

# Человеко-машинное взаимодействие

Учебное пособие

Ветров С.В.

ЗабГУ  
2022

УДК 004.5  
ББК: 32.973  
ISBN \*\*\*\*\*

## Рецензенты

**Е. А. Михайлова**, к.т.н, доцент, заведующая кафедрой Информационных технологий и высшей математики, Читинского института (филиала) федерального государственного бюджетного образовательного учреждения высшего образования «Байкальский государственный университет».

**Н. В. Пешков**, к.ф-м.н., доцент, заведующий кафедрой Прикладной механики и математики, Забайкальского института железнодорожного транспорта – филиала Федерального государственного бюджетного образовательного учреждения высшего образования «Иркутский государственный университет путей сообщения» в г. Чите.

## Ветров, Сергей Владимирович

Человеко-машинное взаимодействие / С. В. Ветров ; Забайкальский государственный университет. – Чита : ЗабГУ, 2022. – 184 с.

Учебное пособие предназначено для учебно-методической поддержки дисциплины «Человеко-машинное взаимодействие».

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

УДК 004.5  
ББК: 32.973  
ISBN \*\*\*\*\*

# Оглавление

<b>Введение</b>	<b>7</b>
<b>1 Человек</b>	<b>11</b>
1.1 Эргономика и когнитивная психология . . . . .	11
1.2 Внимание . . . . .	11
1.2.1 Привлечение внимания . . . . .	13
1.2.2 Локус внимания. . . . .	14
1.2.3 Состояние потока. . . . .	15
1.3 Автоматичные задачи . . . . .	16
1.4 Память . . . . .	20
1.5 Ментальные модели . . . . .	22
<b>2 Пользовательский интерфейс</b>	<b>31</b>
2.1 Понятие интерфейса . . . . .	31
2.2 Классификация видов ПИ . . . . .	32
2.3 Парадигмы интерфейса . . . . .	38
2.4 Модальность . . . . .	41
<b>3 Дизайн и восприятие</b>	<b>45</b>
3.1 Дизайн . . . . .	45
3.2 Восприятие . . . . .	49
3.3 Цвет . . . . .	54
<b>4 Проектирование UX</b>	<b>65</b>
4.1 Проектирование пользовательского интерфейса . . .	65
4.2 Проектирование UX . . . . .	67
4.2.1 Уровни UX . . . . .	67
4.2.2 Уровень стратегии . . . . .	69

4.2.3	Метод персонажей . . . . .	71
4.2.4	Диаграмма прецедентов . . . . .	74
4.2.5	Уровень структуры . . . . .	76
4.2.6	Дизайн навигации и диаграммы потоков . . . . .	77
4.2.7	Уровень компоновки . . . . .	83
4.2.8	Уровень поверхности . . . . .	89
4.2.9	Figma . . . . .	92
4.3	Методология проектирования . . . . .	107
<b>5</b>	<b>Юзабилити и тестирование</b>	<b>111</b>
5.1	Юзабилити . . . . .	111
5.2	Юзабилити-тестирование . . . . .	119
5.2.1	Понятие юзабилити-тестирования . . . . .	119
5.2.2	A/B-тестирование . . . . .	120
5.2.3	Тестирование с наблюдением . . . . .	121
5.2.4	Метод карточной сортировки . . . . .	122
<b>6</b>	<b>Анализ интерфейса</b>	<b>125</b>
6.1	Количественная оценка пользовательского интерфейса	125
6.2	Закон Фиттса . . . . .	125
6.3	Закон Хика и проблемы выбора . . . . .	133
6.4	GOMS . . . . .	135
6.5	Информационная эффективность интерфейса . . . . .	140
6.5.1	Информативность интерфейса . . . . .	140
6.5.2	Оценка информативности . . . . .	141
<b>7</b>	<b>Типографика и тексты</b>	<b>147</b>
7.1	Понятие типографики . . . . .	147
7.2	Классификация шрифтов . . . . .	148
7.3	Основные понятия из типографики . . . . .	153
7.3.1	Выбор шрифта . . . . .	156
7.3.2	Шрифтовая иерархия . . . . .	160
7.3.3	Выравнивание текста и длина строки. . . . .	160
7.4	Текст и синтаксис интерфейса . . . . .	161

<b>Заключение</b>	<b>173</b>
<b>Литература</b>	<b>175</b>
<b>Предметный указатель</b>	<b>182</b>



# Введение

Учебное пособие предназначено для учебно-методической поддержки дисциплины «Человеко-машина взаимодействие». В пособии освещены все основные темы рабочей программы, оно призвано кратко осветить основные вопросы создания пользовательских интерфейсов. Может служить отправной точкой для углублённого изучения профессиональной литературы.

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. Интерфейс не учитывал в полной мере, что будет использован в сложной боевой обстановке, во время утомительного дежурства. Незаметно переназначенные программой идентификаторы целей привели к катастрофе.

Пользовательские интерфейсы могут быть настолько плохи, что становятся причинами катастрофы. Но гораздо чаще, интерфейсы "всего лишь" снижают продуктивность пользователей, приводят

к потере данных, вызывают неудовлетворённость работой, мешают электронной коммерции, делают продукт неконкурентоспособным.

Мы до сих пор сталкиваемся с проблемами в интерфейсах в своих повседневных дела. Читатели, наверняка могут привести много примеров, когда им не удалось совладать с программой, решая несложную задачу, удавалось допустить явные ошибки в работе, которые не были заметны до самого конца или приходилось тратить много времени на рутинную работу, которая, казалось бы, должна быть давно поручена компьютерам.

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

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

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

Даже если над продуктом работали отличные архитекторы, программисты и тестировщики продукт будет может стать нежизне-

способным, 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 (см. пример в параграфе 5.2.2).

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*<sup>1</sup> 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 объективно оценить качество 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.

## Вопросы

### 1. Почему важно уделять внимание интерфейсам?

---

<sup>1</sup>Под продуктом будем понимать программу, безразлично для какого устройства, сайт, веб-приложение.

2. Какие проблемы возникают при проектировании интерфейса?
3. С какими проблемами в пользовательском интерфейсе вы сталкивались?
4. В чём состоит важность качественного пользовательского интерфейса в сравнении с другими показателями качества программного продукта?
5. Почему программисты склонны создавать плохие пользовательские интерфейсы?
6. Часто ли программы не позволяли вам упростить рутинную работу?
7. Часто ли вы сталкивались с программами, которые были больше *помощником* чем инструментом?

# 1 Человек

## 1.1 Эргономика и когнитивная психология

Руководства по разработке продуктов, взаимодействующих с человеком физически, обычно содержат информацию, основанную на свойствах и возможностях опорно-двигательного аппарата и органов чувств. 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.

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

## 1.2 Внимание

**Внимание** – избирательная направленность восприятия на тот или иной объект.

Д. Канеман предложил считать внимание ресурсом. Любая относительно сложная задача или отвлекающий фактор "расходуют" часть внимания человека [4]. 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 как можно быстрее мог снова вернуться к работе и допускал как можно меньше ошибок из-за невнимательности.

В том числе поэтому, покупая билет на поезд на сайте железнодорожной компании, мы чаще всего заполняем не все необходимые данные сразу, а поэтапно 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 и избирательно. 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.

### 1.2.1 Привлечение внимания

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

**Эффект выскакивания** помогает сократить время обнаружения элемента интерфейса [63]. Эффект выскакивания – это независимость скорости поиска целевой стимула (отличного от множества одинаковых) 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.

Движение – ещё один быстрый способ привлечь внимание пользователя. Однако его можно использовать в относительно узком круге сценариев.

**Прерывание опыта** – еще один способ привлечь внимание пользователей, особенно когда это делается во время рутинной задачи. Если есть важное объявление или нужно, чтобы пользователи ознакомились с какой-то информацией, то отображение этой информации и просьба ознакомиться с ней может быть эффективной техникой (рис. 1.2). Но пользователю обычно потом требуется

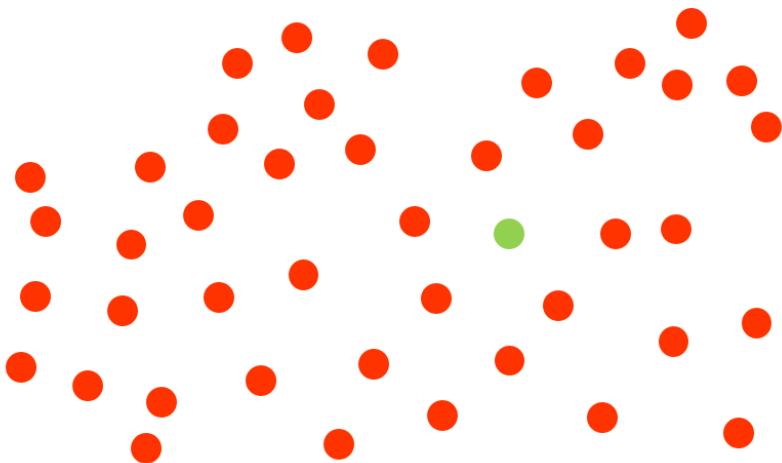


Рисунок 1.1. Благодаря эффекту выскакивания люди быстро и автоматически находят красный круг среди множества других кругов одного цвета.

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.

### 1.2.2 Локус внимания.

Идея или предмет, на котором сосредоточено внимание, называют **локусом<sup>1</sup> внимания** (locus of attention) [3].

Локус внимания только один. Поэтому человек может не замечать то, что находится вне локуса внимания. Например, не обращать внимания на раскладку клавиатуры, когда в локусе его внимания находится кодирование алгоритма.

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

---

<sup>1</sup>лат. locus – место – место, область

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

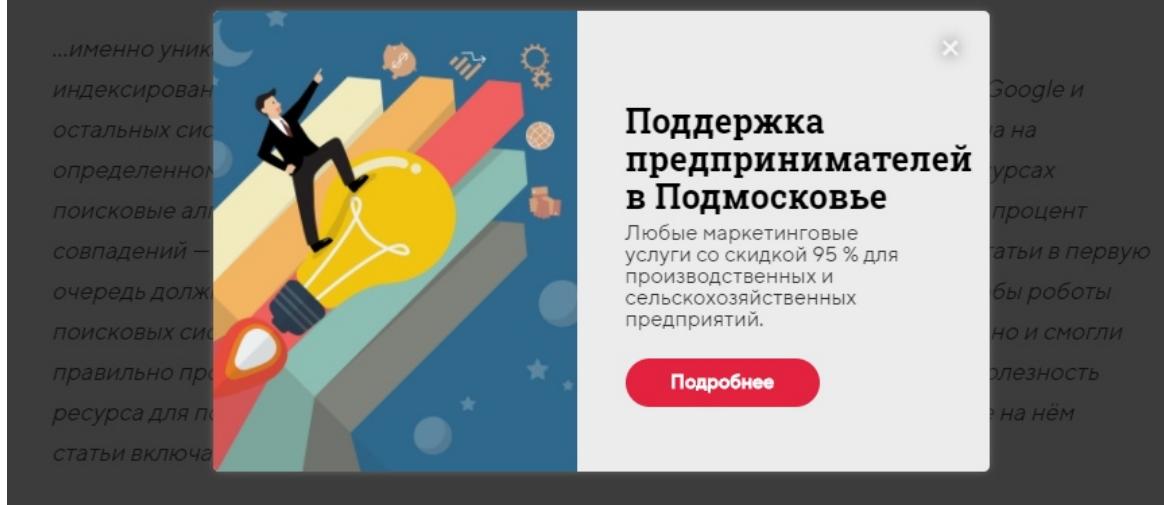


Рисунок 1.2. Всплывающее во время просмотра страницы – это *прерывание опыта* – способ привлечь внимание.

Локус может измениться неожиданно. Зазвонил телефон и в локусе внимания 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. А мысленные усилия на такие переключения, создают ложное ощущение, что проделан большой объём интеллектуальной работы.

### 1.2.3 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 людей более продуктивными стоит проектировать продукты, вызывающие и поддерживающие состояние потока, и прикладывать все возможные усилия, чтобы избежать любого по-

ведения, потенциально способного разрушить поток. Если программа выбивает пользователя из потока, ему трудно возвращаться в это продуктивное состояние [71].

### 1.3 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 получаются, если человек об этом не задумывается. Но как только задумается, то может 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

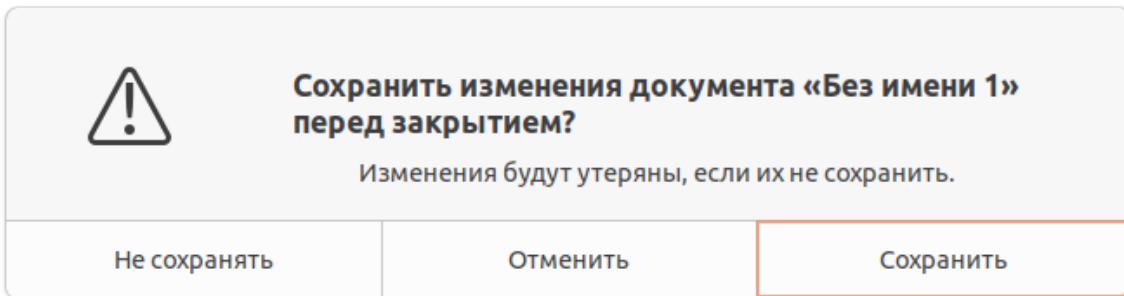


Рисунок 1.3. Если дизайн всех диалоговых окон содержит одинаковый порядок кнопок, то пользователю будет легко сформировать привычку нажимать на нужную кнопку практически не глядя после загрузки ОС, то в определённый момент замечает: вводя пароль легче ошибиться если начать думать о нём во время ввода.

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

Пользователи запоминают расположение часто используемых кнопок (например как на рис. 1.3, элементов меню и могут ими пользоваться практически не читая надписи на этих кнопках и пунктах меню. 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 поменили местами. По той же причине упорядочивать элементы меню в программе в зависимости от частоты их использования плохая идея.

С другой стороны, интерфейс может способствовать формированию вредных пользовательских привычек. Злоупотребление всплы-

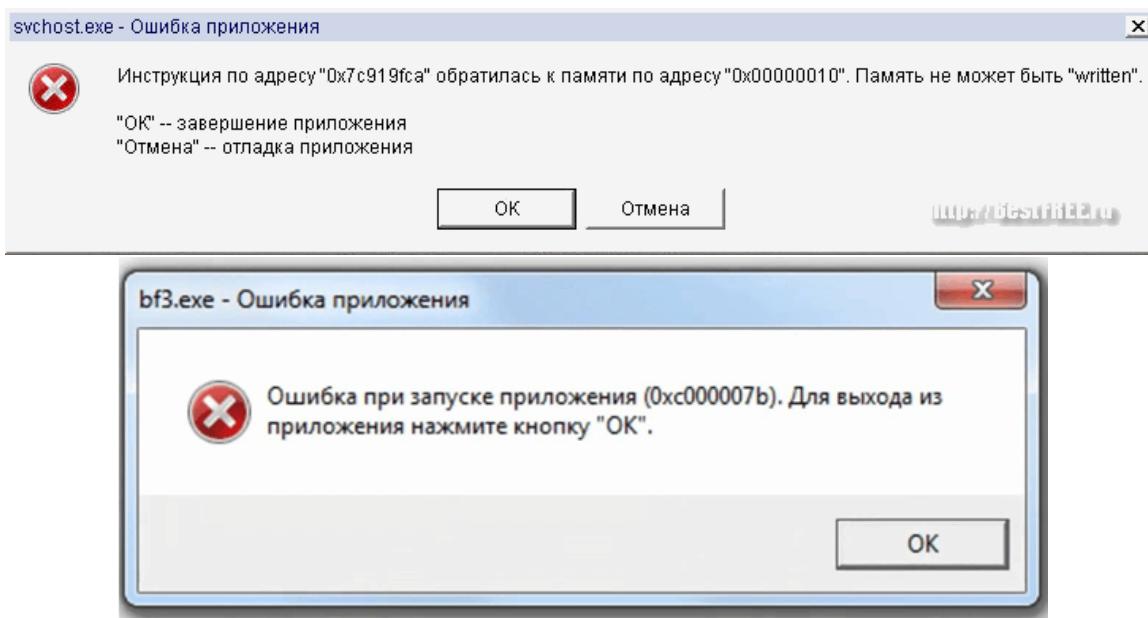


Рисунок 1.4. Злоупотребление диалоговыми окнами с бесполезными для пользователя сообщениями часто приводит к формированию привычки закрывать любые диалоговые окна не вникая в содержимое

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

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



Рисунок 1.5. Приложение Фейсбук для iOS и Андроид подстраивается под привычки пользователей этих операционных систем располагая панель с вкладками так, как принято в данной ОС.

Привычки трудно менять. Поэтому человеку бывает трудно, переключится на использование новой клавиатуры или новой, сильно изменившей интерфейс, программы. Это может удержать пользователя от перехода на продукт конкурентов, или наоборот помочь перейти на ваш, если старые привычки пользователей окажутся учтены в новой программе. Многие программисты имеют привычку нажимать комбинацию клавиш  $Ctrl+S$  для сохранения файла исходного кода после каждого логически завершённого изменения. Абсолютное большинство программ интерпретируют эту комбинацию клавиш одинаковым образом.

Другая польза автоматических задач – они не отвлекают для поль-

зователя и он может выполнять несложные манипуляции не отвлекаясь на интерфейс программы.

## 1.4 Память

Человеческую память можно разделить на два вида: долговременную и кратковременную [59]. 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.

Эксперимента Джорджа Миллера показали, что кратковременная память человека способна запоминать в среднем восемь десятичных цифр, девять двоичных цифр, семь букв алфавита и пять односложных слов [60]. Все запоминаемые сущности не были связаны друг с другом по смыслу. Выявленная закономерность получила название "Магическое число семь плюс-минус два".

Это правило широко освещается в интернет-публикациях по проектированию интерфейсов. Однако опыт показывает прямое применение этого правила, например для скрытия редко используемых пунктов меню, ограничения числа элементов в списках и т.п. не приводит к повышению продуктивности пользователя [7, 62].

Продолжительность хранения информации (при условии, что нет повторения) около 20 сек. После 30 сек. след информации становится настолько хрупким, что даже минимальная *интерференция* разрушает его. Это ещё одна причины учитывать смену локуса внимания пользователя во время использования программы. Чтобы вернуться к работе пользователю нужно заново погрузиться в решаемую задачу (рис. 1.6), желательно как можно быстрее и с меньшими усилиями.

Сохранение не до конца оформленного заказа в интернет-магазине,

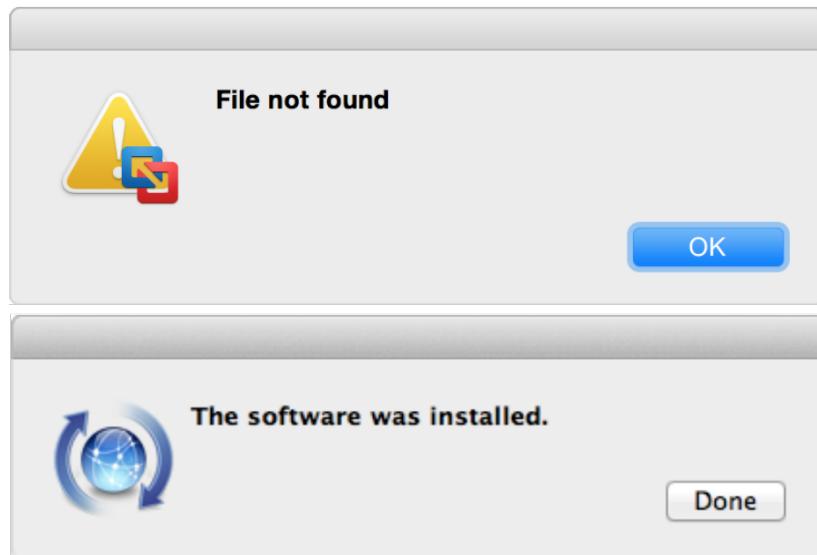


Рисунок 1.6. Программа не должна заставлять пользователя держать в голове информацию, если она сама может её легко хранить и отображать. Если человек отвлёкся, то ему придётся вспоминать о каком файле (верхнее окно) и о какой программе (нижнее окно) ему сообщают [7].

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 сначала предлагает выбрать вид конвертируемых единиц (длина, масса, объём, валюта), потом выбрать исходную валюту (рубли из большого списка валют), выбрать целевую валюту (снова из большого списка) и только потом предложить записать значение.

Программа не должна заставлять пользователя держать в голове слишком много информации. С учётом того, что пользователь и так в данный момент может совершать сложную интеллектуальную работу. Это же и касается случаев, когда пользователю нужно сделать выбор – удержать в голове большое число вариантов довольно трудно



Рисунок 1.7. 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 [7].

(смотрите также параграф 6.3 о законе Хика). Поэтому в отдельных случаях можно либо сразу сделать выбор за пользователя либо предлагать ему небольшое количество готовых вариантов.

## 1.5 Ментальные модели

**Ментальные модели**<sup>2</sup> представляют понимание человека как нечто функционирует [1].

Примеры ментальных моделей: способ завязывания шнурков, способ завести автомобиль; понимание того, что будет если зажать кнопку выключения телефона; местонахождение ближайшей кофейни; способ использования дрели.

Человек учится использовать объект, основываясь на наблюдениях и опыте использования (рис. 1.11). Мы строим ментальные модели основываясь на объекте (иногда достаточно просто посмотреть на него), читая руководство пользователя или спрашивая дру-

<sup>2</sup>Модель – это упрощённое представление действительного объекта и/или протекающих в нём процессов



Рисунок 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?

гих 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. 1.10). Если представлять, что внутри фотоаппарата сидит чёртик, способный рисовать увиденное за миллисекунды, то это не обязательно помешает фотоаппаратом пользоваться.

**Модель реализации** описывает *фактическое устройство* чего либо.

1. Download

2. [Download](#)

3.

Download

Рисунок 1.9. Какие ментальные модели относительно этих трёх элементов у вас возникают?



Рисунок 1.10. Модель реализации и 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. Таким модели, предлагаемые дизайнером, называются **моделями представления** или **моделями дизайнера**.

Модели модели могут и не должны 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 анимация открытия этой программы показывает 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

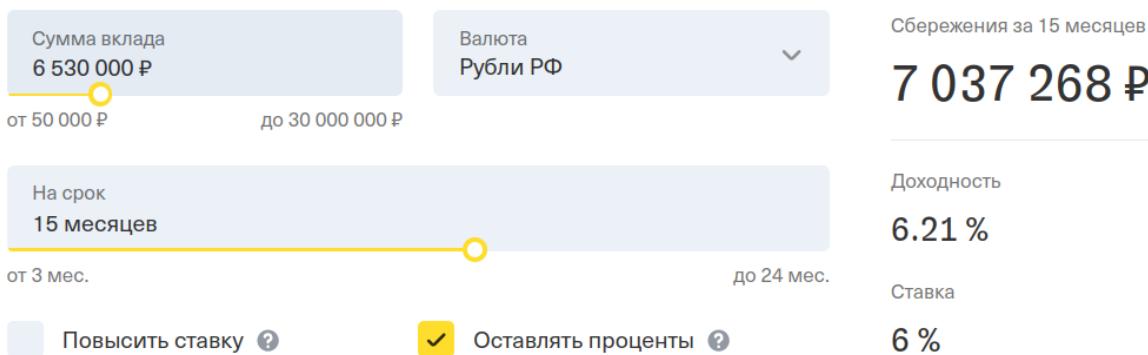


Рисунок 1.11. Вместо сложной таблицы, где перечислены все возможные ставки и сроки вклада на сайте банка есть интерактивный калькулятор. С ним легко экспериментировать меняя ползунками срок и суммы вклада, итоговая сумма рассчитывается автоматически при изменении параметров. Это помогает пользователю понять как устроен вклад - т.е. сформировать ментальную модель

она уже соответствует ментальной модели пользователя. Программы обычно имеют довольно сложное устройство, и принципы их построения скорее всего не знакомы пользователю. Поэтому, модель представления не должна полностью повторять модель реализации.

**Проклятие знания** – когнитивное искажение, более информированным людям чрезвычайно сложно рассматривать какую-либо проблему с точки зрения менее информированных.

Разработчику программы трудно представить, какие затруднения может вызывать программа 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 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 Ресон 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 [68].

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

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

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

Разработчику не стоит 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 все необходимую информацию, быстро и точно выполнять все действия, решая собственные задачи в

программе и заниматься её обслуживанием. 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, берегать пользователя от совершения ошибок, а если он их совершил, то давать возможность исправить минимальными усилиями. При решении сложных задач, программа должна помогать пользователю освоить новые ментальные модели и полагаться на имеющиеся, использовать способность человека к формированию привычек для ускорения взаимодействия с программой.

## Вопросы

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

12. Что такое проклятие знания? Как оно мешает программисту разрабатывать интерфейс?
13. Как *должны* соотносится ментальная модель и модель представления?
14. Что такое парадокс активного пользователя?

