширинный шрифт от пропорционального? Где полезно использовать моноширинный шрифт?
6. Что такое акцидентный шрифт? В каких случаях его используют?
7. Какие шрифты называют нейтральными, а какие хара*к*терными
8. Что такое шрифт и гарнитура?
9. Какие бывают начертания?
10. Что такое переменный шрифт?
11. Опишите рекомендации по выбору шрифтов.
12. Что такое шрифтовая иерархия? Как её можно показывать?
13. Что такое сканирующее чтение?
14. Приведите рекомендации по интерфейсным текстам для форм.
15. Как стоит подписывать кнопки?
16. Приведите рекомендации для написания сообщений об ошибках.

Литература

- Бирман, И. Пользовательский интерфейс / И. Бирман. — Дизайн-бюро Артёма Горбунова, 2017. — 419 с.
- Ильяхов, М. Пиши, сокращай: Как создавать сильные тексты / М. Ильяхов, Л. Сарычева. — Альпина Паблишер, 2021. — 440 с
- Горбунов А. С. Типографика и вёрстка. — М.: Изд-во Бюро Горбунова, 2015. – 175 с.
- Шпикерманн, Э. О шрифте — Манн, Иванов и Фербер (МИФ), 2016. — 280 с.

Заключение

В первую очередь пособие призвано показать, что пользовательский интерфейс – важная часть практически любого программного продукта. Продукт должен быть *полезным*, *понятным* и *эффективным*. Это то, что пользователи могут напрямую оценить и то, что влияет не успешность продукта.

Чтобы разобраться, как создавать продукты, наделённые этими качествами, данного пособия недостаточно. Без разбора и анализа других продуктов, без собственной практики весь приведённый материал окажется бесполезен. Большую часть задач, возникающих в процессе дизайна интерфейса, удастся решить самостоятельно. Рекомендованная в конце каждого раздела литература поможет понять, какие из большой части реализованных решений были удачными и почему.

Но по-настоящему оценить созданный продукт смогут только пользователи. Поэтому практика по созданию любого продукта, результат которой не увидел ни один пользователь, окажется не сильно полезнее освоения исключительно теории. Потому что любой продукт создаётся для пользователя.

***

Замечания можно направлять на почту vetrovsv@outlook.com. Автор будет признателен за любые конструктивные предложения.
Последняя версия пособия:
github.com/ivtipm/HCI/releases/download/StudBook/StudBook.pdf

Библиографический список

1. Бирман, И. Пользовательский интерфейс / И. Бирман. – Москва: Дизайн-бюро Артёма Горбунова, 2017. – 419 с.
2. Буч, Г. Язык UML. Руководство пользователя / Г. Буч, Д. Рамбо, И. Якобсон; пер. с англ. Н. Мухин. – 2-е изд – Москва: ДМК Пресс, 2006. – 496 с.: ил.
3. Выготский Л. С. Психология / Л. С. Выготский. – М.: Эксмо-Пресс : АпрелЬ Пресс, 2000. – 1008 с.
4. Гарретт, Дж. Веб-дизайн: книга Джесса Гаррета. Элементы опыта взаимодействия / Дж. Гаррет; пер. с англ. С. Иноземцев. – Санкт-Петербург: Символ-Плюс, 2008. – 192 с.
5. Главред: [сайт]. – URL: glvrd.ru (дата обращения: 23.01.2022). – Текст: электронный.
6. Головач, В. В. Дизайн пользовательского интерфейса. Искусство мыть слона / В. В. Головач. – URL: https://leadmachine.ru/wp-content/uploads/2013/library/user_interface_design.pdf (дата обращения 28.01.2022). Текст: электронный.
7. Голомбински, К. Добавь воздуха! Основы визуального дизайна для графики, веба и мультимедиа / К. Голомбински, Р. Хаген. – Санкт-Петербург: Питер, 2013. – 272 с.
8. Горбунов А. С. Типографика и вёрстка / А. С. Горбунов. – Москва: Изд-во Бюро Горбунова, 2015. – 175 с.
9. ГОСТ Р МЭК 60447-2000. ИНТЕРФЕЙС ЧЕЛОВЕКО-МАШИННЫЙ. Принципы приведения в действие = Man-machine interface. Actuating principles: государственный стандарт Российской Федерации – URL: https://docs.cntd.ru/document/1200025203 (дата обращения: 21.01.2022). – Текст: электронный.
10. ГОСТ Р МЭК 60073-2000. Интерфейс человекомашинный. Маркировка и обозначения органов управления и контрольных устройств. Правила кодирования информации = Basic and safety principles for man-machine interface, marking and identification. Coding principles for indication devices and actuators: национальный стандарт Российской Федерации: утверждён и введён в действие постановлением Госстандарта России от 8 декабря 2000 г. № 348-ст: дата введения 2002-01-01 / разработан и внесен Техническим комитетом по стандартизации ТК 33 «Электротехника». – URL:

https://docs.cntd.ru/document/1200025202 (дата обращения: 21.01.2022). – Текст: электронный.