## Литература

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



## 2 Пользовательский интерфейс

### 2.1 Понятие интерфейса

**Интерфейс** – граница между двумя функциональными объектами, требования к которой определяются стандартом; совокупность средств, методов и правил взаимодействия (управления, контроля и т. д.) между элементами системы [15].

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-человеком и программно-аппаратными компонентами компьютерной системы<sup>1</sup>.

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.

---

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

## 2.2 Классификация видов ПИ

Приведём классификацию видов пользовательского интерфейса.

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

Зачастую, конкретный ПИ является комбинацией нескольких видов интерфейса.

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

**Материальный ПИ** (tangible user interface, TUI) – это разновидность интерфейса пользователя, в котором взаимодействие человека с электронными устройствами происходит при помощи материальных предметов и конструкций.

**Жестовый ПИ** используется в устройствах с сенсорными экранами – мобильных устройствах, интерактивных панелях, где основные жесты – это касание (тап), свайп, поворот и раздвигание

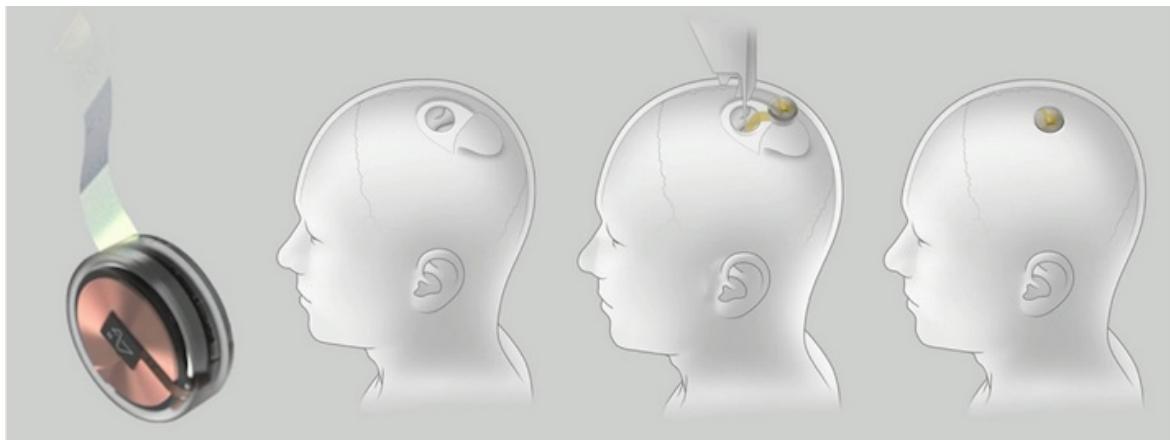


Рисунок 2.1. Имплантируемый нейрокомпьютерный интерфейс Neuralink.



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



Рисунок 2.3. SandScape – виртуальный ландшафт и его объекты, взаимодействуют с пользователем, формирующим этот ландшафт с помощью песка. Демонстрация: [vimeo.com/44538789/](https://vimeo.com/44538789/)

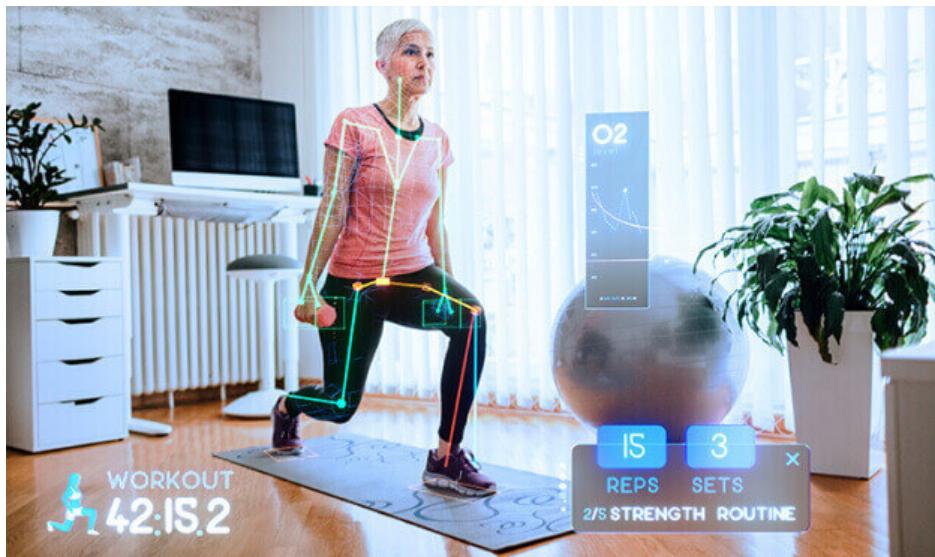


Рисунок 2.4. Azure Kinect DK. Система компьютерного зрения распознаёт положение тела. Демонстрация: [youtube.com/watch?v=OoJWmmOlws8](https://youtube.com/watch?v=OoJWmmOlws8)

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.) или распознавать жесты с видео (Microsoft Kinect, TrackIR, Azure Kinect DK и др.).

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



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

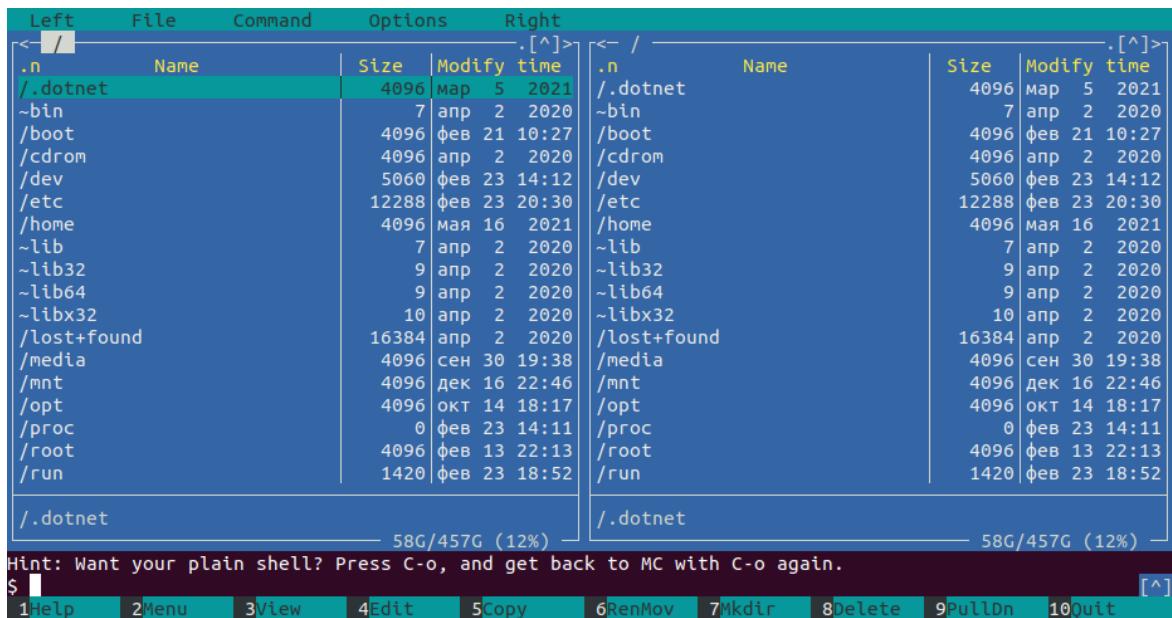


Рисунок 2.6. GNU Midnight Commander – файловый менеджер с текстовым ПИ.

**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: легко автоматизировать действия, он не требует специальных вычислительных ресурсов. Возможность автоматизации можно рассматривать как дополнительную функциональность, 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<sup>2</sup> чем с ГИП, особенно если учесть возможность автомати-

<sup>2</sup>для оценки времени на совершение действий см параграф про GOMS

The screenshot shows a terminal window with the following session:

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

Рисунок 2.7. ZSH - командная оболочка для UNIX, с гибким механизмом автодополнения команд, исправления опечаток, цветовым кодированием состояний репозиториев git.

ческого дополнения команд.

Но такой тип интерфейса требует от пользователя многих ментальных моделей – знания команд, из параметров и принципов работы с ними. Эти ментальные модели сложнее формируются (плохая изучаемость, learnability<sup>3</sup>), 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.

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

<sup>3</sup>см. параграф о критериях качества интерфейса

нил текстовый, но не вытеснил до конца. Здесь нет необходимости знать текстовые команды, название многих утилит их аргументов и параметров. К тому же эти команду нужно вводить 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 интерфейса.

## 2.3 Парадигмы интерфейса

Существуют три основных парадигмы интерфейса [10]:

- 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 модель. 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 точ-

ным 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) 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 скевоморфной графике показан объём предметов: свет, тени, блики и текстуры.

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 изу-



Рисунок 2.8. 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.



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

чении 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. Но пользователь легко может понять принцип её работы. Это - идiomатическое 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 [7].

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. Передача 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.



Рисунок 2.10. В графическом редакторе режим определяет форму курсора. В iOS в режиме редактирования домашнего экрана иконки подрагивают [7].

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.

Модальная ошибка привела к крушению самолёта рейса 214 «Азиан-эйрлайнс» в июле 2013 года. Пилот не осознавал, что в текущем режиме автопилота не работает автомат тяги, и продолжал управлять самолётом, не следя за скоростью [7].

Режим часто не является лоукосом 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 (рис. 2.10)

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

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

Кнопка Шифт включает квазирежим ввода прописных букв — бук-

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

## **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 Дизайн и восприятие

## 3.1 Дизайн

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 [16].

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

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

Дизайн не обязательно предполагает статичность. Дизайн – это то не то, как предмет выглядит, а то, как он работает<sup>1</sup>. Дизайн взаимодействия с пользователем – отдельная отрасль дизайна хороший тому пример. Модель представления – тоже объект дизайна.

**Дизайн-система** – набор компонентов, правил, предписаний инструментов для повышения качества и скорости разработки про-

---

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

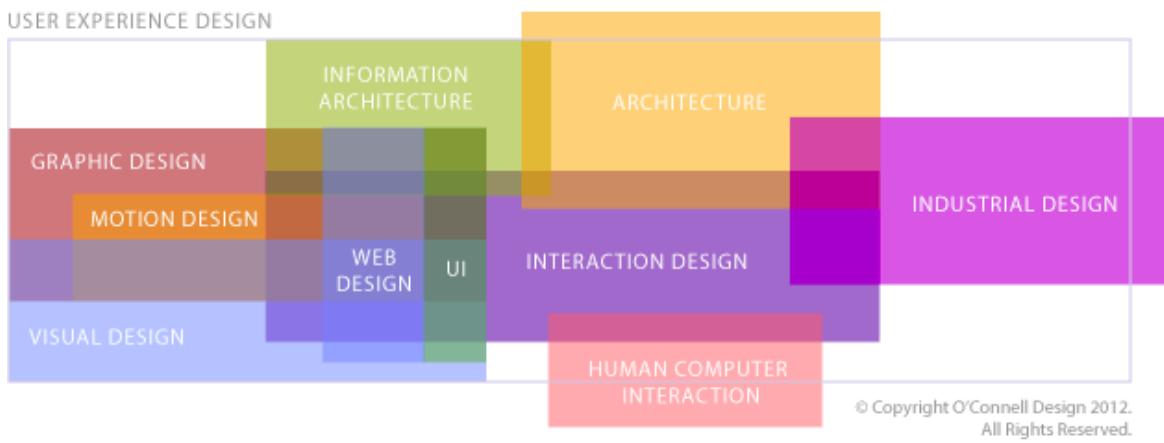


Рисунок 3.1. Отрасли дизайна

дуктов, 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. 3.4);
- Библиотеку готовых компонентов (UI-кит), шаблоны;
- Наборы UX-паттернов (как организовывать навигацию, диалоговые окна, формы и т.д.);
- Документацию, правила, рекомендации (например material design [33], Apple [32]).

Некоторые компании публикуют свои дизайн-системы, например mail.ru (рис. 3.2). Готовые наборы базовых элементов интерфейса помогают ускорить разработку макетов. Многие дизайнеры публикуют такие наборы, например их можно найти на сайте [figma.com/community](https://figma.com/community).

Дизайн-система призвана систематизировать работу дизайнера, в том числе с помощью правил. Однако в Интернете приводится большое число рекомендаций относительно дизайна вообще и визуального дизайна в частности. Стоит относиться к ним в долей скепсиса и рассматривать в лучшем случае как лучшую практику (best practice), так как большинство материалов не содержат ссыл-

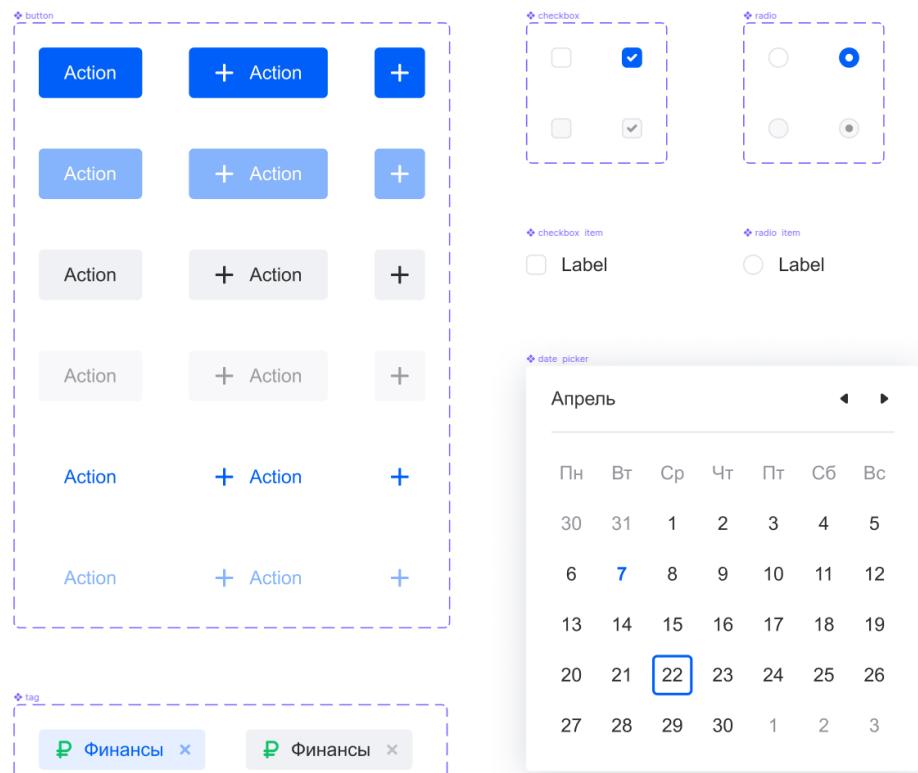


Рисунок 3.2. Часть библиотеки готовых компонентов дизайн-системы Paradigm (mail.ru) в онлайн-сервисе для создания макетов интерфейсов Figma.

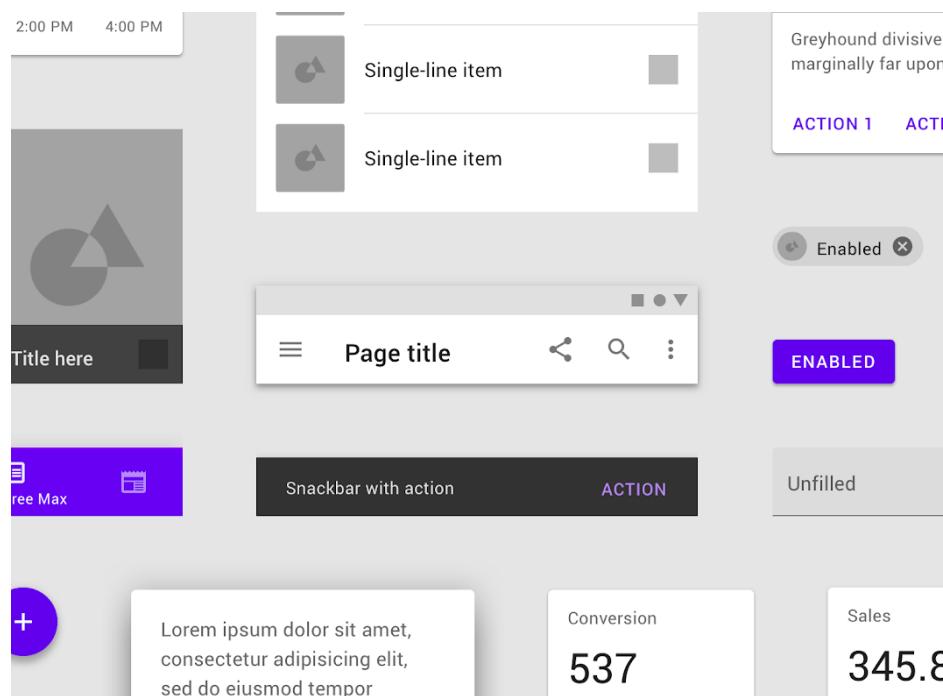


Рисунок 3.3. Часть библиотеки готовых компонентов дизайн-системы Material Design.

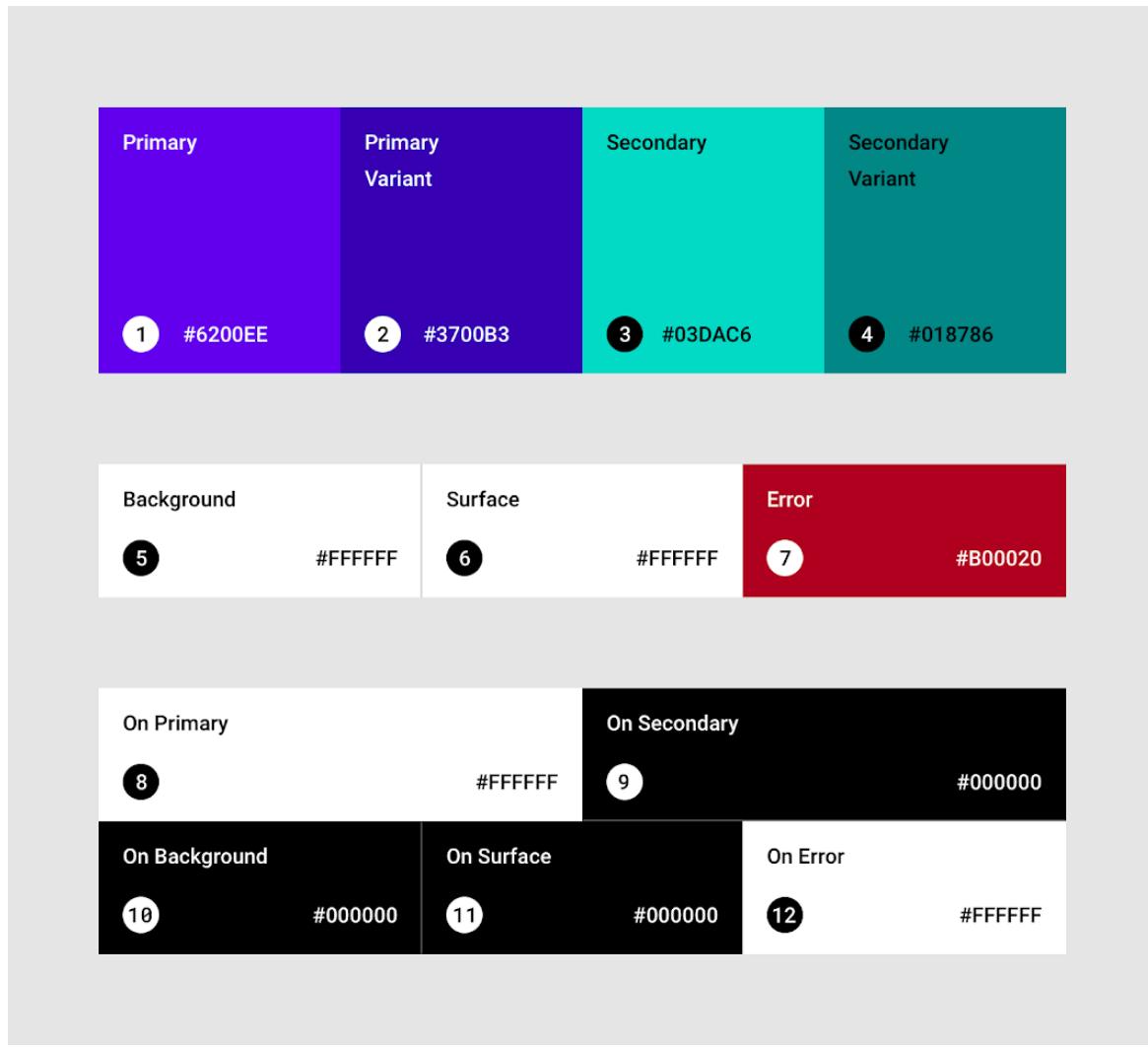


Рисунок 3.4. Цветовая схема рекомендованная Material Design

ки 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 Восприятие

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

- ощущение – отражение отдельных свойств и предмета;
- восприятие – 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 объясняющего принципа гештальтпсихология выдвигает принцип целостности.

Первичными данными психологии являются целостные структуры – гештальты.

Примером противоположной работы восприятия может служит расстройство восприятия – предметная агнозия: человек видит предметы как сумму отдельных частей, но не может составить целостный образ.

Целостность восприятия и его упорядоченность достигаются благодаря следующим принципам:

- Близость (Law of Proximity) – стимулы, расположенные рядом,

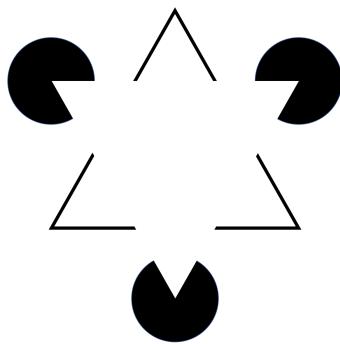


Рисунок 3.5. Треугольник Канижа. На изображении нет четко отчерченного белого треугольника, но человеческое восприятие достраивает его.

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

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

Дilemma делать ли подпись к картинке снизу или сверху, име-

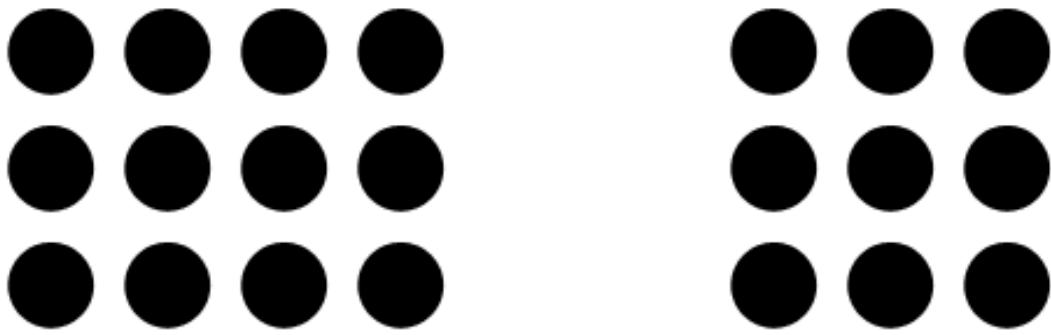


Рисунок 3.6. Принцип близости – близко расположенные объекты воспринимается связанными по смыслу [65].



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

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

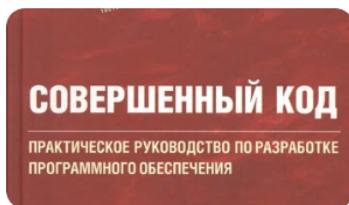
YouTube · 22 декабря 2021



### **1. Проектирование интерфейсов. Введение | Технострим**

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

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



### **Совершенный код Практическое руководство. 1 4...**

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

[chitai-gorod.ru](http://chitai-gorod.ru) · реклама | 16+



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

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

[plus.yandex.ru](http://plus.yandex.ru) · реклама

Рисунок 3.7. Принцип схожести: реклама в поисковой выдаче выглядит похожей на сами результаты поисковой выдачи.

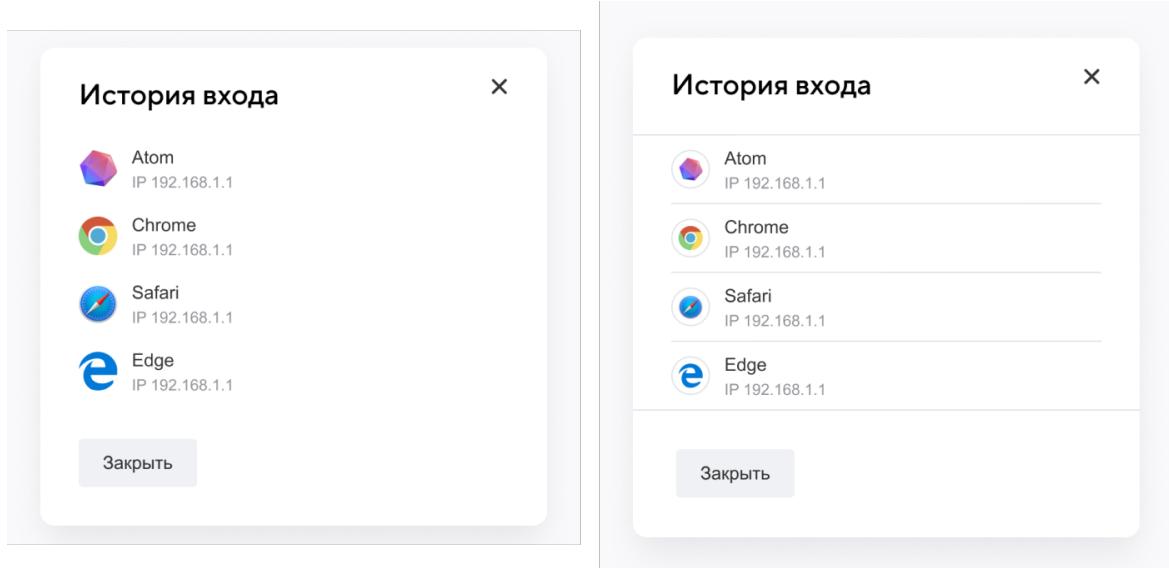


Рисунок 3.8. Отделение объектов друг от друга с помощью только пустого пространства лучше, чем с помощью разделителей. Они могут восприниматься как визуальный шум и способствовать обратной задаче – объединения разных групп объектов. Изображение из [31]

ет 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).

**Правило:** внутренние расстояния должны быть меньше внешних – следствие закона близости [17].

За счёт использования принципов непрерывности и близости можно избавиться от лишних графических элементов: рисунки 3.14 и 3.15. Колонки таблицы были выровнены и хорошо определяются без разграничающих линий, отделяемые друг от друга пустотой (расстоянием)<sup>2</sup>

<sup>2</sup>Анимация пошагового улучшения таблицы:  
[raw.githubusercontent.com/ivtipm/HCI/master/etc/88a654c7e364453f8a17481a6da9dda4.gif](http://raw.githubusercontent.com/ivtipm/HCI/master/etc/88a654c7e364453f8a17481a6da9dda4.gif)

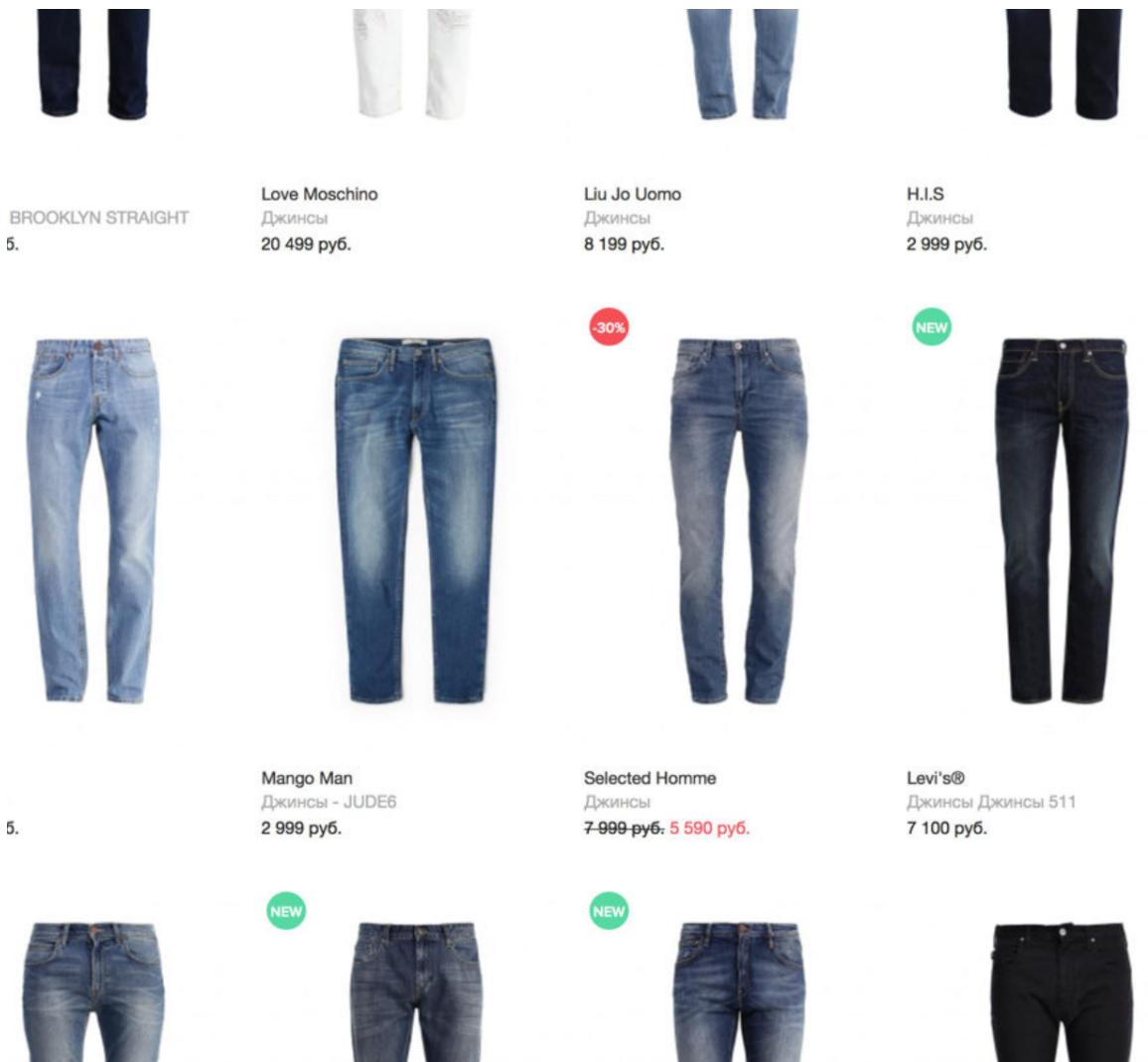


Рисунок 3.9. Дизайнер не использовал принцип близости: нельзя понять к каким изображениями относятся подписи. Нужно сделать подписи ближе к связанным изображениям.

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

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

Рисунок 3.10. Следствие принципа близости: внутренние расстояния (между буквами в слове) должны быть меньше внешних (между словами)

МОСКОВСКИЙ  
МЕТРОПОЛИТЕН  
ИМЕНИ В.И.ЛЕНИНА

МОСКОВСКИЙ  
МЕТРОПОЛИТЕН  
ИМЕНИ В.И.ЛЕНИНА

Рисунок 3.11. Верху: принцип "внутреннее  $\leq$  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

Восприятие способно не только дополнять увиденное, но и исказять. 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, 3.17).

### 3.3 Цвет

Модель RGB (red, green, blue — красный, зелёный, синий) или КЗС — аддитивная<sup>3</sup> цветовая модель (рис. 3.19), описывающая способ кодирования цвета для цветовоспроизведения с помощью трёх цветов, которые принято называть основными. Выбор основных цветов обусловлен особенностями физиологии восприятия цвета сетчаткой человеческого глаза.

Интенсивность каждого цвета представляется в виде одного ок-

<sup>3</sup>аддитивная модуль — модуль цвета получаются путём добавления к чёрному цвету; отсутствие излучения — нет никакого цвета — чёрный, смешение всех трёх в определённой пропорции — даёт белый

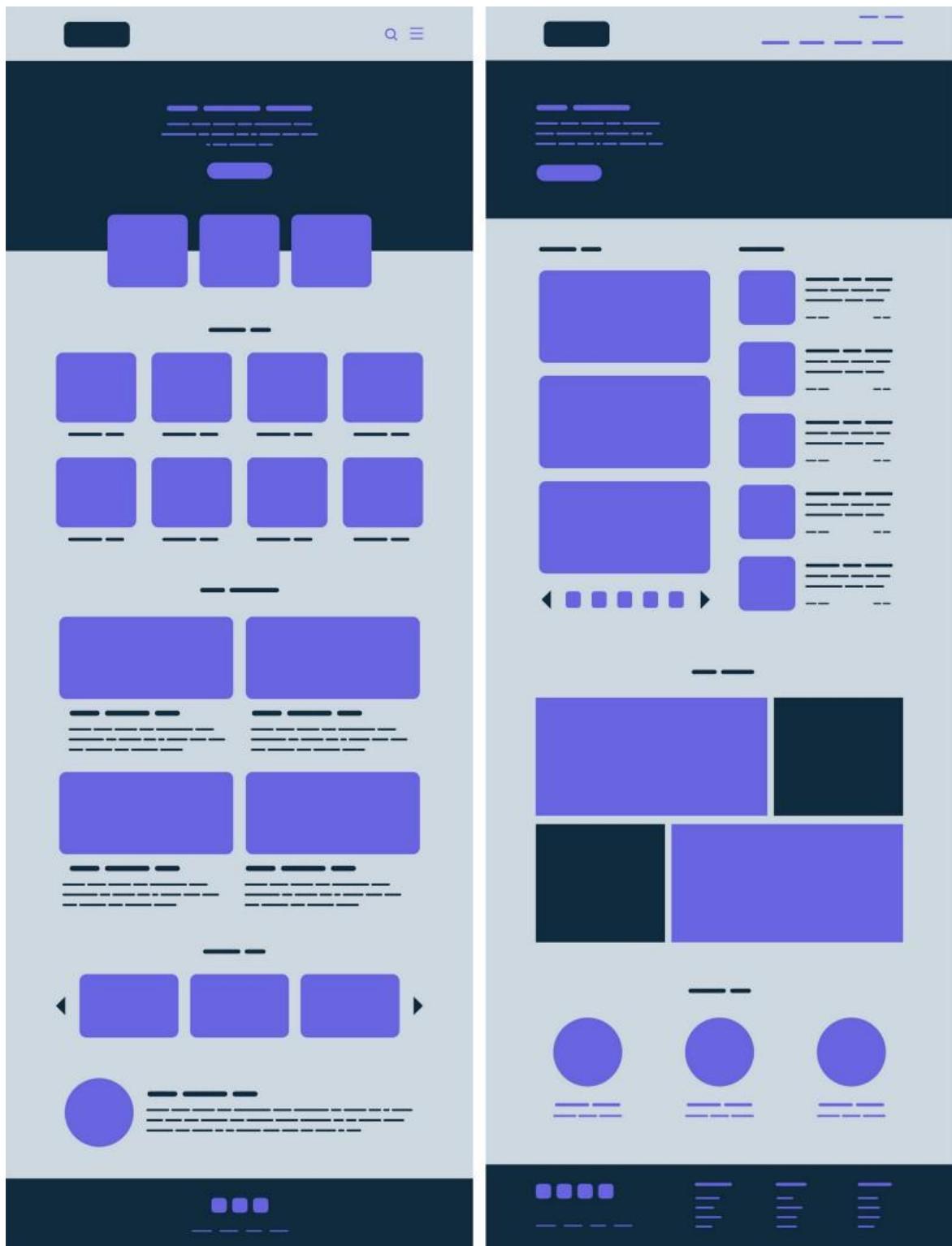


Рисунок 3.12. Принцип близости. Большое значение имеет пустота, воздух – он отделяет объекты друг от друга. Расстояние между отдельными группами элементов – большое, между элементами в группе – меньше.

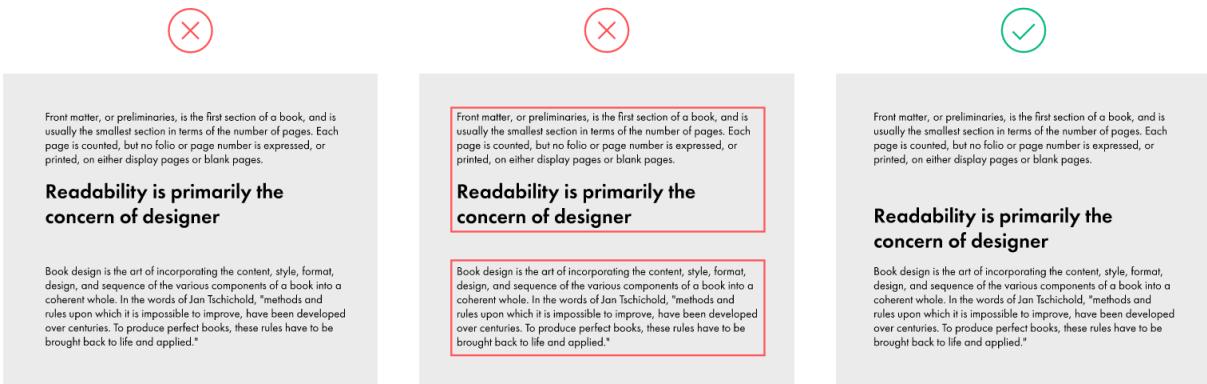


Рисунок 3.13. Частая ошибка: заголовок текста “висит” между абзацами, в то время как должен быть ближе к следующему абзацу, к которому он и относится.

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

Рисунок 3.14. 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 не по разрядам.

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 DiBlase	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

Рисунок 3.15. Таблица после исправления ошибок в дизайне. Выравнивание текста по левому краю и пустоты между колонками создают их чёткие границы. Нет нужды в отделении их друг от друга линиями. Равномерный белый фон уменьшает количество визуального мусора.

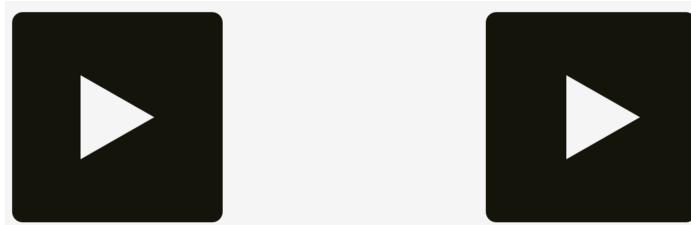


Рисунок 3.16. Центр треугольника слева совпадает с центром квадрата, но воспринимается так будто расположен левее. Правый треугольник расположен так, что половина его площади лежит слева от вертикальной оси симметрии квадрата.



Рисунок 3.17. Многие буквы слегка выходят за базовую линию (нижняя линия) и за медиану (верхняя линия) чтобы их размер не воспринимался меньшим, чем он есть на самом деле.

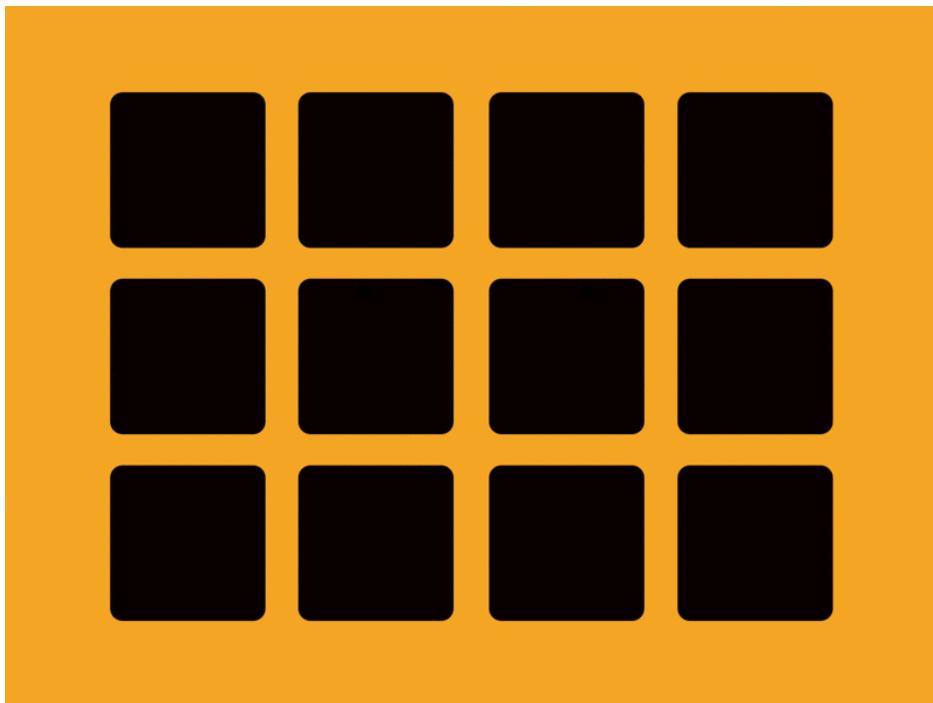


Рисунок 3.18. Решётка Германа: места на пересечении линий, разделяющих квадраты, воспринимаются имеющими серые пятна.

тета, 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 395fb6 светлее? Как сделать этот цвет менее насыщенным? Необходимые изменения трудно выразить в виде изменения интенсивности красного, зелёного и синего.

Модель HSV призвана решить этот недостаток. HSV (Hue, Satura-

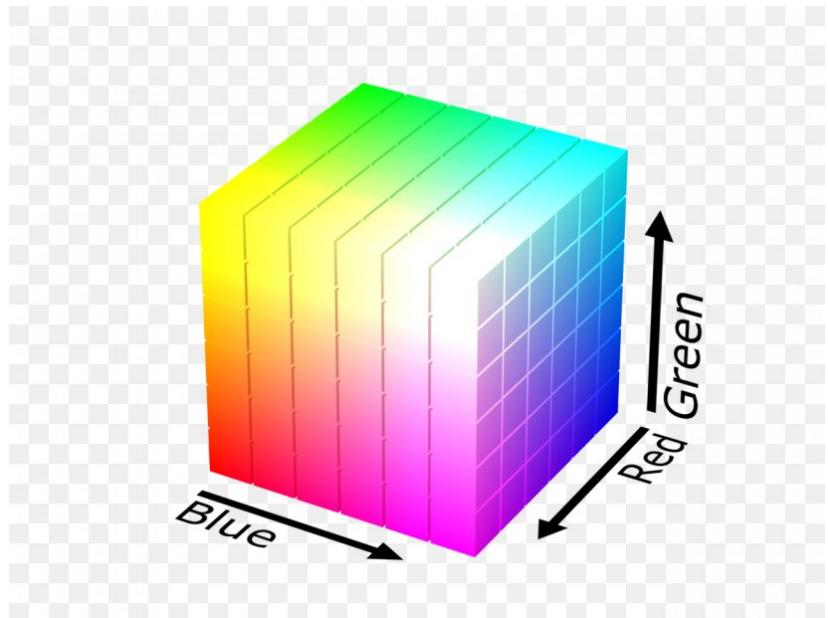


Рисунок 3.19. Три цветовые координаты в модели RGB. Скрытая вершина куба окрашена в чёрный.

tion, Value — тон, насыщенность, значение) или HSB (англ. Hue, Saturation, Brightness — тон, насыщенность, яркость) — цветовая модель, в которой координатами цвета являются (см. рис 3.20):

- Hue — цветовой тон, (например, красный, зелёный или синеголубой). Варьируется в пределах  $0\text{--}360^\circ$ , однако иногда приводится к диапазону  $0\text{--}100$  или  $0\text{--}1$ .
- Saturation — насыщенность. Варьируется в пределах  $0\text{--}100$  или  $0\text{--}1$ . 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  $0\text{--}100$  или  $0\text{--}1$ .

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:  $222^\circ$ , 69%, 71%. В такой записи для изменения насыщенности и яркости нужно изменить только по одному параметру. Все современные графические редакторы,

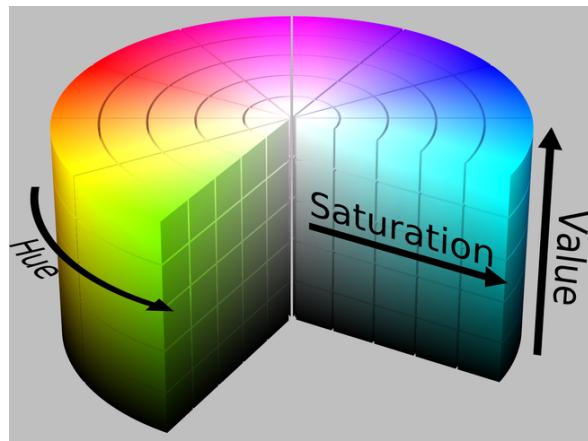


Рисунок 3.20. Модель HSV (HSB). Чтобы изменить только насыщенность (saturation) или только светлоту (value) нужно изменить единственное число в коде цвета.

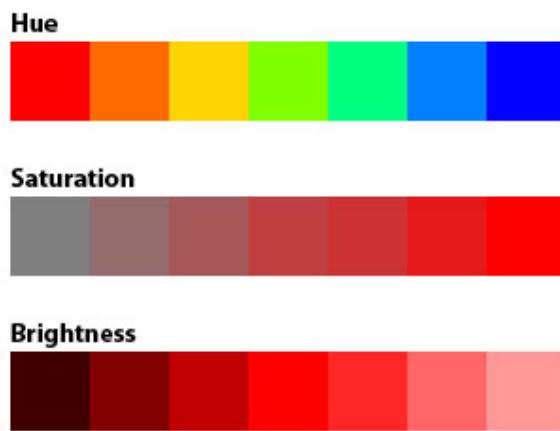


Рисунок 3.21. Пример изменения насыщенности (Saturation) и яркости (Brightness) для одного тона (Hue).

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

Интерфейс пользователя – это посредник между программой и человеком. Его цель – помочь решить задачу пользователя, но не привлекать к себе внимание (рис. 3.23, 3.22) и, тем более, не давать большую нагрузку на пользователя. Поэтому в интерфейсах стоит использовать цвет преимущественно как инструмент, помогающий пользователю взаимодействовать с программой (рис. 3.24). Насыщенные и яркие цвета привлекают внимание. Поэтому, такие цвета занимать небольшую площадь экрана, привлекая вни-

мание только к самым важным элементам интерфейса и не рассеивать внимание пользователя. Фон напротив – не должен привлекать внимания вовсе. При этом текст и другие элементы интерфейса на любом фоне должны давать высокий контраст.

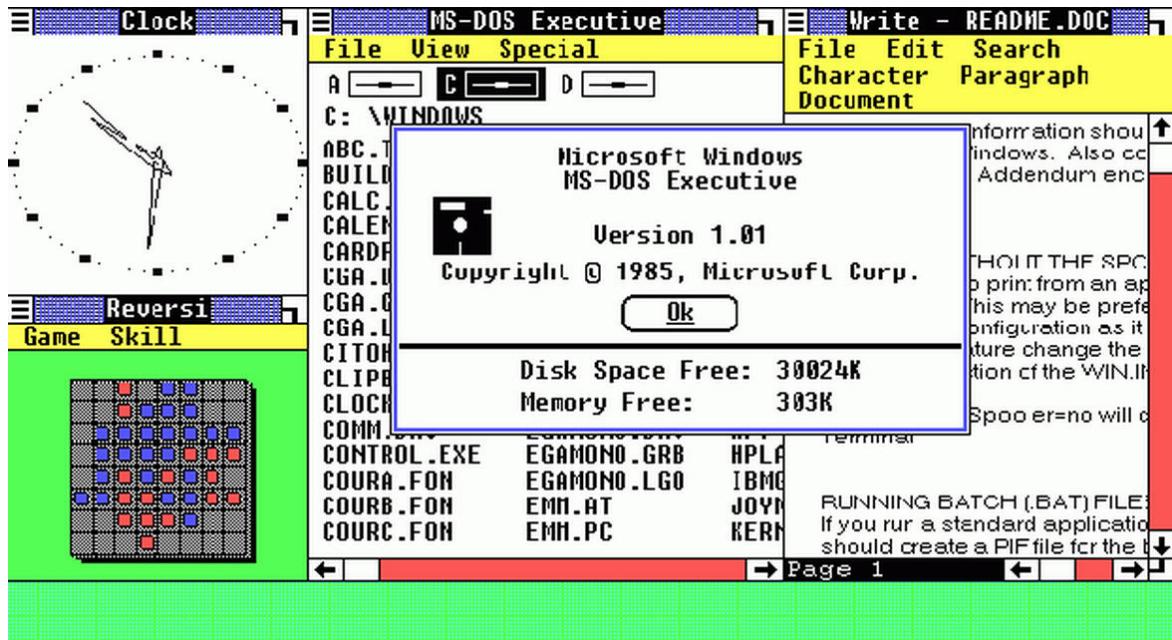


Рисунок 3.22. Дизайн интерфейса Windows 1 был создан преимущественно без участия дизайнеров. Абсолютное большинство современных интерфейсов, несмотря на возросшие технические возможности дисплеев, имеют более сдержанную палитру. Задача пользовательского интерфейса – не привлекать к себе внимание, а помочь решить задачу пользователя.

Подбор цветовых схем – сложная задача. Отправной точкой могут служить руководства по стилю бренда и платформы, для которой разрабатывается продукт (например руководства Android или iPhone). Ещё один вариант изучение цветовых схем похожих продуктов или макетов на Dribbble.com или behance.net. Для макета или страницы можно составить набор цветов и разобрать в их назначении. 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.