11. Дакетт, Д. HTML и CSS. Разработка и дизайн веб-сайтов / Д. Дакетт. – Москва: Эксмо, 2013. – 480 с.

12. Демарко, Т. Человеческий фактор: успешные проекты и команды / Т. Демарко, Т. Листер ; пер. с англ. М. Зислиса. – Санкт Петербург: СимволПлюс, 2005.

13. Ильяхов, М. Пиши, сокращай: Как создавать сильные тексты / М. Ильяхов, Л. Сарычева. – Альпина Паблишер, 2021. – 440 с.

14. Канеман, Д. Внимание и усилие /Д. Канеман; под ред. А. Н. Гусева. – Москва: Смысл, 2006. – 287 с.

15. Канеман, Д. Думай медленно… решай быстро / Д. Канеман. – Москва: АСТ, 2013. – 710 с.

16. Кесенбери, Е. Сторителлинг в проектировании интерфейсов. Как создавать истории, улучшающие дизайн / Е. Кесенбери, К. Брукс, – Москва: МИФ, 2013. – 310 с.

17. Круг, С. Не заставляйте меня думать / С. Круг. – Москва: Э, 2018. – 256 с.

18. Купер, А. Основы проектирования взаимодействия / А. Купер, М. Р. Рейманн; пер. с англ. М. Зислис. – Санкт-Петербург: Символ-Плюс, 2018. – 720 с.

19. Купер, А. Психбольница в руках пациентов. Алан Купер об интерфейсах / А. Купер. – Санкт-Петербург: Питер, 2018. – 384 с.

20. Мандел, Т. Разработка пользовательского интерфейса / Т. Мандел. – Москва: ДМК Пресс, 2007. – 418 с.

21. Машнин Т. Bootstrap: Быстрое создание современных сайтов / Т. Машнин. – Москва: Литрес, 2021. – 210 с.

22. Медведев, В.Ю. Роль дизайна в формировании культуры / В. Ю. Медведев. – Санкт-Петербург: СПГУТД, 2004. – 108 с.

23. Мильчин, А. Э. Методика редактирования теста / А. Э. Мильчин. – Изд. 3-е, перераб. и доп. – Москва: Логос, 2005. – 524 с.

24. Норман, Д. Дизайн привычных вещей, обновлённое и дополненное издание / Д. Норман. – Москва: МИФ, 2021. – 384 с.

25. Одегов, Ю. Г. Эргономика: учебник и практикум для академического бакалавриата / Ю. Г. Одегов, М. Н. Кулапов, В. Н. Сидорова. – Москва: Юрайт, 2018. – 157 с.

26. Орфограммка:[сайт]. – URL: orfogrammka.ru (дата обращения: 23.01.2022). – Текст : электронный.

27. Пользовательский интерфейс. – Текст: электронный // Бюро Горбунов: [сайт]. – URL: bureau.ru/books/ui/demo/4 (дата обращения: 20.08.2021).

28. Раскин, Д. Интерфейс: новые направления в проектировании компьютерных систем / Д. Раскин; пер. с англ. Ю. Асотов. – Санкт-Петербург: СимволПлюс, 2007. – 272 с.

29. Руководство по Figma. — Текст: электронный // designer: [сайт]. — URL: slashdesigner.ru/figma-guide (дата обращения: 26.01.2022).

30. Сакулин, С. А. Основы интернет-технологий: HTML, CSS, JavaScript, XML : учебное пособие / С. А. Сакулин. – Москва: МГТУ им. Н. Э. Баумана, 2017. – 112 с.

31. Унгер, Р. UX-дизайн. Практическое руководство по проектированию опыта взаимодействия / Р. Унгер, К. Чендлер. – Москва: Символ-Плюс, 2011. – 336 с.

32. Хабр: Принципы гештальта в дизайне пользовательского интерфейса. URL: https://habr.com/ru/company/cloud4y/blog/347444 (дата доступа: 13.09.2021. – Текст: электронный.