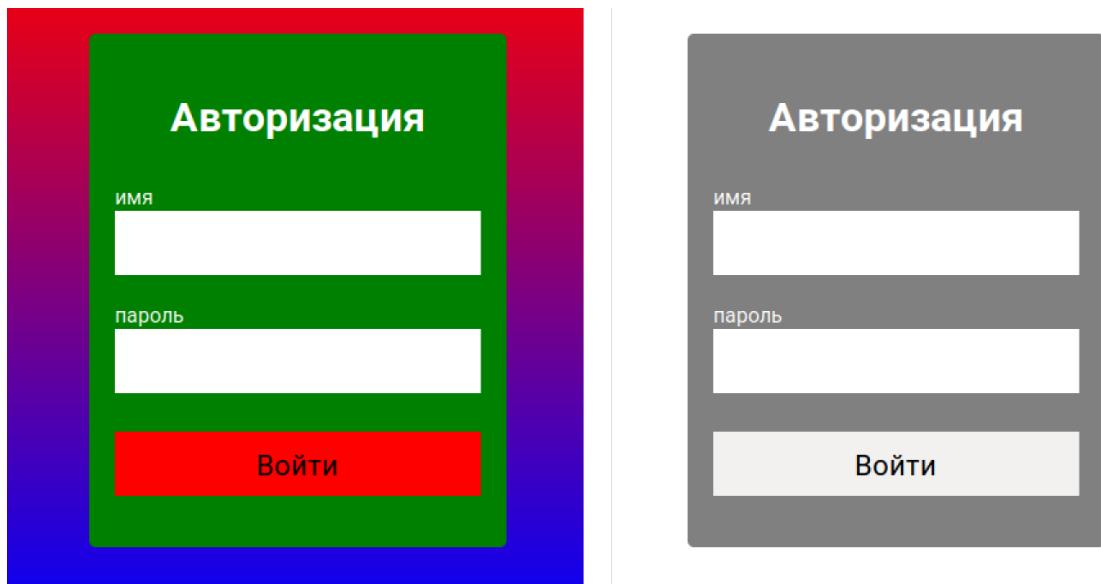


Рисунок 3.23. Большое количество визуальных стимулов, даже при сохранении высокого контраста и различимости может проигрывать в эстетическом плане “скучным” цветовым схемам.

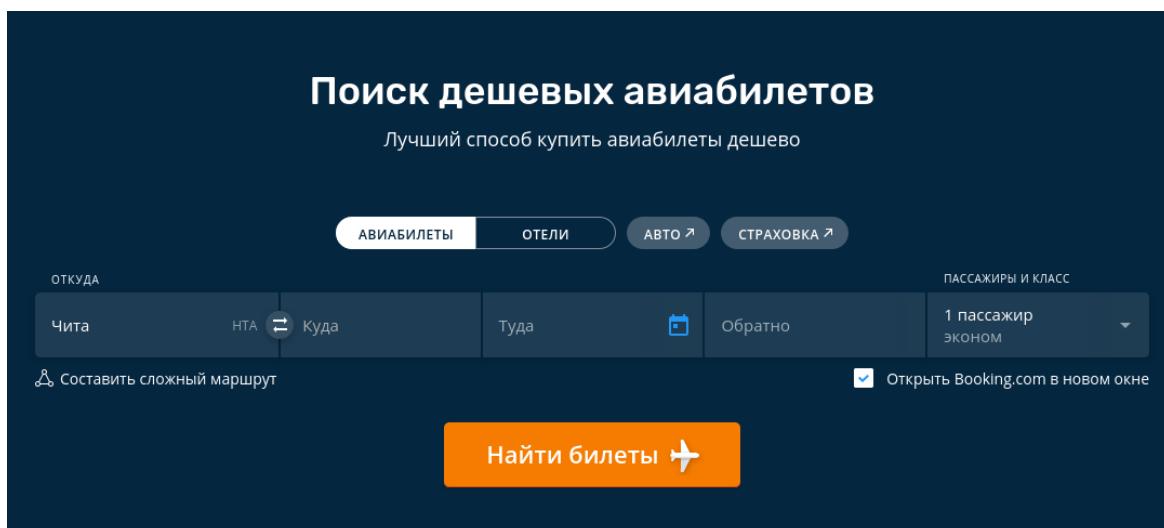


Рисунок 3.24. В основе акцентирования внимания цветом – эффект выскакивания.

## Вопросы

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

## Литература

1. Голомбински, К. Добавь воздуха! Основы визуального дизайна для графики, веба и мультимедиа / К. Голомбински, Р. Хаген. — : Питер, 2013. — 272 с. — Текст : непосредственный.
2. 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<sup>1</sup> 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 [64].

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.

---

<sup>1</sup> 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<sup>2</sup>. Таким образом, можно создать положительную субъективную оценку от продукта, который имеет относительно серьёзные 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 сложиться впечатление, что она не выполнена или выполнена с ошибкой.

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

---

<sup>2</sup>как и бренд ≠ качество товара

## 4.2 Проектирование UX

### 4.2.1 Уровни UX

Таким образом, задача создания продукта может рассматриваться как задача создания положительного UX. Опыт пользователя плохо формализуется в силу того, что может отличаться 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 [9]. 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 пользователей – бесполезен. Здесь же мы должны понять 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.

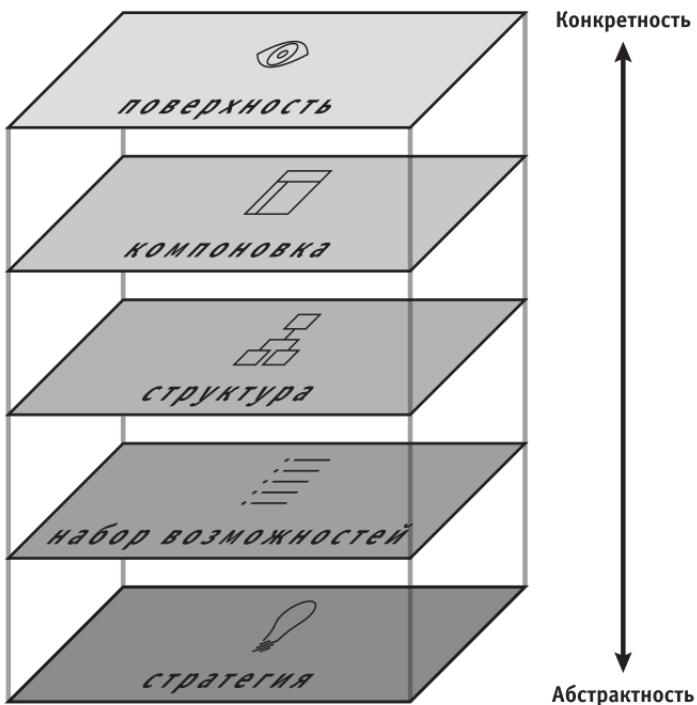


Рисунок 4.1. Пять уровней проектирования UX.

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

Когда готовы требования к функционалу 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 поль-

зователя, 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. Далее, развивая каждое из принятых решений, проводят продукт к его конечному виду (рис. 4.2).

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).

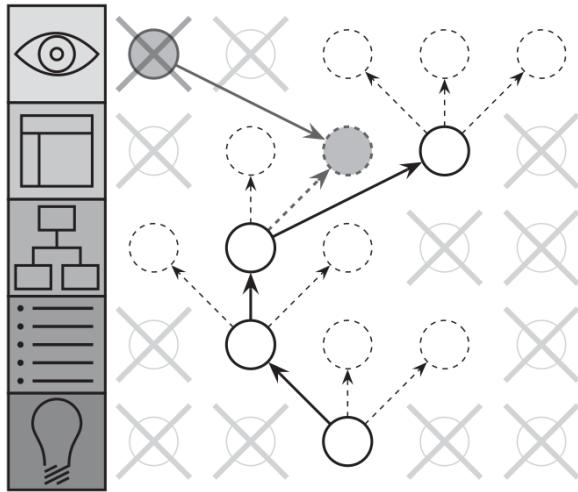


Рисунок 4.2. Принимая решения на каждом уровне проектирования, разработчик сужает диапазон решений на каждом следующем уровне. Это упрощает проектирование. Решения, принятые на разных уровнях одновременно, могут быть не согласованы.

Здесь возможны несколько вариантов:

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

**Изучение заказчика.** Основной источник по стилю – брендбук<sup>4</sup>.

Возможно, понадобится написать для заказчика техническое задание<sup>5</sup>.

Заказчик не всегда стремится удовлетворить потребности пользователя. Иногда он стремится эти потребности создать. Например, предлагает дополнительные товары в интернет-магазине, делает таргетированную рекламу.

**Одна из проблем дизайна продуктов – дизайн продуктов дол-**

<sup>3</sup>Enterprise Resource Planning, планирование ресурсов предприятия. 1С: Предприятие – пример ERP ПО

<sup>4</sup>Брендбук – официальный документ компании, в котором описывается концепция бренда, и другие данные, включая полное руководство по фирменному стилю

<sup>5</sup>Техническое задание (ТЗ, техзадание) – документ или несколько документов, определяющих цель, структуру, свойства и методы какого-либо проекта, и исключающие двусмысленное толкование различными исполнителями

жен быть предназначен для пользователей, но чтобы быть реализованным он должен понравиться заказчику. Который, как правило, неспециалист 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 Алланом Купером. Он помогает структурировать целевую аудиторию продукта, задокументировать потребности и особенности значимых групп пользователей. Предполагается, что продукт проектируется для персонажей. Этот подход также используется в маркетинге.

Персонажи описываются на основе исследования потенциальных пользователей. Если данных о целевой аудитории недостаточ-

но, то условный персонаж, описание которого частично состоит из гипотез, тоже чаще всего оказывается полезен.

Описание персонажа обычно включает в себя:

- 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 свои цели?

Конкретный набор характеристик персонажа может меняться 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 пользуется 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.

Дмитрий, 25 лет.

Работает системным администратором, студент, учится заочно.

**Цели:**

- Хочет получить образование и диплом бакалавра в ИТ (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.

**Используемые устройства:** смартфон с ОС Андроид 10, ноутбук с ОС Windows 10.

#### 4.2.4 Диаграмма прецедентов

Для описания функциональных возможностей может использоваться **use case** диаграмма (**диаграмма прецедентов**)

Диаграмма вариантов использования (use case diagram, диаграмма прецедентов) – UML-диаграмма, отражающая отношения между актёрами и прецедентами [13].

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 1-3 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 соответствуетциальному сервису системы, определяет один из вариантов её использования и описывает типичный способ взаимодействия пользователя с системой.

Для примера приведём фрагмент диаграммы вариантов использования (рис. 4.3) банковского приложения, которая иллюстрирует

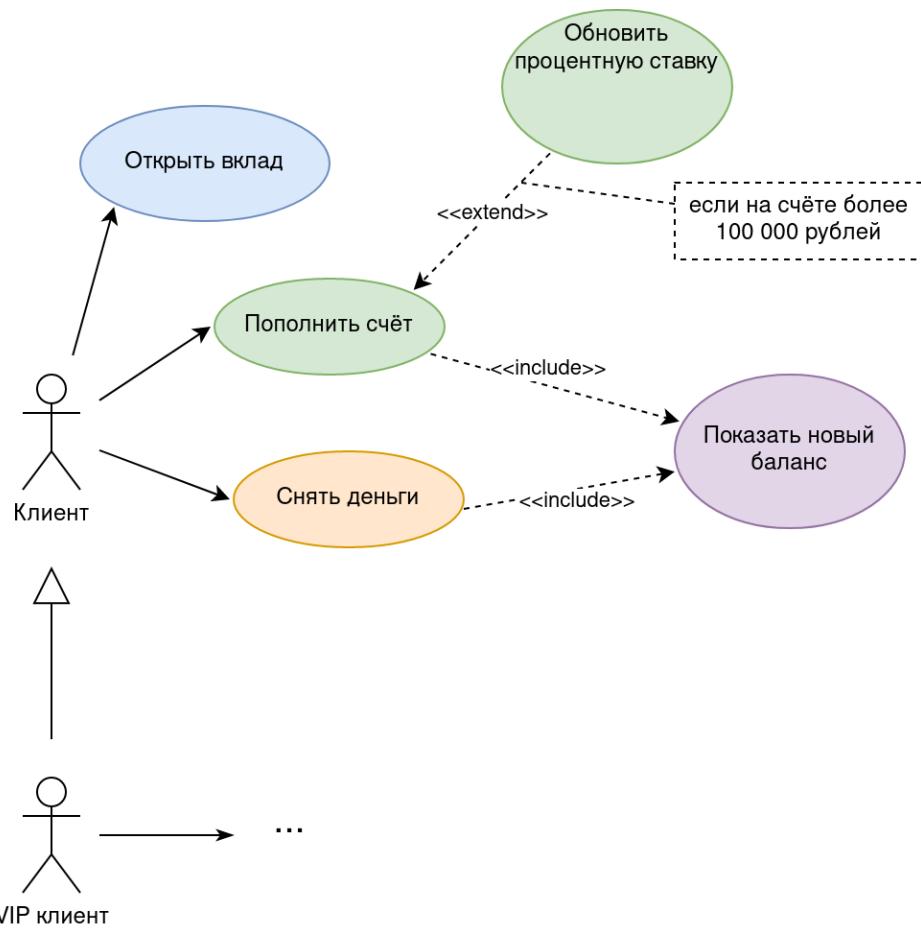


Рисунок 4.3. Фрагмент диаграммы вариантов использования для банковского приложения. Extend – расширение прецедента, include – включение. Отношение между “Клиентом” и “VIP клиентом” – обобщение. Для удобства восприятия не связанные по смыслу прецеденты окрашены в разные цвета.

основные элементы и отношения между ними, но не претендует на полноту. У пользователя могут быть две роли: клиент, VIP клиент. 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 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 предцентров из диаграммы вариантов использования, оценить на сколько удобна и понятна такая структура для решения типичных задач персонажей.

#### **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. 4.5). 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. 4.5). 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 ме-

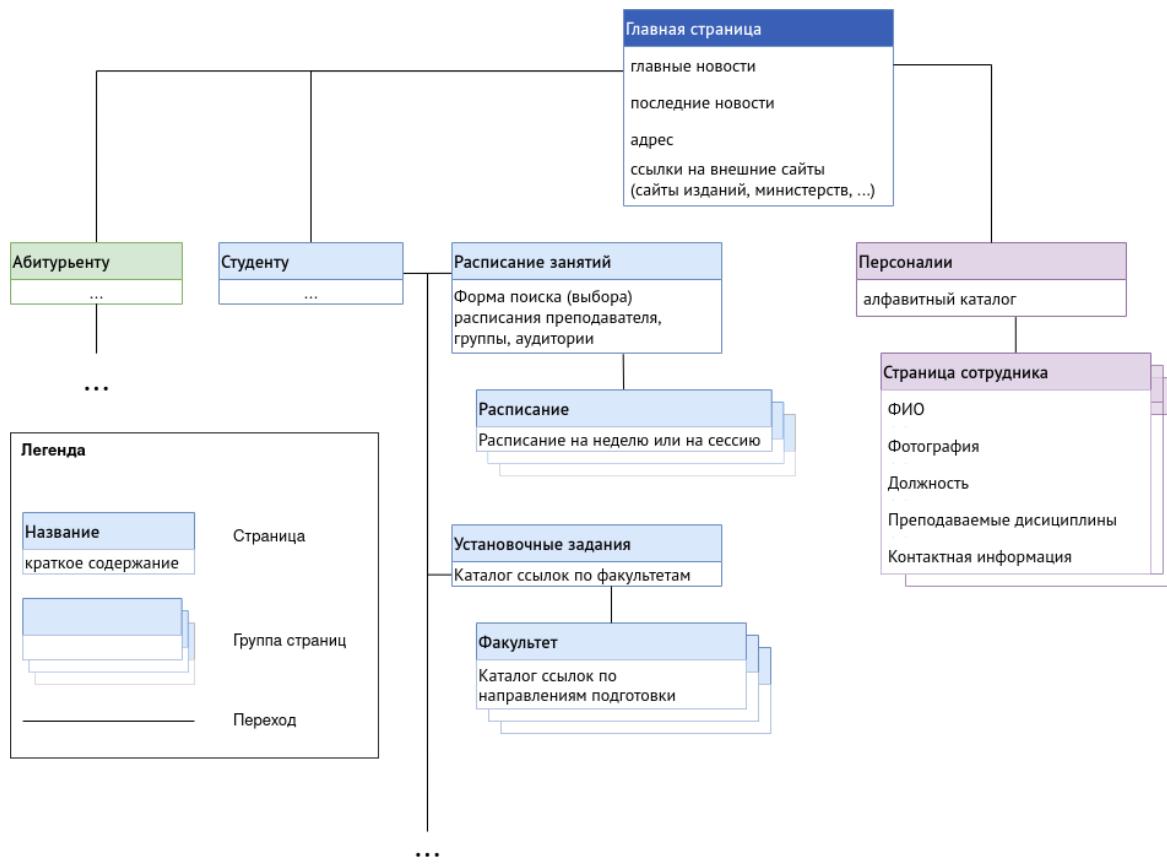


Рисунок 4.4. Фрагмент диаграммы страниц сайта университета. “Страница сотрудника”, “Факультет”, “Расписание” – набор страниц с одинаковым назначением, но разным содержанием.

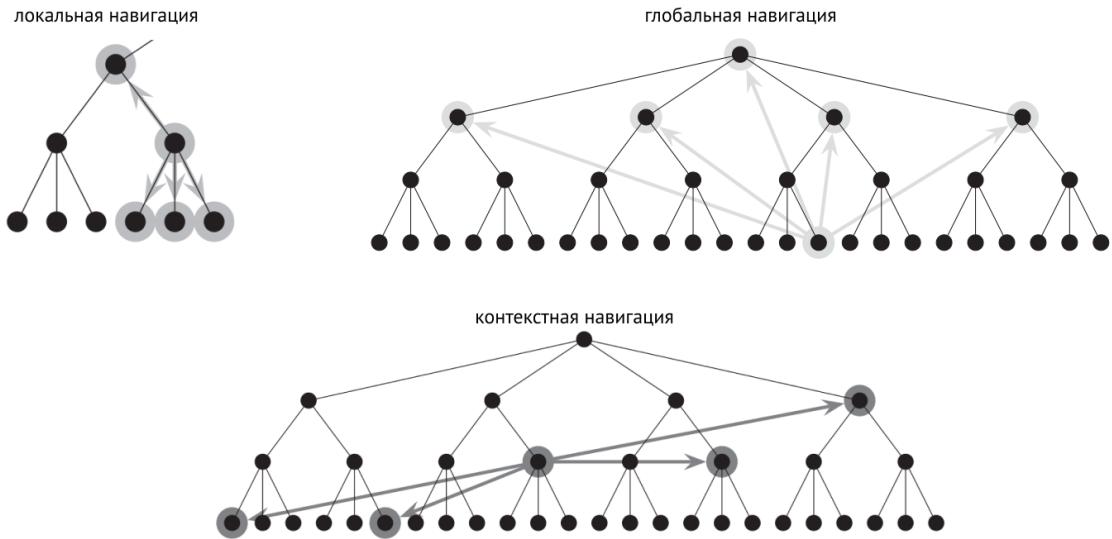


Рисунок 4.5. Отдельные виды навигации.

ню 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<sup>6</sup> (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 тестирования.

<sup>6</sup>вайрфрейм – 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

Рисунок 4.6. Основные элементы диаграммы потоков задач.

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 [27]. 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 по-

смотреть не на статичную структуру продукта, а на процесс его использования. 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.



Рисунок 4.7. 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 нескольких фраз, кратко суммирующих задачи и цели, он поможет преобразо-

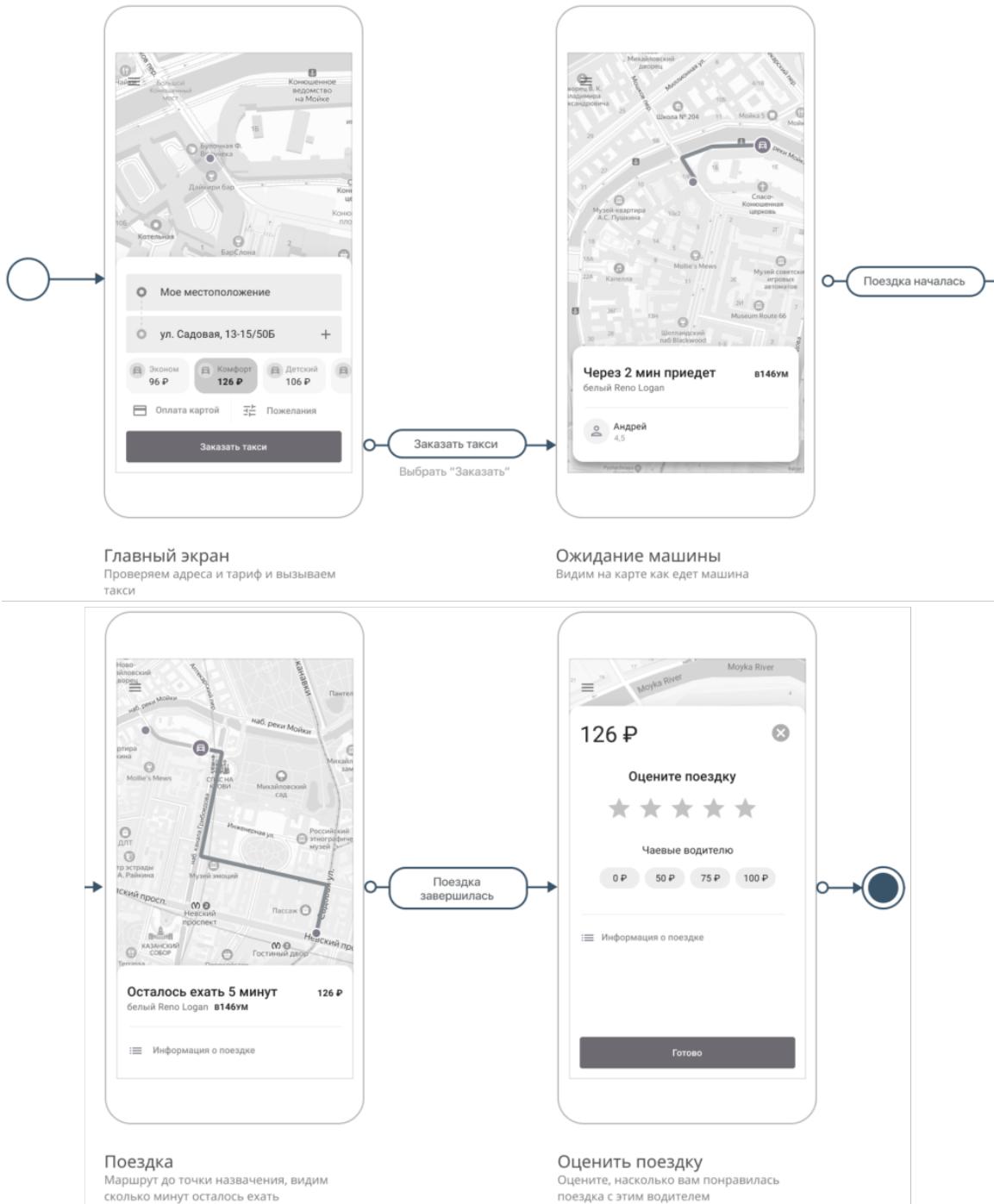


Рисунок 4.8. Диаграмма потока (screen flow) заказа такси в мобильном приложение представленная в виде последовательности экранов и действий.

вать 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 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.4.9).

Низко детализированные эскизы полезны для быстрого создания экранов. Легко создавать несколько вариантов одного экрана (рис. 4.10), 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 дизайнера 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. 4.15).

Есть хороший способ убедиться, что визуальный дизайн эффективно задействует иерархию и отношения, – дизайнеры называют



Рисунок 4.9. Эскиз окна веб-страницы: видна компоновка элементов интерфейса, основные элементы подписаны, изображение приведено схематично, палитра не задана.

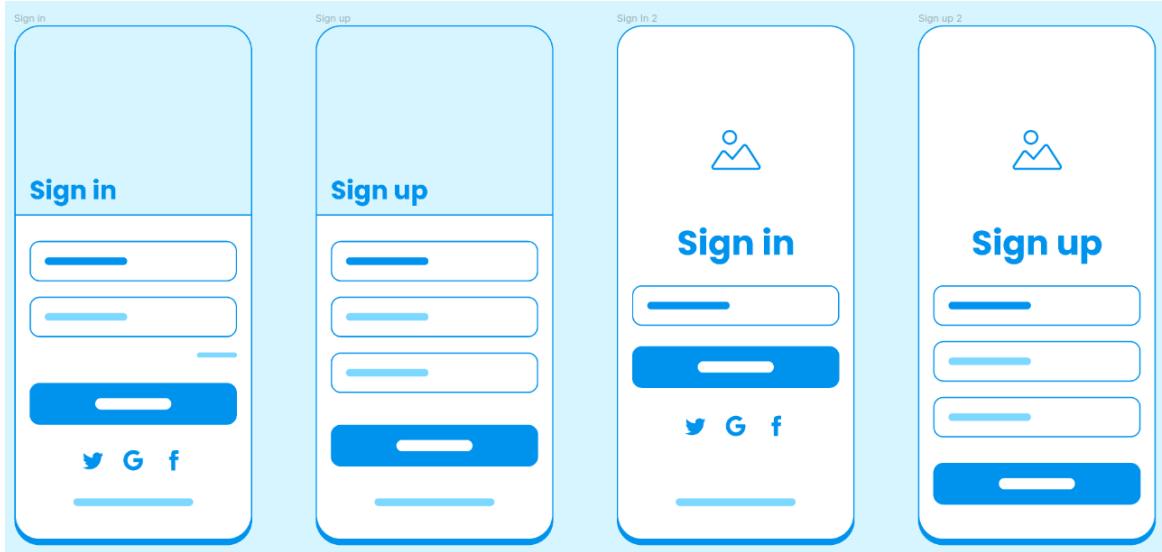


Рисунок 4.10. Низко детализированные варианты экранов приложения. Выполнены в Figma, собраны из библиотеки готовых компонентов. Обозначены элементы интерфейса, расположение отдельных надписей, выбранная палитра играет техническую роль и не отражает конечный вид интерфейса.

ют этот прием тестом с прищуриванием (squint test)[10]. 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 жизнен-

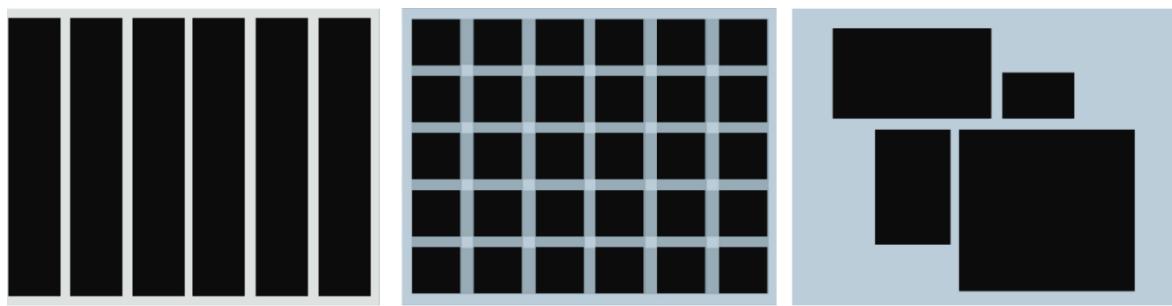


Рисунок 4.11. Композиционные сетки: колоночная, модульная, блочная. Прямоугольники, размечающие пространство макета, всего располагаются с отступом друг от друга. Элементы интерфейса могут занимать пространство равное одному или нескольким прямоугольникам.

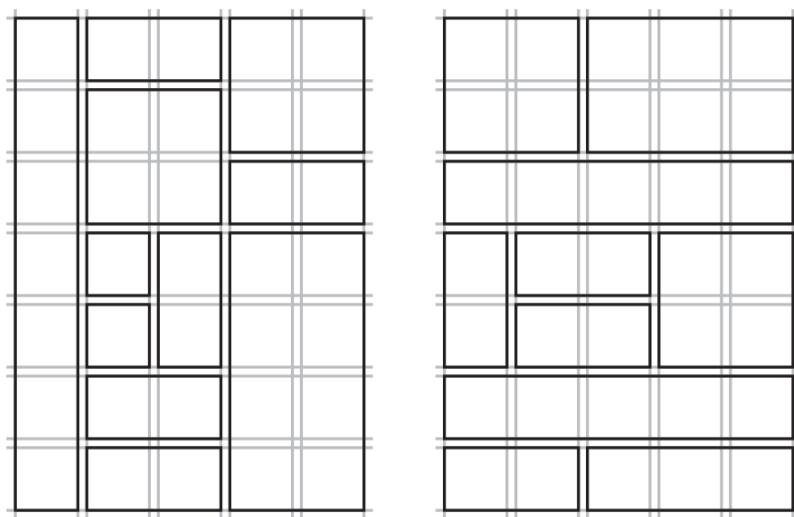


Рисунок 4.12. Варианты расположения блоков в модульной сетке.

ное пространство для менее важных элементов и элементов более низкого уровня.

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.

Композиция должна не только в точности следовать сетке, но и структурировать **эффективный логический маршрут** через ин-

**The Grid System**

The ultimate resource in grid systems.

*"The grid system is an aid, not a guarantee. It permits a number of possible uses and each designer can look for a solution appropriate to his personal style. But one must learn how to use the grid; it is an art that requires practice."*

Josef Müller-Brockmann

Articles	Tools	Books	Templates
<b>Musings on the Relationship Between Grids and Guides</b> An article that takes a look at the relationship between the grid and the use of guides. 06.Feb.2011	<b>GuideGuide</b> A columns, rows and mid-points panel for Photoshop CS4 & CS5. 06.Feb.2011	<b>Ordering Disorder: Grid Principles for Web Design</b> Ordering Disorder is a book by Khoi Vinh that delivers a definitive take on grids and the Web and provides both the big ideas and techniques of grid-based design. 11.Nov.2010	<b>960px Grid Template</b> A selection of 960 wide uniform grid plates ranging from 2 columns to 16-columns for both Adobe Photoshop and Fireworks. 11.Nov.2010

Рисунок 4.13. Блоки с текстом занимают один или несколько модулей. Модули изображены розовыми прямоугольниками.

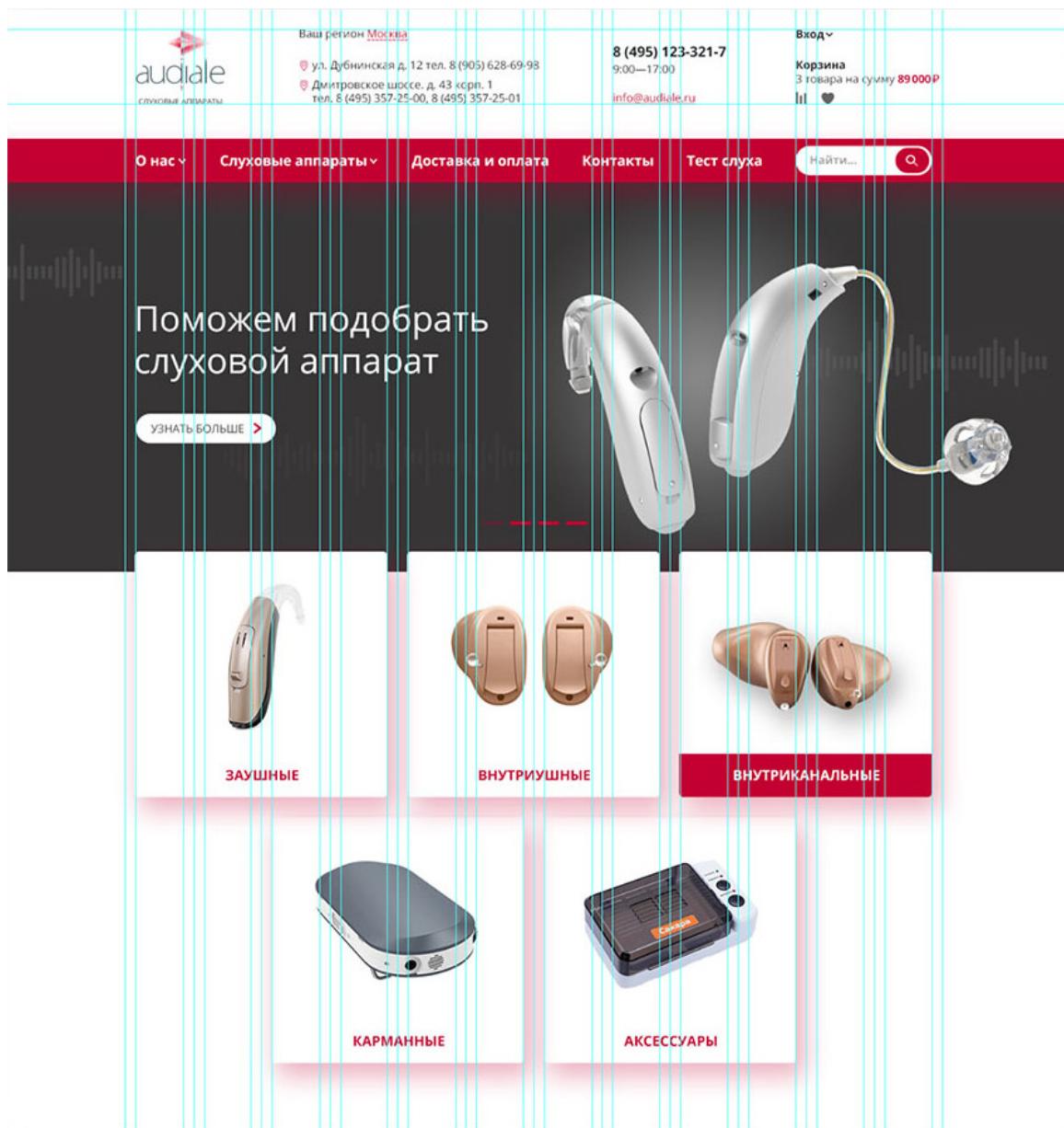


Рисунок 4.14. 12-и колоночная сетка, используется в Bootstrap – свободный наборе инструментов для создания сайтов и веб-приложений.

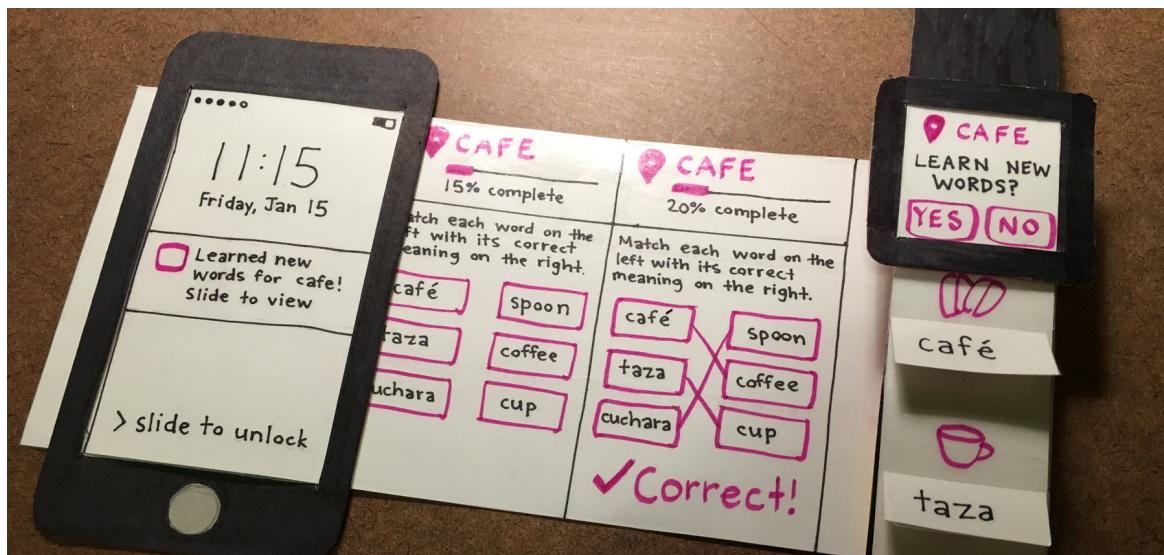


Рисунок 4.15. Бумажный эскиз пользовательского интерфейса.

терфейс для пользователей, принимая во внимание тот факт, что (в случае западных языков) взгляд движется сверху вниз и слева направо (рис. 4.16).

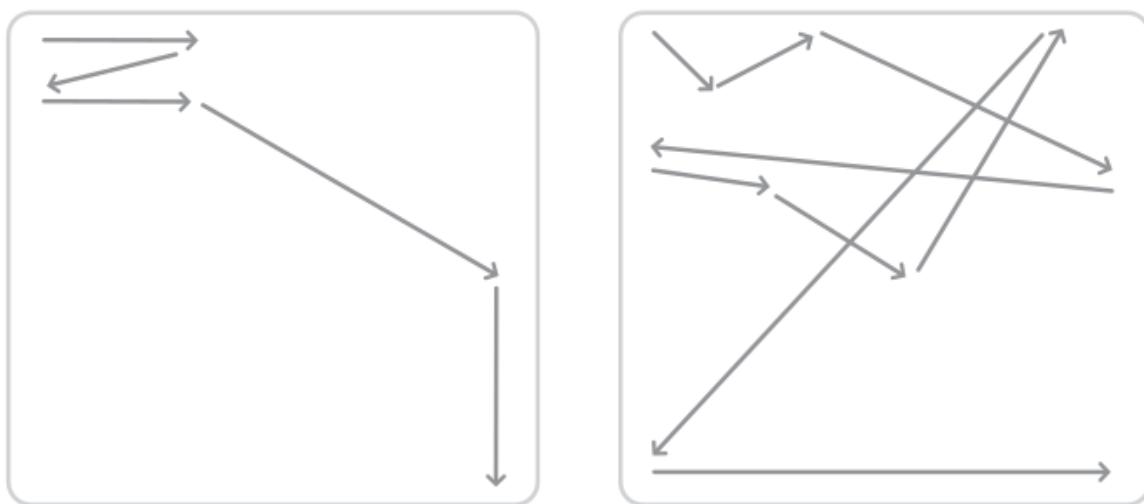


Рисунок 4.16. Хороший логический маршрут (слева) взаимодействия с элементами интерфейса направлен слева на право, не требует от пользователя беспорядочного перемещения взгляда, в отличии от варианта справа

#### 4.2.8 Уровень поверхности

На этом этапе интерфейс получает свой окончательный вид. Должна быть выбрана цветовая палитра, нарисованы иконки и созданы

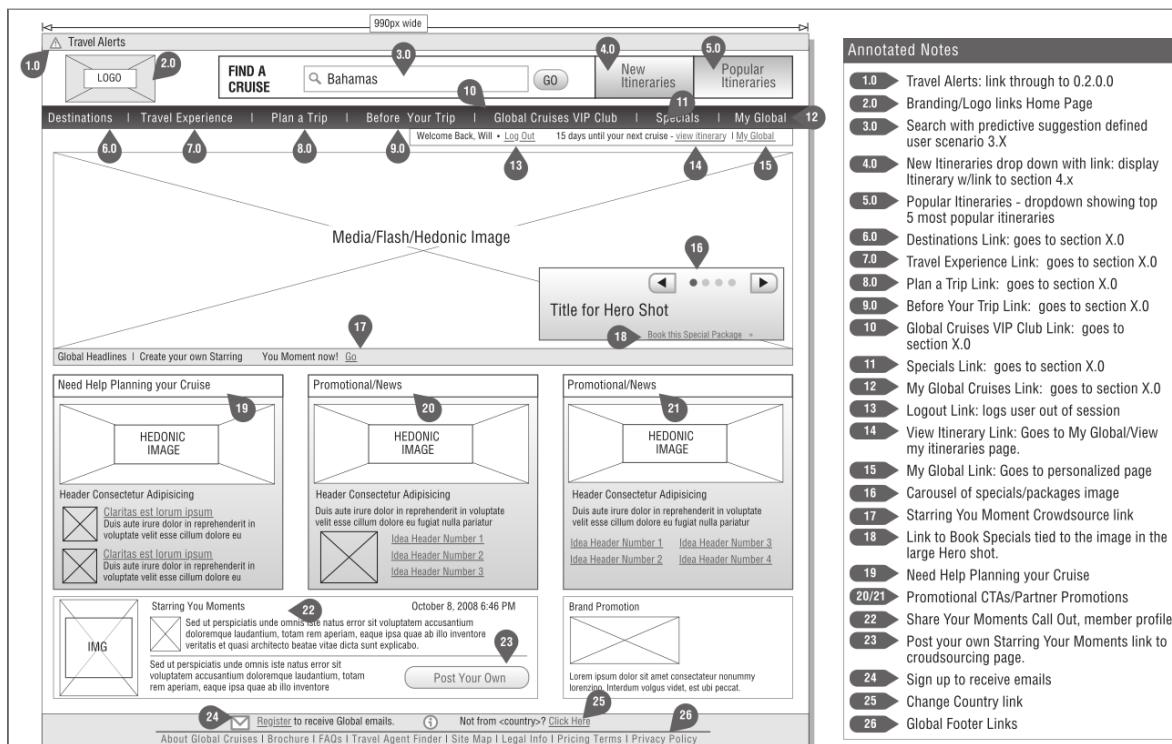


Рисунок 4.17. Аннотированный макет.

изображения, окна или страницы наполнены содержанием.

Всё вышеописанное представляется в окончательном макете (рис. 4.18) или прототипе. Макет, демонстрирующий облик интерфейса пользователя обычно называют мокапом (mockup). Прототип, в отличие от макета, предполагает имитацию взаимодействия с пользователем. 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 нужны и полезны также разработчиками (программистам), руководителям проекта и заказчику. Причём, для них имеет смысл создать аннотированный макет, в котором есть пояснения к поведению и назначению элементов интерфейса (рис. 4.17).

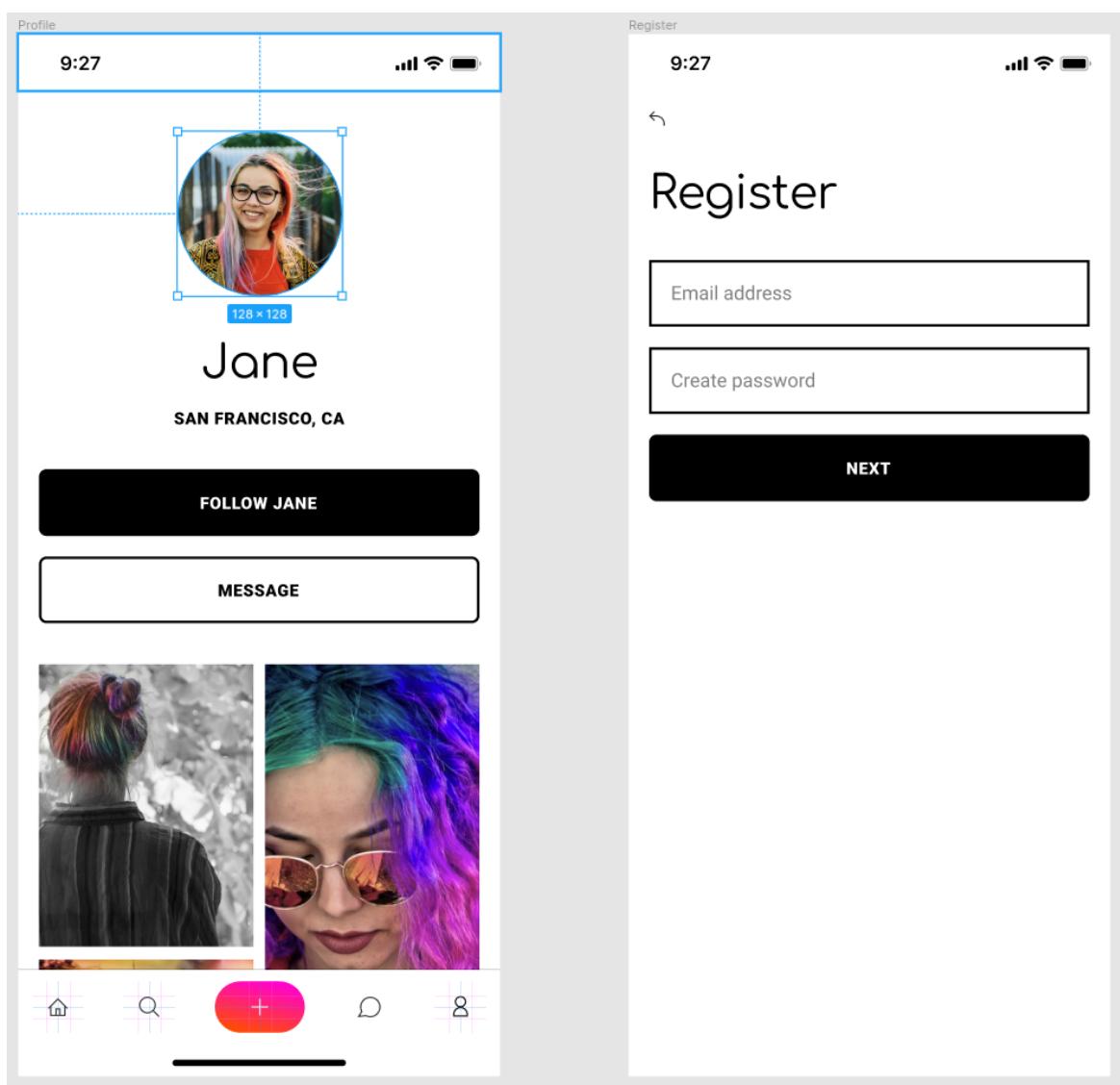


Рисунок 4.18. Макет пользовательского интерфейса в Figma.

#### 4.2.9 Figma

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.

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

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 версткой). Исторически для создания макетов широко используются графические редакторы, например Photoshop. В 2010-х начали появляться редакторы векторной графики, ориентированные на проектировании интерфейсов. Sketch и Figma – наиболее популярные из них.

Далее будет коротко рассмотрены основы работы в Figma.com. Полное руководство по Figma можно найти в официальной документации, или, например, в книге [58].

**Фреймы** – главные строительные элементы макета. На панели слоёв обозначаются знаком, похожим на  $\sharp$ . Фреймы, в первую очередь, используются для того чтобы представлять отдельную страницу или окно приложения. Рекомендуется именовать фреймы и все вложенные в них элементы так, чтобы по названию можно было судить о их назначении. Допустимы названия на русском и содержащие пробелы.

Элементы пользовательского интерфейса – кнопки, меню, поля

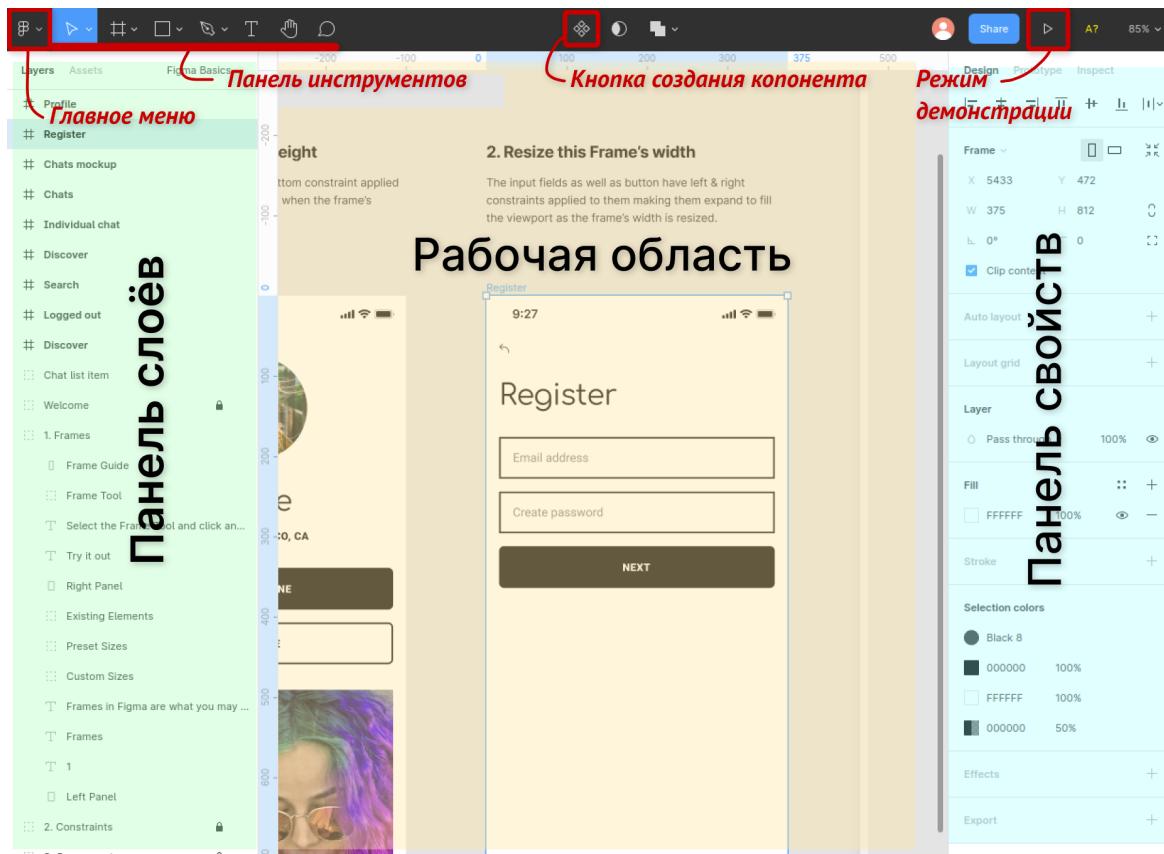


Рисунок 4.19. Интерфейс Figma.

ввода и т.д. могут быть созданы самостоятельно с помощью простых инструментов (рис. 4.21) или использованы из любой доступной библиотеки [figma.com/community/ui\\_kits](https://figma.com/community/ui_kits).

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 расположения элементов по сетке помогают сократить<sup>7</sup> количество вариантов компоновки элементов, ускорить проектирование интерфейса.

Если шаг сетки составляет четыре пикселя, то все расстояния между элементами и должны быть кратны четырем пикселям.