33. Хьюбел, Д. Глаз, мозг, зрение / Д. Хьюбел. – Москва: Мир, 2003. – 240 с.

34. Шпикерманн, Э. О шрифте / Э. Шпикерманн. – Москва: МИФ, 2016. – 280 с.

35. Applying Writing Guidelines to Web Pages. – Текст: электронный // Nielsen Norman Group: [сайт]. – URL: https://www.nngroup.com/articles/applying-writing-guidelinesweb-pages (дата обращения: 21.02.2022).

36. Jakobus, B. Mastering Bootstrap 4: Master the latest version of Bootstrap 4 to build highly customized responsive web apps / B. Jakobus. – 2nd Edition. – Birmingham: Packt Publishing Ltd, 2018. – 354 p.

37. Carroll, J. M. Paradox of the active user / J. M. Carroll, M. B. Rosson. – URL: https://www.semanticscholar.org/paper/Paradox-of-the-active-user-Carroll-Rosson/f560d34c1266abdea5a6fd2ddc11961995acf1a3 (дата обращения: 21.01.2022). – Текст: электронный.

38. Cooper, A. About Face: The Essentials of Interaction Design / A. Cooper. – Edition. – Noboken: Wily Publish ng, 2014 – 722 p.

39. Csikszentmihalyi, M. Flow: The Psychology of Optimal Experience / M. Csikszentmihalyi. – New York: Niesten Norman Group, 1990. – 336 p.

40. Design and code Windows apps: [сайт]. – URL: https://docs.microsoft.com/en-us/windows/apps/design/ (дата обращения: 21.02.2022). – Текст: электронный.

41. Fluent Design System: [сайт]. – URL: https://www.microsoft.com/design/fluent/ (дата обращения: 21.01.2022). – Текст: электронный

42. GNOME Human Interface Guidelines: [сайт]. – URL: https://developer.gnome.org/hig/ (дата обращения: 21.01.2022). – Текст: электронный.

43. Google Fonts: [сайт]. – URL: https://fonts.google.com/ (дата обращения: 15.09.2021). – Текст: электронный.

44. How to test your application with 5 users before you launch! – Текст: электронный // USERSNAP: [сайт]. – URL: https://usersnap.com/blog/user-tests-5-users/ (дата обращения: 28.03.2022).

45. Human Interface Guidelines: [сайт]. – URL: https://developer.apple.com/design/human-interface-guidelines/ (дата обращения: 21.07.2021).

46. KDE Human Interface Guidelines: [сайт]. – URL: https://develop.kde.org/hig/ (дата обращения: 21.01.2022). – Текст: электронный.

47. Norman, D. The Definition of User Experience (UX) / D. Norman, N. Jakob. – New York: Nielsen Norman Group, 2021. – 249 p.

48. MacKenzie, I. S. Fitts' law / I. S. MacKenzie // In K. L. Norman J. Kirakowski (Eds.), Handbook of human-computer interaction, 2018. P. 349–370.

49. Material Design Guidelines: [сайт]. – URL: https://material.io/design/guidelines-overview (дата обращения: 21.07.2021).

50. Mew K. Learning Material Design / K. Mew. – Biminham: Packt Publishing Ltd, 2015. – 186 p.

51. Miller, G. A. The Magical Number Seven, Plus or Minus Two / G. A. Miller // The Psychological Review. – 1956. – Vol. 63. – P. 81–97.

52. Paradigm. – Текст: электронный // Дизайн-система Mail.ru Paradigm: [сайт]. – URL: https://paradigm.mail.ru/ (дата обращения: 23.01.2022). – Текст: электронный.

53. Schneider, W. Controlled and automatic human information processing: Detection, search and attention / W. Schneider, R. M. Shiffrin // Psychol. Review. – 1977. – Vol. 84. – No. 1. – P. 1–66.

54. Systems and software engineering Vocabulary. – URL: https://www.cse.msu.edu/ cse435/Handouts/Standards/IEEE24765.pdf (датиа обращения: 21.01.22). – Текст: электронный.

55. Squire, L. R. The medial temporal lobe, the hippocampus, and the memory systems of the brain / L. R. Squire, B. J. Knowlton. – 2nd ed. – Cambridge; MA: MIT Press., 2000. P. 765–780.

56. Bootstrap 4 – Responsive Web Development / S. Moreto, M. Lambert, B. Jakobus, Jason Marah. – Birmingham: Packt Publishing, 2013 – 128 p.