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

<sup>7</sup> 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

Frame	
▶ Phone	
▶ Tablet	
▼ Desktop	
MacBook Pro 14"	1512 × 982
MacBook Pro 16"	1728 × 1117
Desktop	1440 × 1024
iMac	1280 × 720
▶ Presentation	
▶ Watch	
▶ Paper	
▶ Social media	
▶ Figma Community	
▶ Archive	

Рисунок 4.20. Заготовки для фреймов. Как правило ширина фрейма сохраняется во время разработки макета, а высота изменяется в зависимости от контента.

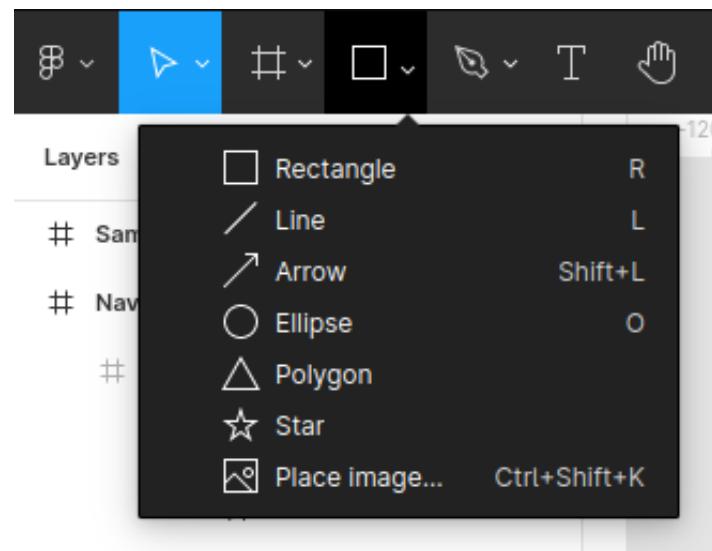


Рисунок 4.21. Простые фигуры для рисования элементов интерфейса.

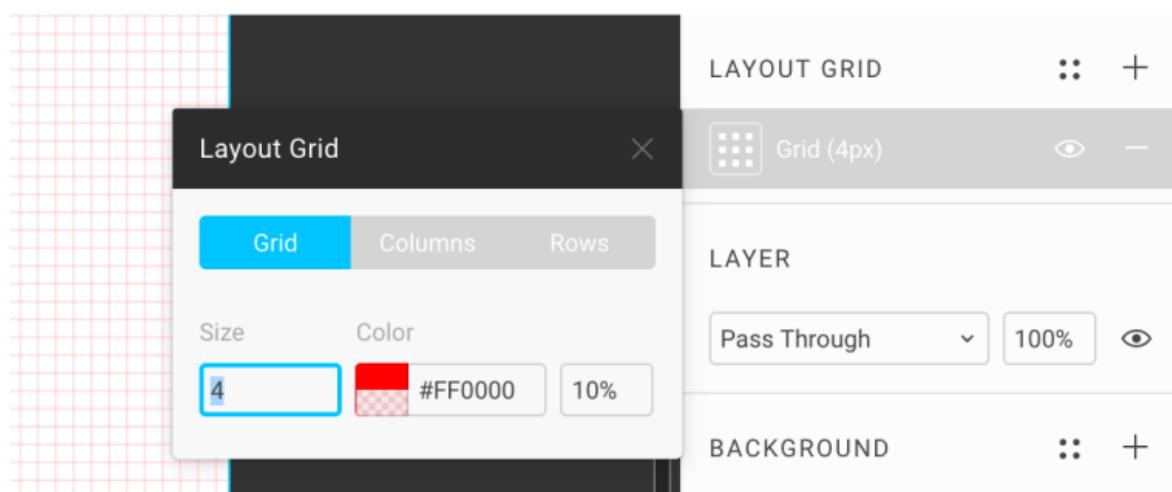


Рисунок 4.22. Свойство фрейма: Layout grid, регулярная сетка – grid.

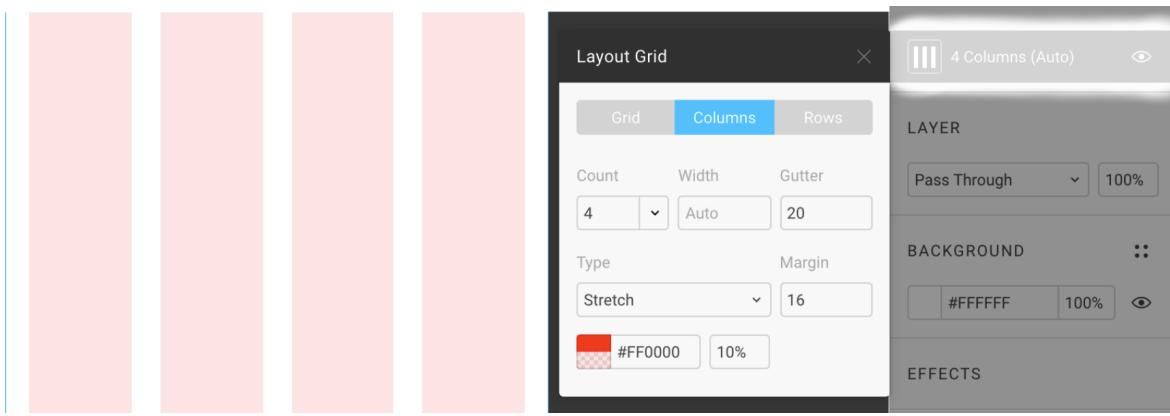


Рисунок 4.23. Свойство фрейма: Layout grid, колоночная сетка - columns. Элемента интерфейса в такой сетке занимают ширину одной или несколько колонок (выделены розовым). Count – количество колонок, Gutter – расстояние между колонками, Margin – поля слева и справа соответственно от первой и последней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

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.

Ещё одна хорошая практика – повторное использование ранее созданных элементов. Например кнопок, меню, повторяющихся паттернов: форм, карточек товара, диалоговых окон. Сохраняя общую структуру эти части интерфейса могут иметь разное содержимое. **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.

Рассмотрим создание компонента для кнопки. Создадим прямоугольник (инструмент Rectangle, горячая клавиша - R). Зададим цвет через свойство Fill и закругление углов (рис. 4.24). Добавим текст (горячая клавиша - T), задав размер, цвет с достаточным контрастом с фоном. Зададим выравнивание элементов по центру друг

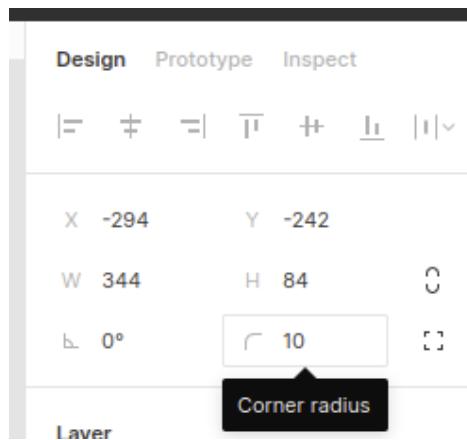


Рисунок 4.24. Задание радиуса скругления прямоугольника.

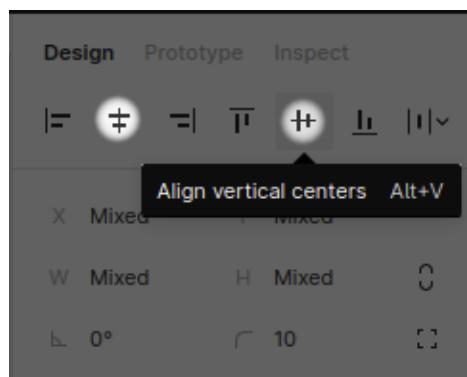


Рисунок 4.25. Выравнивание выбранных элементов относительно общего центра.

друга: выбрать все элементы, выровнять по горизонтальному и вертикальному центру 4.25. Не снимая выделения создадим компонент ( $Alt + Ctrl + K$ ) или нажав значёк в виде составного ромба на панели инструментов (рис. 4.19).

Текст надписи на кнопке должен быть привязан к центру компонента (рис 4.25). Это избавит от необходимости передвигать текст вручную при изменении размеров компонента.

Создадим три копии компонента, изменив во второй копии текст и размер, в третей – текст и цвет. 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 цве-

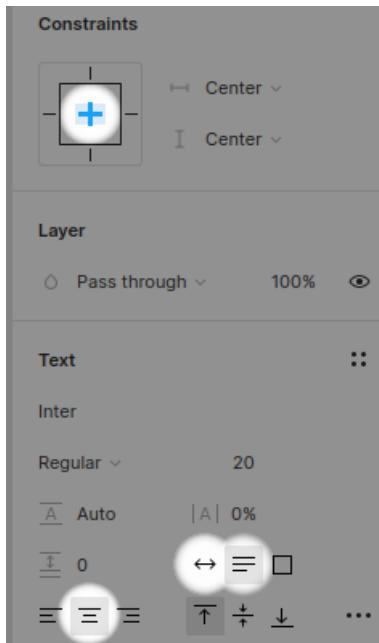


Рисунок 4.26. Constraints: привязка элемента к центру компонента, при изменении размеров компонента текст останется в его центре. Text: задать автоматический (минимально возможный) размер текстового блока и расположение текста по центру этого блока.

та 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 элюентов интерфейса, чтобы учитывать контекст. Если экземпляр компонента нужно значительно изменить, его можно отвязать от самого компонента, выбрав в контекстном меню экземпляра пункт *Detach instance*.

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 4.28): выделить компонент, в свойствах нажать + напротив Variants. Экземпляры одного компонента легко заменять на его варианты: контекстное меню экземпляра > change

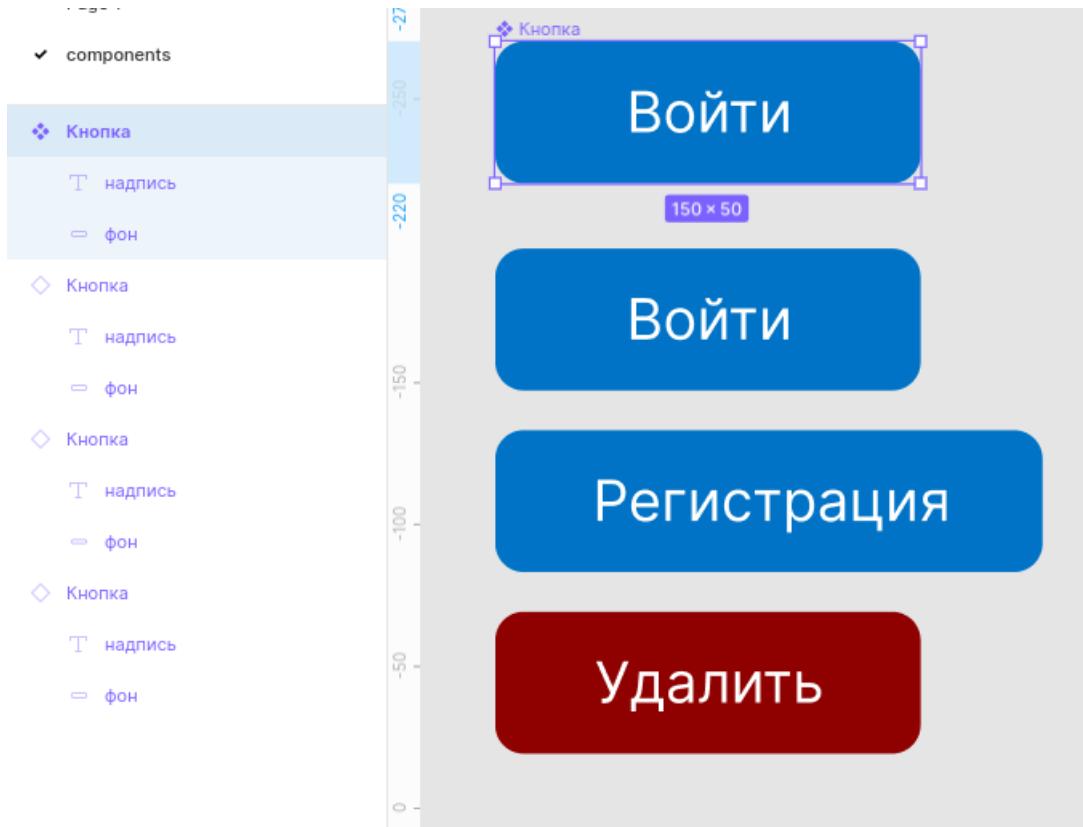


Рисунок 4.27. Компонент (обозначен составным ромбом) и его экземпляры (обозначены ромбами).

variant. При этом изменяются 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. 4.29).

Отдельные элементы макета должны иметь одинаковые свойства, для них нецелесообразно создавать отдельный компонент. Например надписи на всех макетах 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

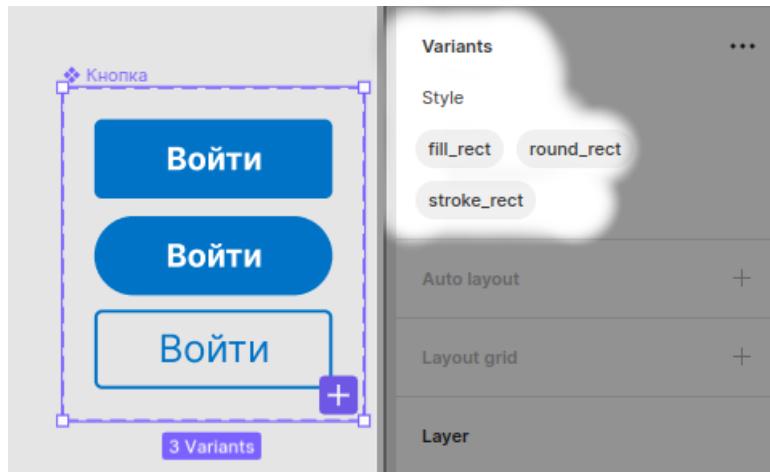


Рисунок 4.28. Варианты компонента “Кнопка”. Каждый вариант формально отличается от другого значением свойства Style (название задано в свойствах компонента): fill\_rect, round\_rect, stroke\_rect.

разделе Color нажать на кнопку с четырьмя точками, в появившемся окне нажать + и в новом окне задать название для цвета (рис. 4.30). Все стили отображаются на панели свойств проекта, которая отображается если не выбран ни один объект (рис. 4.31).

Если в Figma создаётся макет сайта, то можно экспортить свойства объектов в CSS. Описание стилей объектов содержится на панели свойств, на вкладке Inspect 4.33.

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 кнопку. Интерактивный функционал доступен в режиме демонстрации макета (рис. 4.19) Рассмотрим создание перехода: выделим кнопку, на панели свойств переключимся на вкладку Prototype (шаг 1 на рис. 4.35), включится режим редактирования прототипа. У каждого элемента будет нарисована окружность (рис. 4.34), потянув за которую можно указать на любой фрейм. По умолчанию фрейм будет показан после клика на выбранный элемент. В дополнении можно настроить друге действие и анимацию (шаг 3 и 4 на рис. 4.35), которые будут выполнены при взаимодействии с элементом.

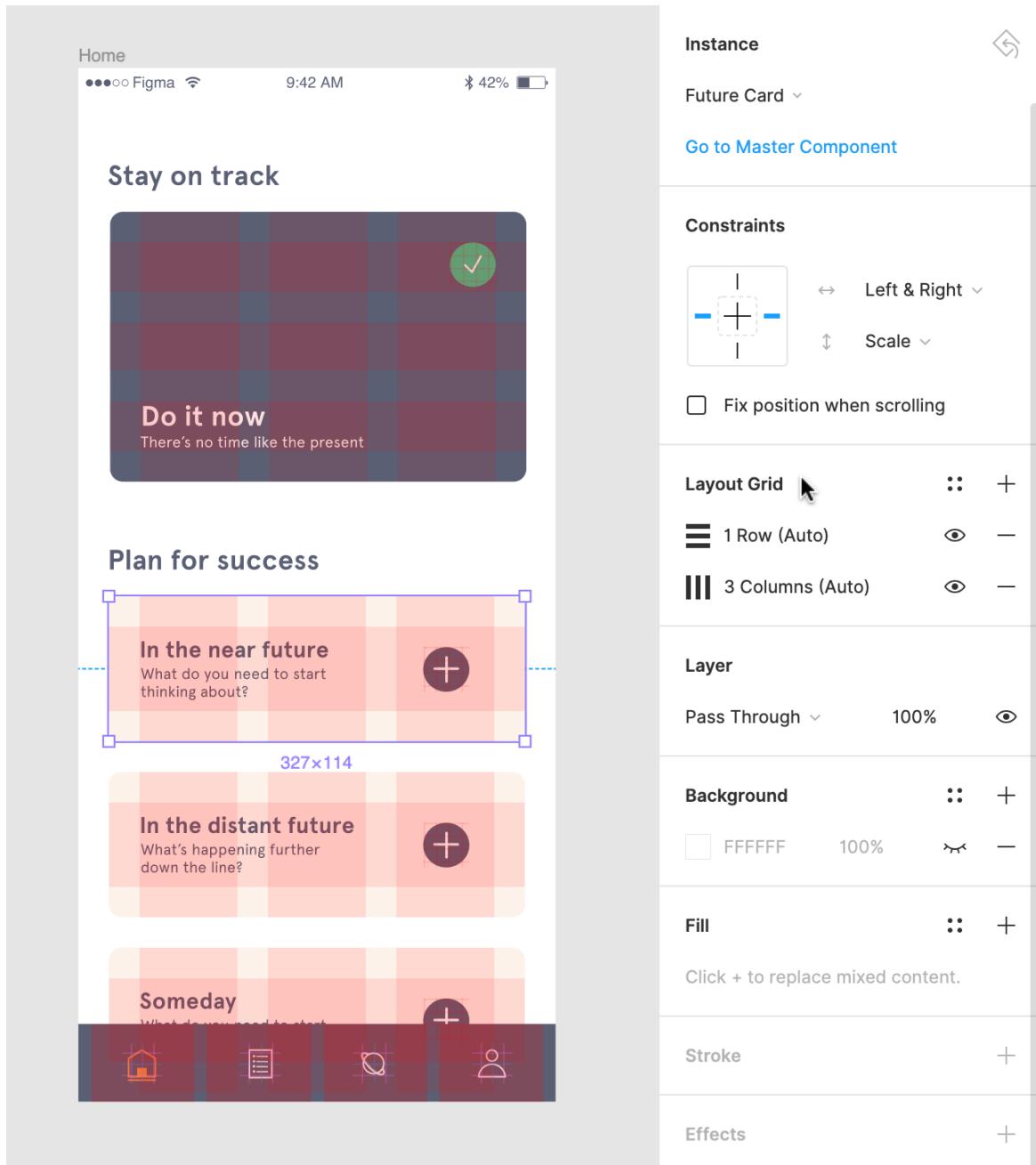


Рисунок 4.29. Комбинированная сетка (layout grid) для компонентов.

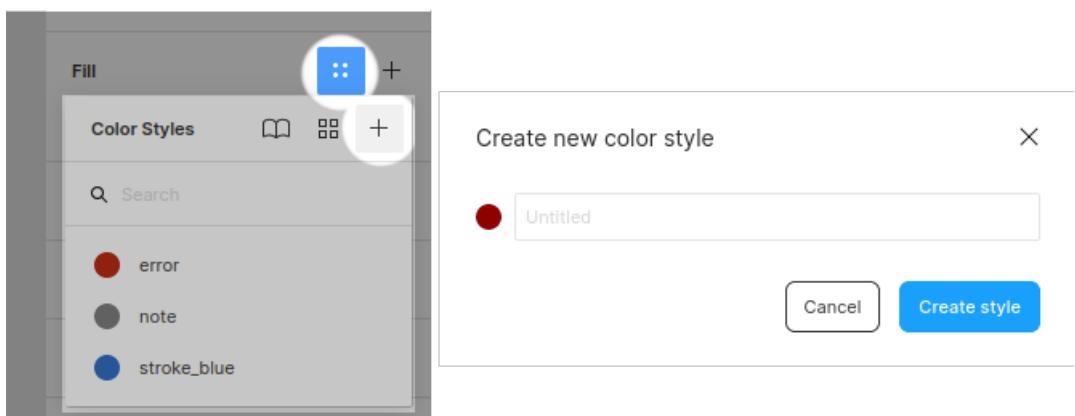


Рисунок 4.30. Панель свойств, создание стиля для цвета.

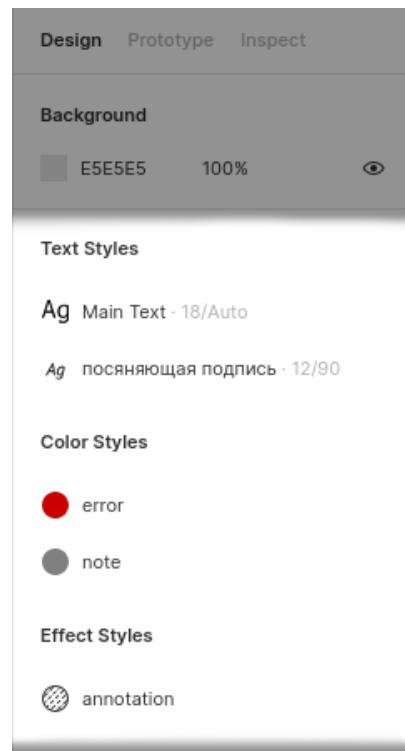


Рисунок 4.31. Перечень существующих стилей на панели свойств проекта (показывается когда не выделено ни одного объекта).

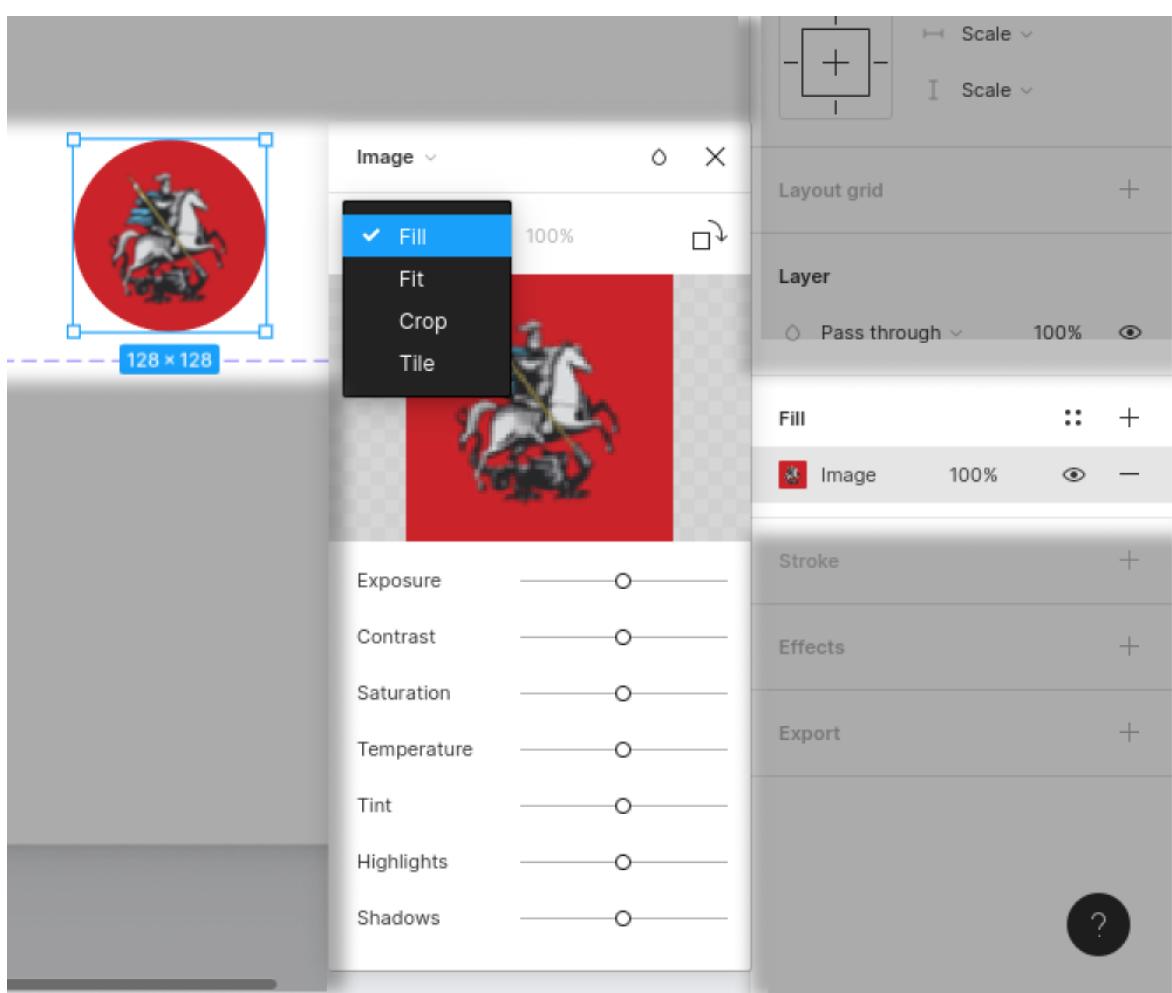


Рисунок 4.32. Изображения в Figma настраиваются через свойство Fill, которое также отвечает за заполнение других объектов цветом. Виды расположения изображения: Fill – заполнение, Fit – заполнение с масштабированием, Crop – заполнение с обрезкой, Tile – заполнение с повторением (плиточное заполнение).