57. The magical number seven, plus or minus two: Not relevant for design. – URL: https://www.edwardtufte.com/bboard/q-and-a-fetchmsg?msg_id=0000U6 (дата обращения: 07.11.2021).

58. The precipitous rise of Figma and fall of InVision: [сайт]. – URL: https://uxdesign.cc/theprecipitous-rise-of-figma-and-fall-of-invision-435f07e8d1b6 (дата обращения: 27.07.2021). – Текст: электронный.

59. Type Scale – A Visual Type Scale: [сайт]. – URL: https://type-scale.com/ (дата обращения: 11.10.2021). – Текст: электронный.

60. Typographic system with modular scale vertical rhythm. – Текст: электронный // Gridlover: [сайт]. – URL: https://www.grid-lover.net/try (дата обращения: 11.10.2021).

61. Tufte, E. R. Envisioning Information / E. R. Tufte. – New York: Graphics Press, 1990. – 126 p.

62. Tufte, E. R. Beautiful Evidence / E. R. Tufte. – New York: Graphics Press, 2006. – 213 p.

63. Tufte, E. R. The Visual Display of Quantitative Information / E. R. Tufte. – New York: Graphics Press, 2001. – 200 p.

64. Tufte, E. R. Visual Explanations: Images and Quantities, Evidence and Narrative / E. R. Tufte. – New York: Graphics Press, 1997. – 156 p.

65. Variable Fonts: [сайт]. – URL: v-fonts.com (дата обращения: 20.02.2022). – Текст: электронный.

66. Visualizing Fittss law. – Текст: электронный // Particletree: [сайт]. – URL: http://particletree.com/features/visualizing-fittss-law/ (дата обращения: 20.08.2021).

Предметный указатель

A/B testing, 121
A/B тестирование, 121

effect
    pop-out, 10

Figma, 91
Free Pascal, 36

GOMS, 138

HSV, 58

mockup, 88

RGB, 58

screen flow, 80

task flow, 80

use case diagram, 74
user experience, 64
user flow, 80
user interface, 28

wire flow, 80
wireframe, 84

автоматичные задачи, 13
архитектура информационная, 77
вайрфрейм, 84
внимание, 9
восприятие, 48
гарнитура шрифтовая, 156
гештальтпсихология, 48
диаграмма
    вариантов использования, 74
    потока, 79
    потоков задач, 80
    прецедентов, 74
    экранов, 80
дизайн, 44
дизайн-система, 44
закон
    Фиттса, 127
    Хика, 136
    близости, 49
интерфейс, 28
    жестовый, 30
    идеоматический, 40
    командной строки, 32
    материальный, 29
    метафорический, 38
    нейрокомпьютерный, 29
    текстовый, 32
интерфейс пользовательский, 28
интерференция внимания, 14
информативность интерфейса, 143
информационная
        производительность
        интерфейса, 144
квазирежим, 42
кернинг, 159
локус внимания, 11
магическое число 7 ± 2, 18
макет
    аннотированный, 91
метод
    карточной сортировки, 125
    персонажей, 70
модальность, 41
модель
    ментальная, 20, 121
    пользователя, 74
    представления, 23
    реализации, 21, 77
мокап, 88
навигация, 79
опыт взаимодействия, 64

ошибка модальная, 41
ошибки и промахи, 117
ощущение, 48
память кратковременная, 18
парадигмы интерфейса, 37
парадокс активного пользователя, 25
персонаж, 70, 123
пользовательская роль, 74
правило
  бесконечной границы, 129
  размера цели, 129
представление, 48
прерывание опыта, 11
принцип гештальта
  близость, 49
  замкнутость, 49
  общая зона, 49
  смежность, 49
  схожесть, 49
  целостность, 49
проклятие знания, 24
прототип пользовательского интерфейса, 88, 102

режим, 41
решётка Германа, 55
сетка, 86
символьная эффективность интерфейса, 148
сканирующее чтение, 163, 165
скевоморфизм, 38
состояние потока, 13
теория близости, 50
тестирование
  коридорное , 123
типографика, 151
треугольник Канижа, 49
уровни опыта взаимодействия, 66
шрифт, 156
  акцидентный, 154
  без засечек, 152
  брусковый, 152
  моноширинный, 155
  переменный, 159
  пропорциональный, 155
  с засечками, 152
шрифтовая иерархия, 163
шум визуальный, 55
эвристики Нильсона, 112
эффект выскакивания, 10, 61
юзабилити, 112
юзабилити-тестирование, 120