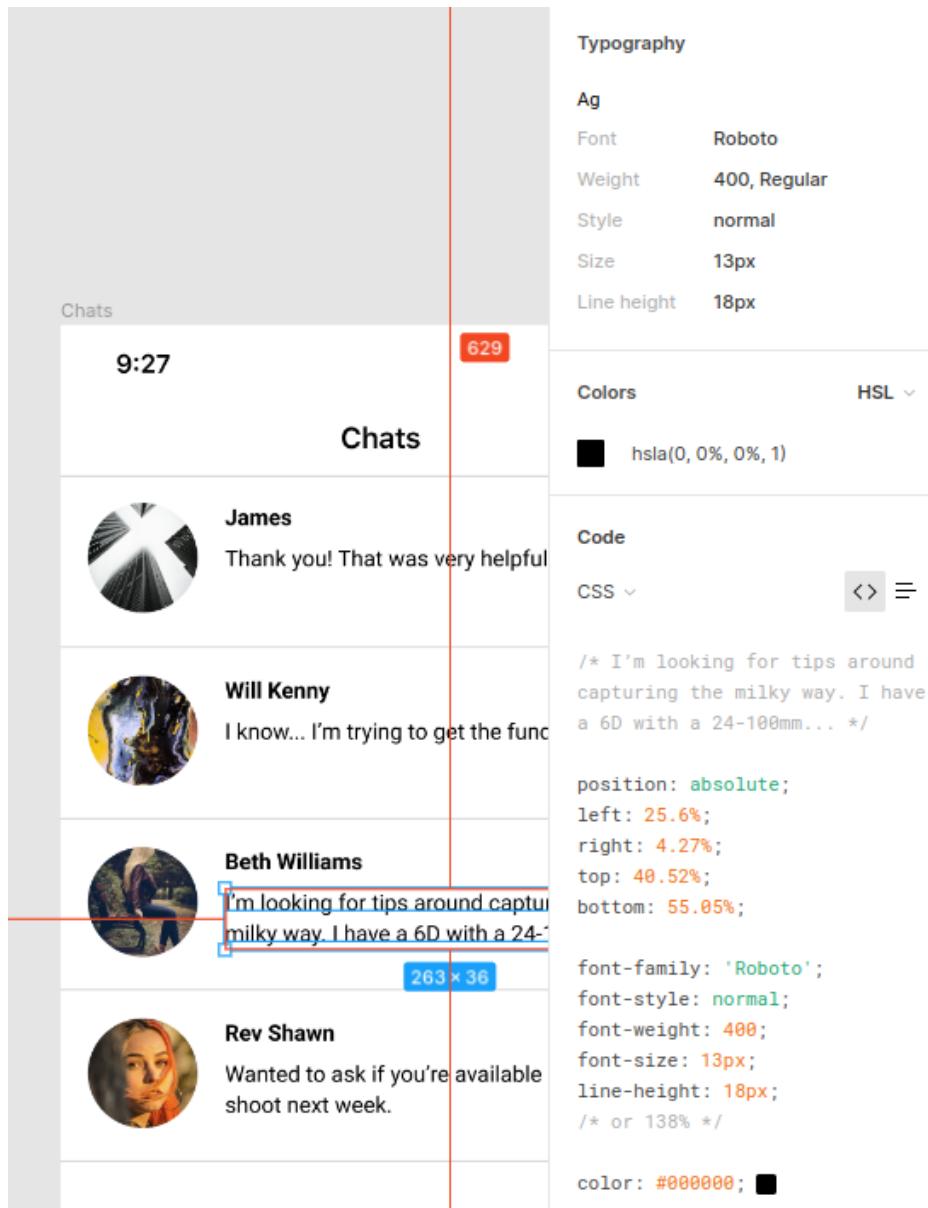


Рисунок 4.33. Figma создаёт CSS код для описания свойств каждого объекта.

Помимо режима демонстрации макета (кнопка share, рис. 4.19) 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 представления изменений.

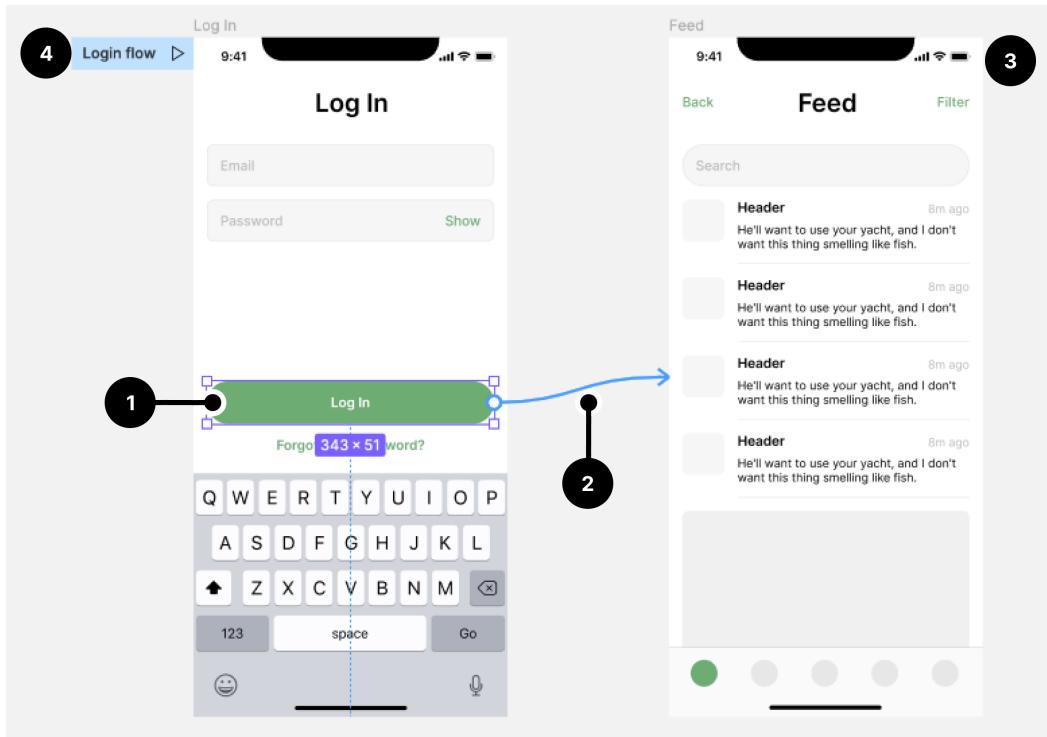


Рисунок 4.34. Режим редактирования прототипа: создание перехода на новый фрейм по клику на кнопку.

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.

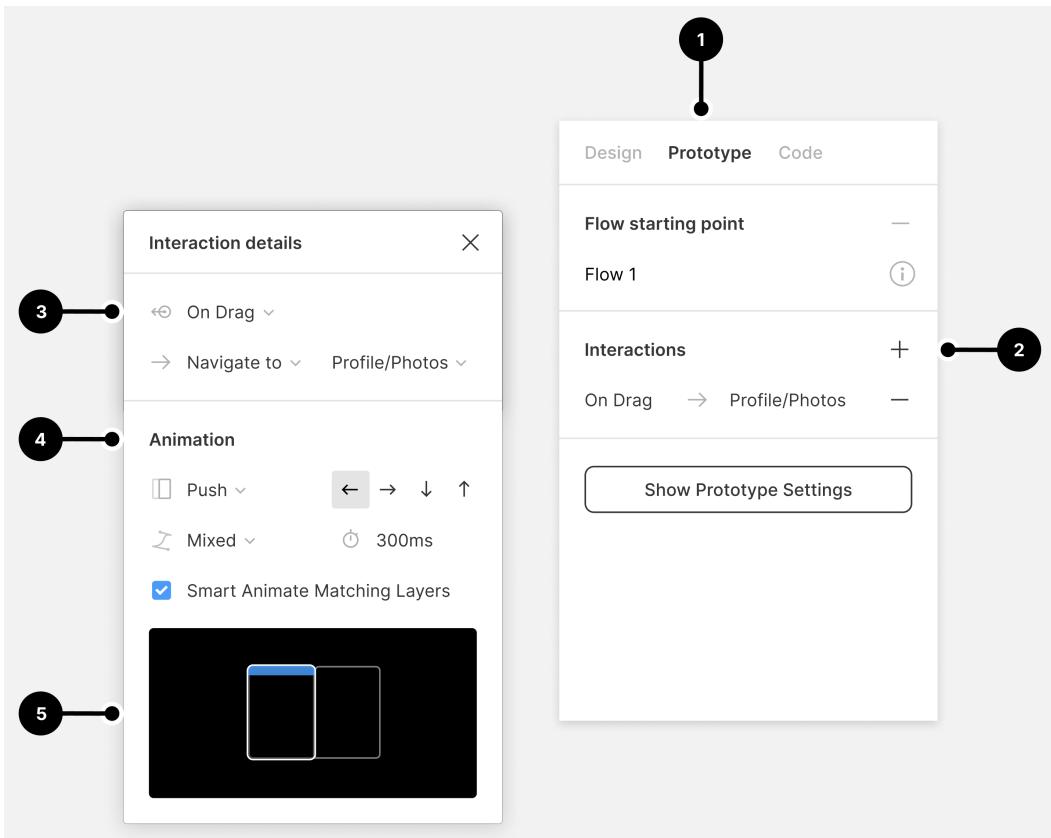


Рисунок 4.35. Режим редактирования прототипа, окно свойств с настройками анимации и целевого действия для перехода на фрейм.

- Нежелательно, чтобы в компонентах был какой-либо осмысленный контент. Их нужно делать максимально обезличенными. Плохо, когда в компоненте кнопки написано «Сохранить» и хорошо, когда «Кнопка».
- Не следует использовать мастер 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.

### 4.3 Методология проектирования

Программы или сайты, решаяшие 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. 4.36).

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 – проектирование идёт сверху вниз, от проектирования наиболее абстрактных аспектов (например, набора прецедентов) к наиболее конкретным (отдельным элементам пользовательского интерфейса). Более детальная схема (рис. 4.37) всех этапов

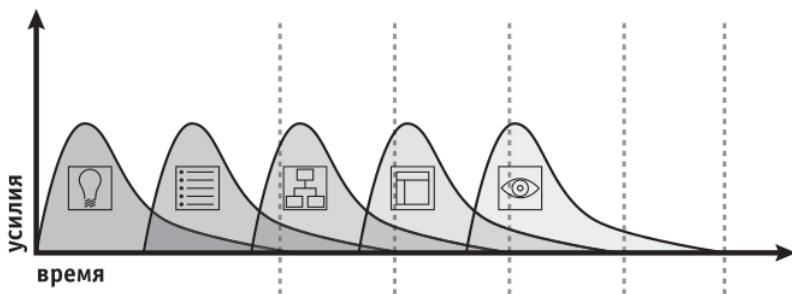


Рисунок 4.36. Каждый следующий этап проектирования должен начинаться до полного завершения предыдущего.

проектирования дана, например, Аланом Купером в книге Основы проектирования взаимодействия [10].

## 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 такое компонент (Figma)? Как его создать? Зачем их использовать?

14. Как влияет изменение свойств компонента на его экземпляры?
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 распределяться 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



Целеориентированное проектирование				
	Активности	Объекты внимания	Участие лиц, принимающих решения	Сдаваемый результат
Исследования	<b>Охват</b> Определение целей и графика проекта	Задачи, сроки, финансовые ограничения, процесс, контрольные точки	Совещания Оценка технических возможностей и затрат	Документ Отчет о проделанной работе
	<b>Аудит</b> Изучение доступных документов и существующих продуктов	Бизнес-планы и маркетинговые планы, стратегия брендинга, маркетинговые исследования, планы развития продуктовой линейки, конкуренты, технологии в соответствующей области		
	<b>Интервью с лицами, принимающими решения</b> Прояснение образа продукта и имеющихся ограничений	Образ продукта, риски, благоприятные возможности, ограничения, логистика, пользователи	Интервью С лицами, принимающими решения, и пользователями	
	<b>Интервью и наблюдения за пользователями</b> Прояснение потребностей пользователей и изучение их поведения	Пользователи, потенциальные пользователи, модели поведения, взгляды, способности, мотивы, характеристики окружения, инструменты, проблемы	Приемка Предварительных результатов исследований	
Моделирование	<b>Персонажи</b> Создание архетипов пользователей и клиентов	Шаблоны поведения пользователей и клиентов, взгляды, способности, цели, характеристики окружения, инструменты, проблемы	Приемка Персонажей	Документ
	<b>Прочие модели</b> Представление особенностей предметной области, не связанных с отдельными пользователями и клиентами	Рабочие процессы, затрагивающие группы людей, характеристики окружения, артефакты		
Выработка требований	<b>Контекстные сценарии</b> Сочинение историй об идеальном опыте пользователей	Как продукт соответствует жизни и среде персонажей и помогает им в достижении целей	Приемка Сценариев и требований	Презентация Анализ пользователей и предметной области
	<b>Требования</b> Определение критических возможностей продукта	Функциональные и информационные потребности, ментальные модели пользователей, императивы проектирования, образ продукта, бизнес-требования, технология		
Проектирование инфраструктуры	<b>Элементы</b> Описание информационной и функциональной модели	Информация, функции, механизмы, действия, объектные модели предметной области	Приемка Общей структуры проекта	Документ Анализ пользователей и предметной области
	<b>Инфраструктура</b> Проектирование общей структуры опыта взаимодействия	Связи между объектами, концептуальные группы, навигационные последовательности, принципы и шаблоны, поток управления, эскизы, раскаровки		
	<b>Ключевые и проверочные сценарии</b> Описание взаимодействия персонажа с продуктом	Как дизайн продукта соответствует идеальной последовательности действий пользователя и учитывает разнообразие вероятных условий		
Детализация	<b>Детальное проектирование</b> Доработка и уточнение деталей	Внешний вид, идиомы, интерфейс, виджеты, поведение, информация, визуализация, бренд, опыт, язык, раскаровки	Приемка Детального проекта	Документ Спецификация формы и поведения
Сопровождение проектирования	<b>Корректировка спецификации</b> Учет новых ограничений и сроков	Поддержание концептуальной целостности дизайна продукта в условиях меняющихся технологических ограничений	Совместное проектирование	Пересмотренный документ Спецификация формы и поведения

# 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 критерииев, которые помогают оценить юзабилити продукта. Приведём показатели Шнейдермана [49], 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<sup>1</sup>) Яакоба Нильсона, помогающих добиться хорошего юзабилити.

---

<sup>1</sup>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.

The screenshot shows a survey form with the following fields and their values:

- Имя (обязательное поле): Константинов
- Отчество (обязательное поле): Константин
- Не имею отчества
- Пол (обязательное поле):
  - Женский
  - Мужской
- Дата рождения (обязательное поле): 09 / 02 / 1980

A modal window titled "Внимание" (Attention) is displayed, containing the message: "В анкете должны быть заполнены все обязательные поля" (All mandatory fields must be filled in). A button labeled "Понятно" (Understood) is at the bottom of the modal.

Рисунок 5.1. Плохая видимость статуса системы. Анкета на сайте [leadersofdigital.ru](http://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)

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 курсор); режим кисти и резинки в графическом редакторе.

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

\* = Required

* First Name:	<input type="text" value="Chuck"/>	
* Last Name:	<input type="text" value="Woolry"/>	
Business Name:	<input type="text"/>	
*Street Address: (no P.O. Boxes)	<input type="text"/>	
Suite/Apt:	<input type="text"/>	
*City:	<input type="text" value="Hollywood"/>	
*State:	CA	CALIFORNIA
*Zip Code:	<input type="text" value="90210"/>	
*Phone number:	<input type="text" value="(310) 123-3234"/>	

APO may incur additional shipping charges.

Рисунок 5.2. 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.

ему терминологию и концепции.

- Не используйте специальные термины из ИТ: авторизация (вход), транзакция (операция, запрос), лейбл (надпись), ...
- Страйтесь 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).

**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 плат-

The screenshot shows a user interface for entering shipping information. At the top, there's a placeholder address: '5120 CHESTNUT STREET'. Below it, there are three input fields: 'CITY' (containing 'San Francisco'), 'STATE' (containing 'California'), and 'ZIP' (containing '9'). Underneath these, a section titled 'CONTACT INFO' contains a red-bordered input field for 'BILLING PHONE NUMBER'. Inside this field is a small character 'I'. Below the input field, a message reads: 'If we have order questions' followed by a red error message: 'Oops. Looks like you forgot your Billing Phone Number.'

Рисунок 5.3. Незаполненное поле выделено. Подсказка дана снизу.  
Нужно немного увеличить размер текста подсказки.

формы.

- Используйте одни и те же слова и обозначения для одних и тех же действий.
- Следуйте дизайн-системе<sup>2</sup>
- 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

<sup>2</sup>Дизайн-система — набор компонентов, правил, предписаний по дизайну

## Заполните форму и мы свяжемся с Вами

Ваше имя\*

Ошибка! Поля обязательные для заполнения.

Адрес эл.почты\*

Ошибка! Поля обязательные для заполнения.

Номер телефона

Введите ваше сообщение

Ошибка! Поля обязательные для заполнения.

\* - обязательное поле

Рисунок 5.4. 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.

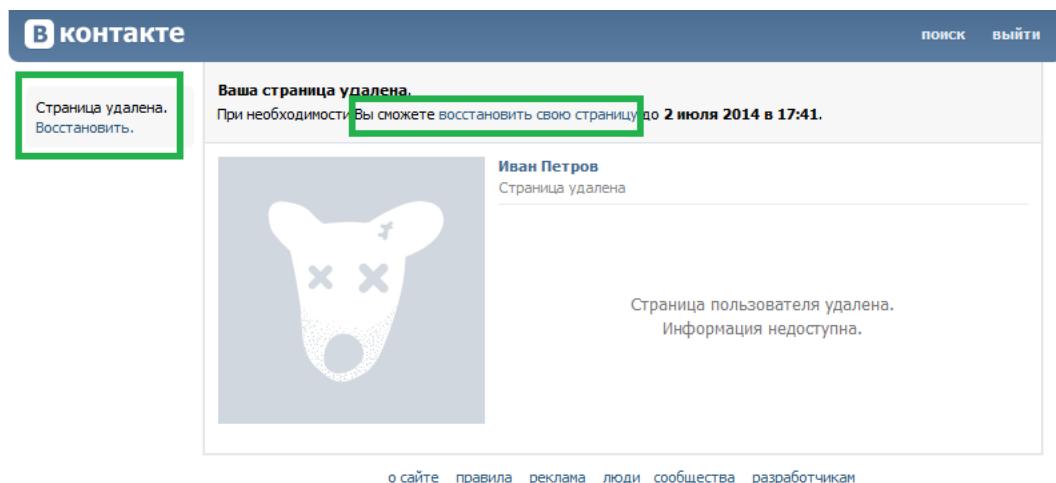


Рисунок 5.5. vk.com откладывает удаление данных на некоторое время, чтобы у пользователя была возможность отменить действие.

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<sup>3</sup>. Другой источник промахов – неточные движения мыши и случайные нажатия клавиш. Например пользователь случайно нажимает на кнопку “Ок”, вместо расположенной рядом<sup>4</sup> 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.

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 (рис 5.7), действия и варианты выбора. Пользователь не должен в одной части диалога запоминать информацию, которая потребуется ему в другой. Антипример приведён на рисунке

<sup>3</sup> см. также параграф 2.4 о режимах

<sup>4</sup> см. параграф о 6.2 прицеливании и законе Фиттса

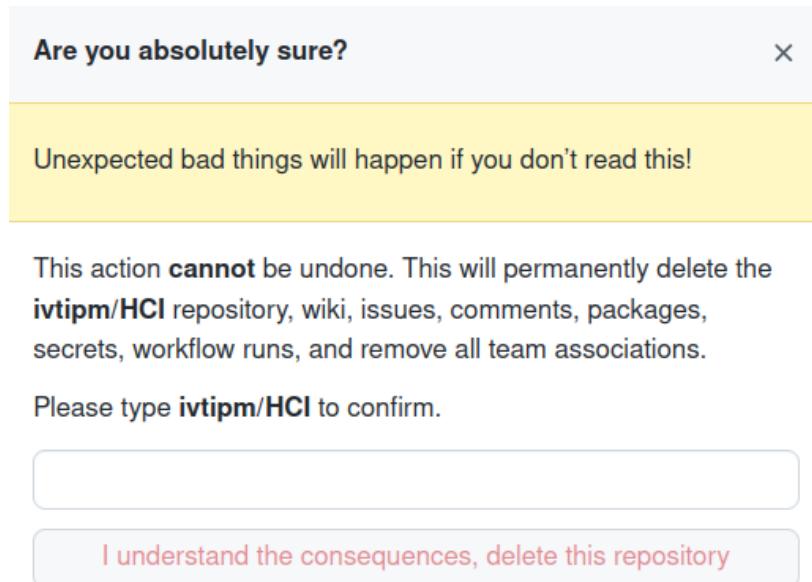


Рисунок 5.6. Github требует ввести название репозитория перед удалением. Это даёт пользователю возможность сделать имя репозитория локусом внимания и шанс остановится перед удалением не того репозитория.

1.7. Инструкции по использованию системы должны быть видимы или легко доступны везде, где возможно.

## 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 отоб-

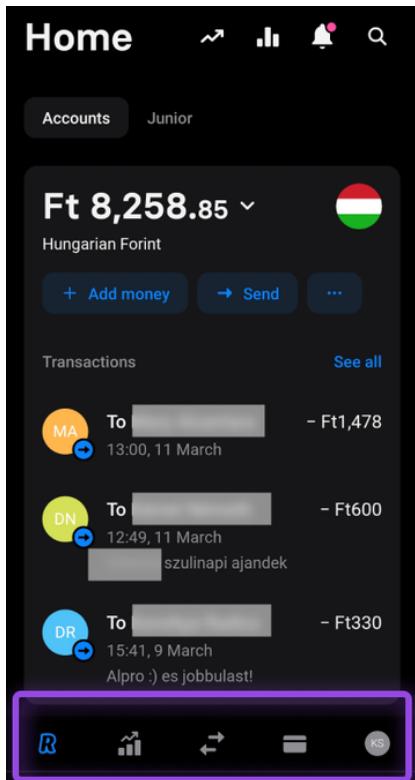


Рисунок 5.7. Значки в нижней части программы не подписаны: пользователю придётся так или иначе вспоминать их назначение. Подписи решили бы эту проблему.

ражаемой информации. При некоторые способы упрощения интерфейса приведено в параграфе 6.5. 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 проводиться 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 сторо-



Рисунок 5.8. А/В 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.

ны, разработчик продукта обычно осознанно или нет вмешивается в действия пользователя, что искажает результаты.

### 5.2.2 А/В-тестирование

Идея А/В-тестирования<sup>5</sup> заключается в следующем:

- Пользователей разделяют на две (или более) группы
- каждой из групп показывай свой вариант (А или В) 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<sup>6</sup>, время проведённое на сайте и др.)

<sup>5</sup>Термин изначально появился в маркетинге

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

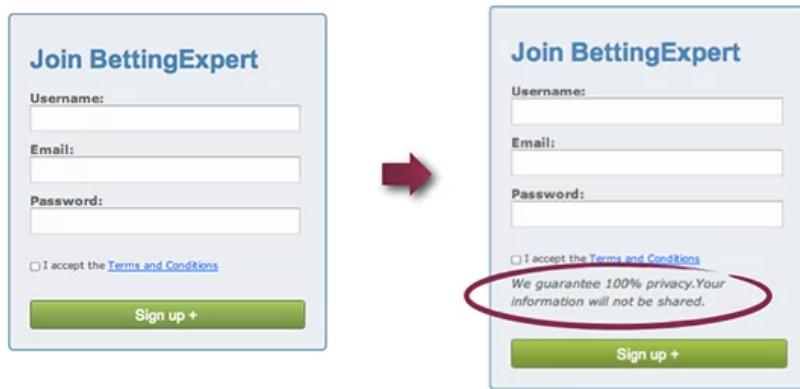


Рисунок 5.9. Два варианта формы регистрации на сайте букмекерской компании. На форме справа добавлено: Мы даём 100% гарантию безопасности ваших данных. Они никому не будут переданы. Результат эксперимента: увеличение количества регистраций на 20% по сравнению с оригинальной формой. Объём выборки: 20257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

### 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.

Наблюдение за тем, как пользователи взаимодействуют с продуктом, нередко позволяет найти для него 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).

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

В качестве пользователей обычно выбираются люди, соответствующие разработанным ранее пользовательским моделям, то есть перв-

сонажам.

**Коридорное тестирование.** На практике далеко не всегда есть возможность пригласить потенциальных пользователей для тестирования. Тем более, если это 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 проклятия знания. Такое тестирование, как правило, не занимает много времени, его можно делать часто.

Стив Круг рекомендует привлекать для тестирования до трёх человек [25]. Они позволяют 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.

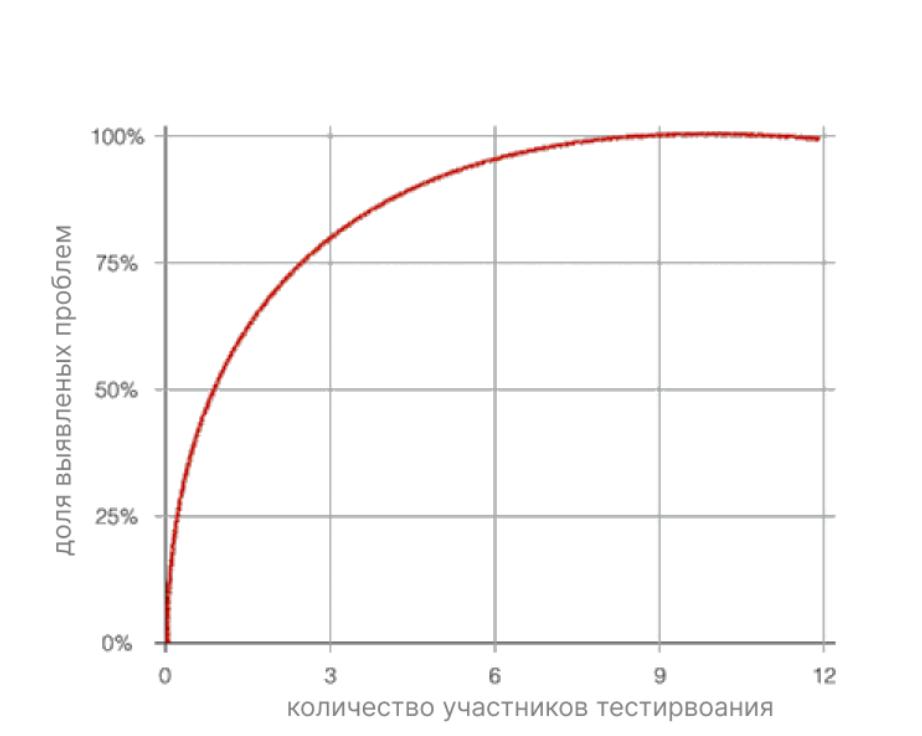


Рисунок 5.10. Небольшое количество участников юзабилити тестирования позволяют выявить значительную часть проблем [69].

## 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 Яакоба Нильсона. Приведите примеры и антипомодели.
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 преимущества?

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<sup>1</sup> 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

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

---

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

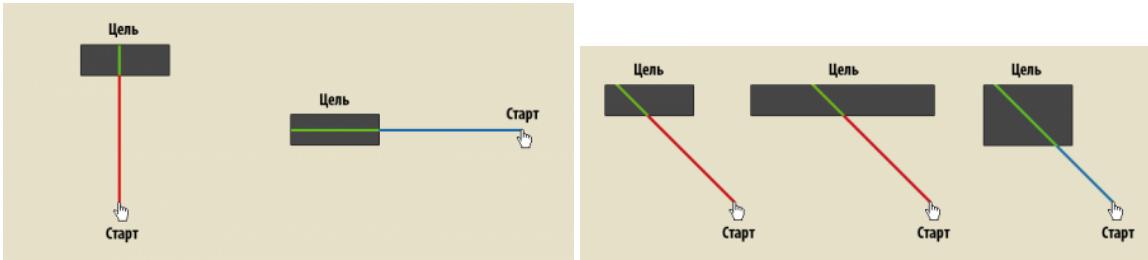


Рисунок 6.1. Закон Фитса: время указания на цель зависит от дистанции до цели (красная и синяя линии) и размера цели вдоль оси движения (зелёная линия). Увеличение ширины цели не всегда даёт выигрыша во времени. Нужно учитывать либо все направления движений, либо самые типичные [66].

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).

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.

Делать все элементы интерфейса большими неразумно. Тем бо-

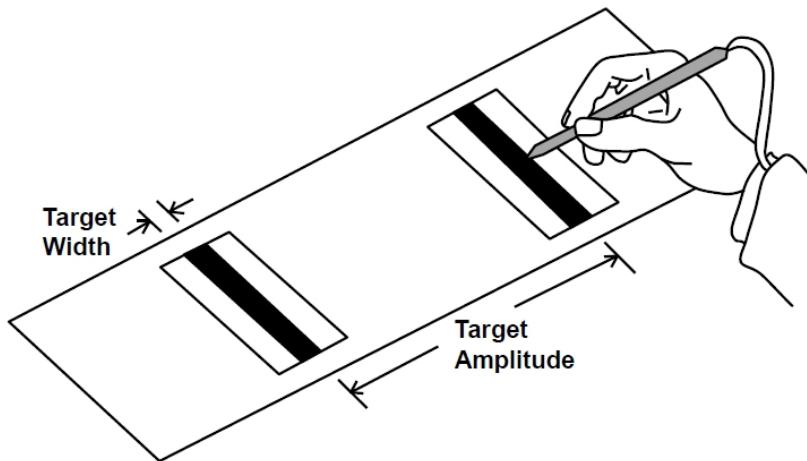


Рисунок 6.2. Оригинальная постановка эксперимента, по результатам которого был описан закон Фиттса. Участник эксперимента должен был как можно быстрее касаться стилусом двух металлических пластин. Эксперимент повторялся для пластин разной ширины и с разным расстоянием между ними [67].

лее, 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.

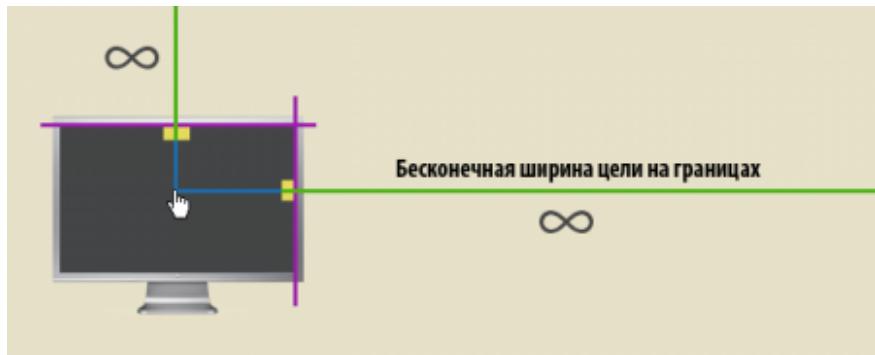


Рисунок 6.3. Правило бесконечной границы. Так как нельзя промахнуться мимо края экрана, ширина или высота цели в направлении границы экрана считается бесконечной.

Правило бесконечной границы работает для панели задач операционной системы Windows (рис. 6.5). Кнопки закрытия окна расположены в правом углу экрана, когда окно развернуто. Вкладки в большинстве браузеров находятся прямо на заголовке окна, то есть возле верхнего края экрана, когда окно развернуто. Полоса прокрутки во многих приложениях расположена в правой части окна, что даже даёт преимущество, когда окно развернуто.

В семействе операционных систем Mac OS в верхней части экрана традиционно располагается панель меню запущенной программы (рис 6.6).

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 ( 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 началь-

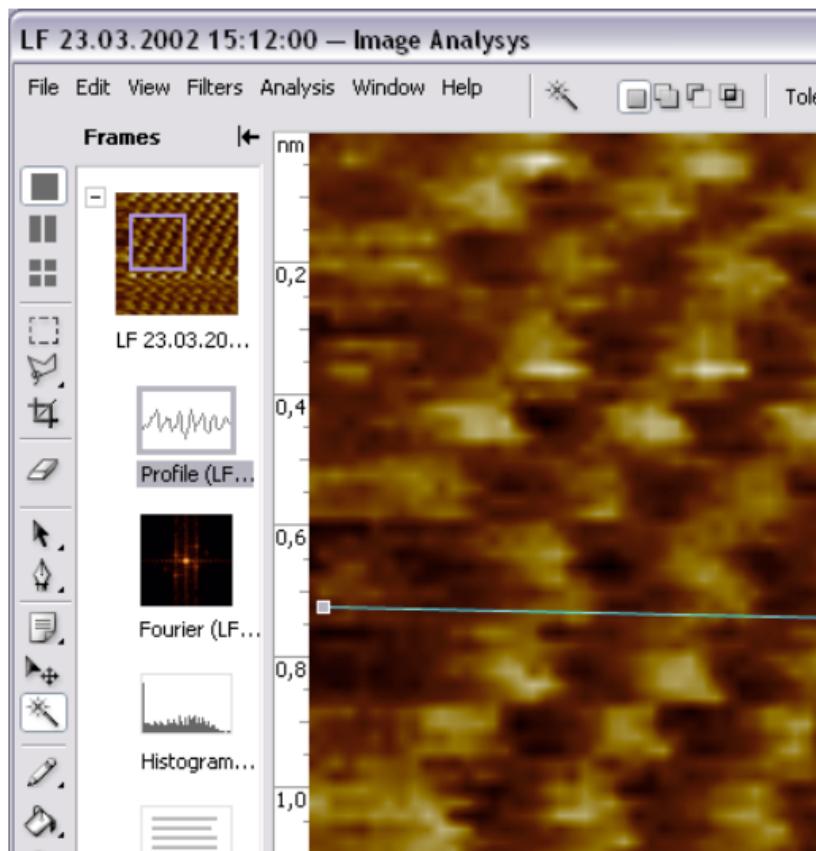


Рисунок 6.4. Правило бесконечной границы: когда окно развернуто во весь экран, кнопки на левой панели инструментов имеют бесконечную ширину (если область нажатия доходит до левой границы окна) [7].



Рисунок 6.5. 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.



Рисунок 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.

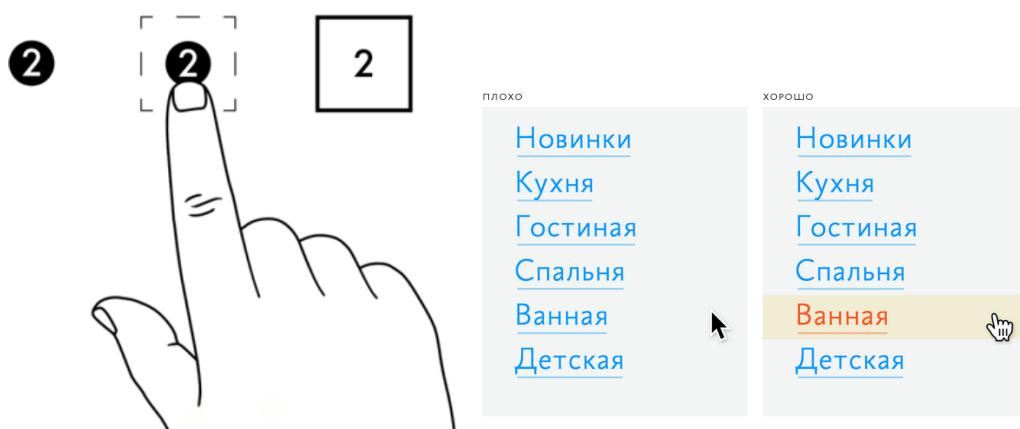


Рисунок 6.7. Если объект нельзя увеличить в размерах, то можно попробовать увеличить невидимую область нажатия вокруг него. Примеры из [bureau.ru/books/ui/demo/4](http://bureau.ru/books/ui/demo/4)

[8]

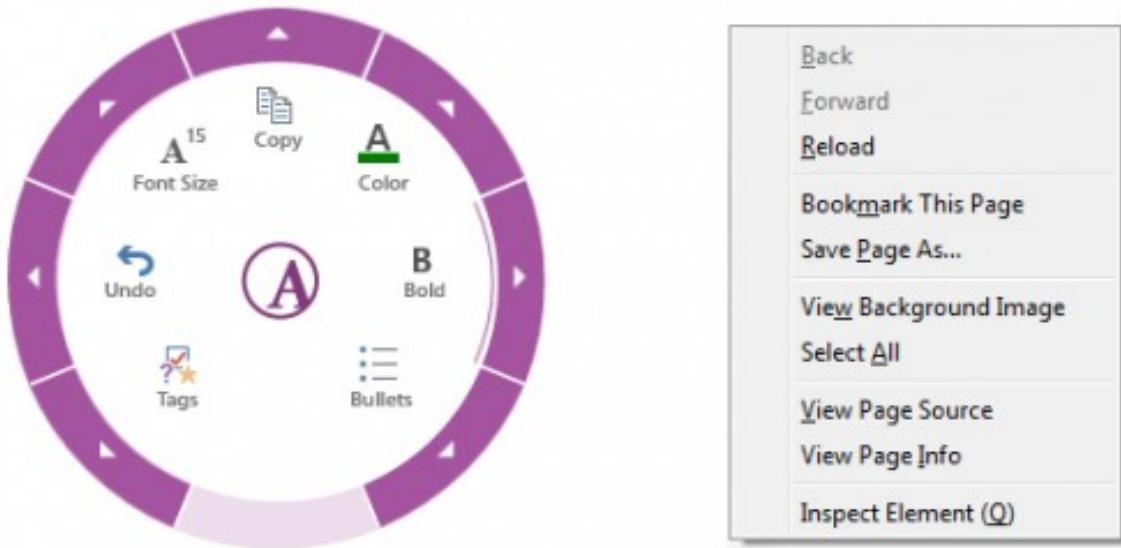


Рисунок 6.8. Круговое контекстное меню. Время указание на все элементы, в отличие от классического контекстного меню, одинаково, а сами элементы больше. Классическое контекстное меню. Самые часто используемые пункты должны быть расположены ближе к указателю мыши (выше)

ной позиции курсора. В круговом контекстном меню оптимизируется и расстояние до всех элементов их размер.

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 нужно 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.

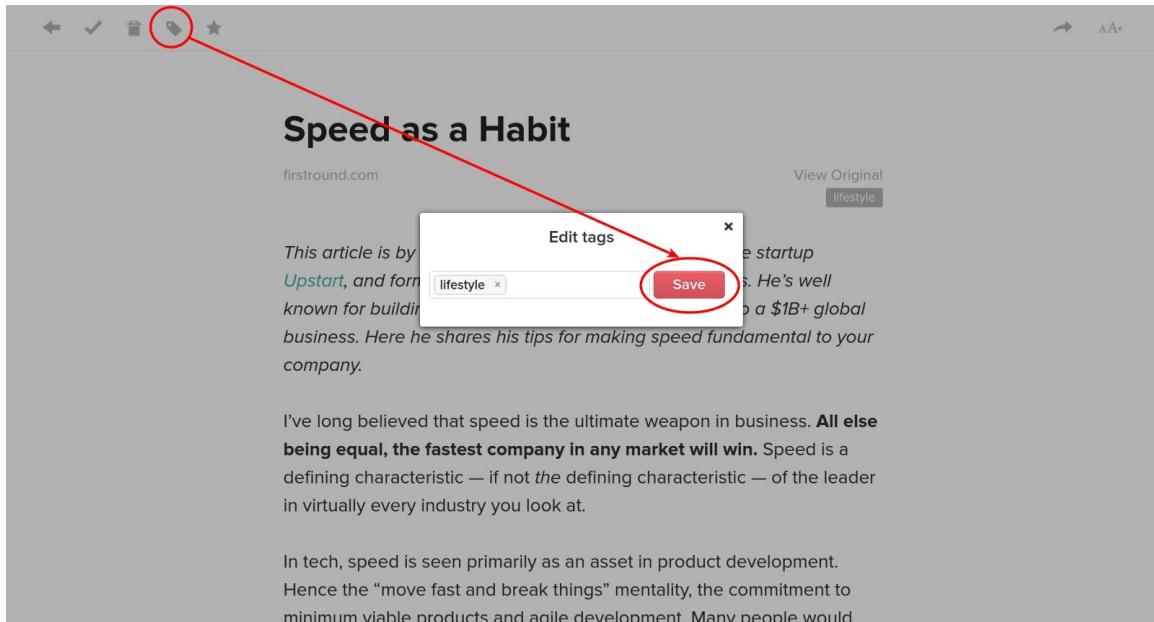


Рисунок 6.9. Ожидается, что кнопки должны быть нажаты последовательно, однако они разнесены на большое расстояние.

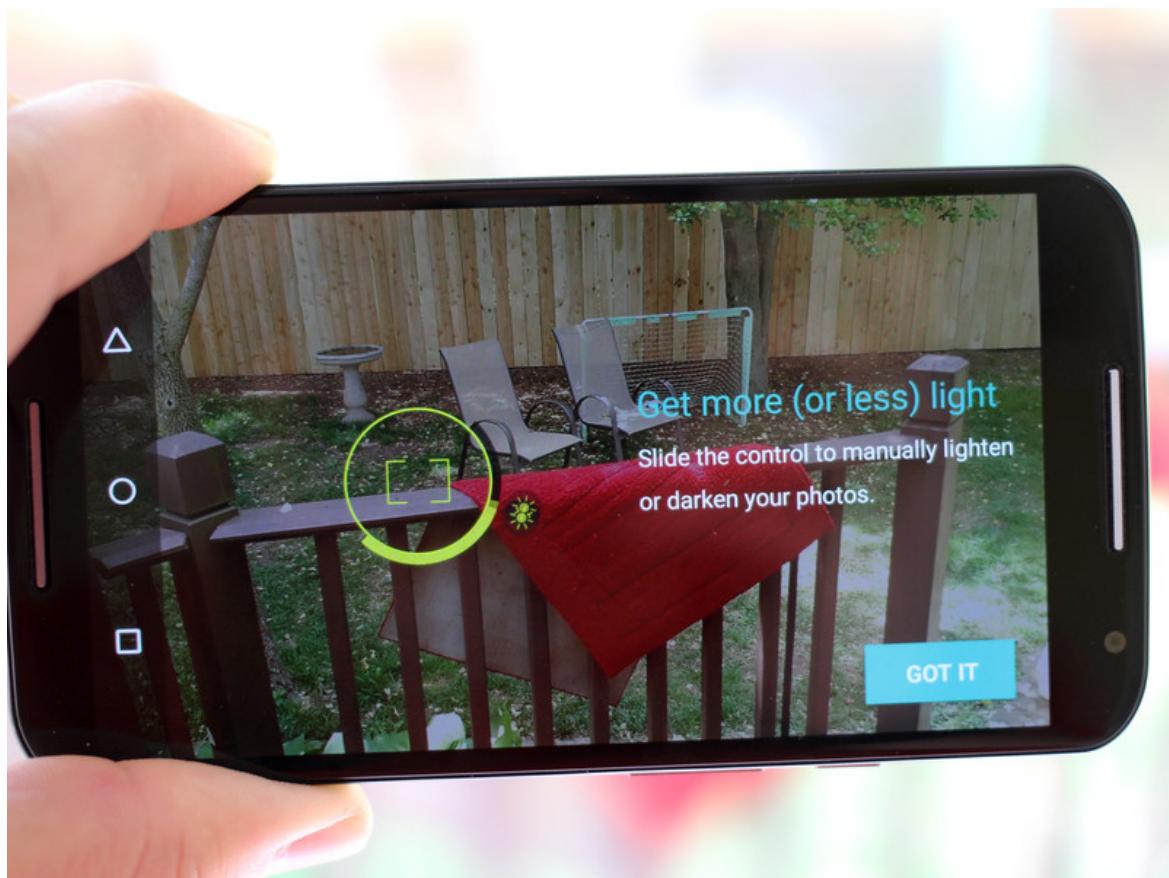


Рисунок 6.10. Расстояние до цели равно нулю. Кнопка появляется в месте тапа.

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

$$T = a + b \cdot \log_2 (n + 1), \quad (6.2)$$

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. За-

кон 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

$$T = a + b \sum_i^n p_i \log_2(1/p_i + 1), \quad (6.3)$$

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<sup>2</sup>. 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<sup>3</sup>. 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.

<sup>2</sup>о том, как придумать перечень категорий или разделов, например, на сайте, читайте в параграфе про метод карточной сортировки

<sup>3</sup>см. также параграф про информационную эффективность интерфейса

Что может ухудшить пользовательский опыт.

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 по алфавиту. Тогда пользователь будет иметь примерное представление где искать нужный 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<sup>4</sup> на клавиатуру и обратно.
- **M** = 1.35 с. Ментальная подготовка. Время необходимое чтобы умственно подготовиться к следующему шагу.
- **R**. Ответ. Ожидание ответа компьютера<sup>5</sup>.

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.

Вычисления времени, необходимого на выполнение действия, с помощью модели GOMS начинаются с перечисления операций из списка жестов модели GOMS. Затем расставляются ментальные операторы согласно правилам. Потом, ментальные операторы удаляются согласно правилам:

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

Операторы M следует устанавливать перед всеми K (нажатие клавиши) и P (указание с помощью ГУВ), предназначенными

<sup>4</sup>графическое устройство ввода данных, например, мышь.

<sup>5</sup>анализ времени отклика компьютеров: 1977-2017

для выбора команд; но перед операторами Р, предназначенными для указания на аргументы этих команд, ставить оператор М не следует.

- **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

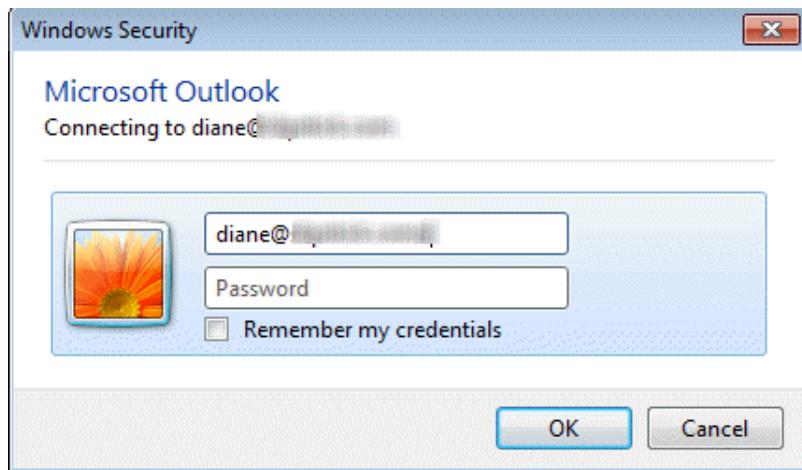


Рисунок 6.11. Окно авторизации в Microsoft Outlook. Пример для оценки времени решения задачи пользователем по модели GOMS.

ним. (Добавление разделителя станет привычным действием, и поэтому разделитель станет частью строки и не будет требовать специального оператора M.) Но если оператор K является разделителем 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. 6.12). 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.

Некоторые способы ввода данных с точки зрения GOMS занимают примерно одинаковое время. Например, задание значения чис-

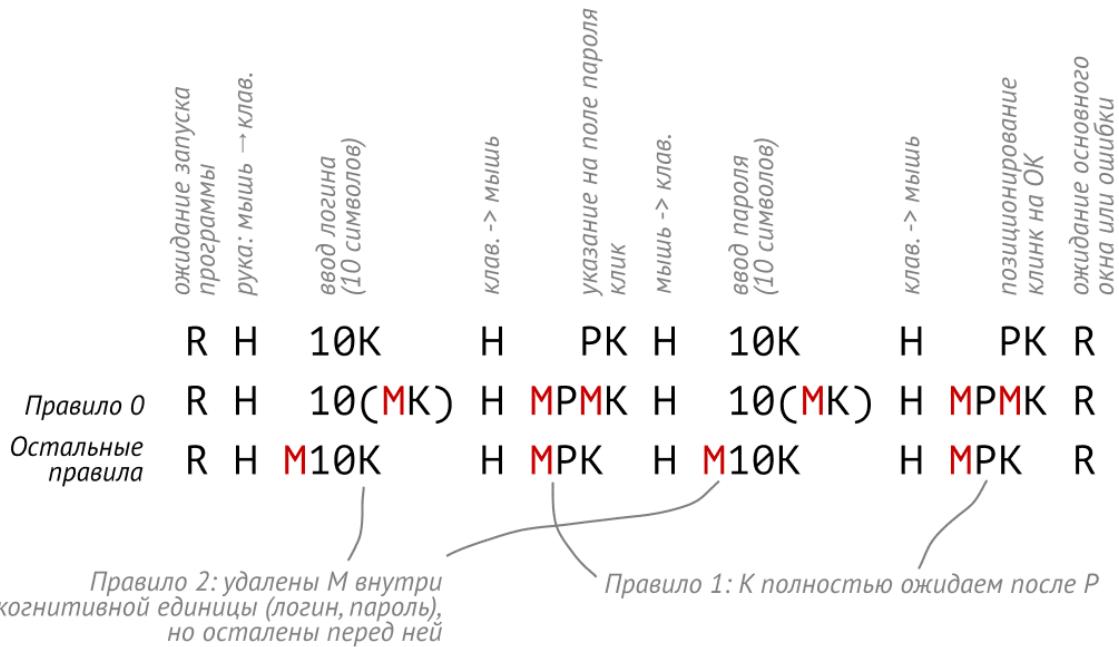


Рисунок 6.12. Начальная запись жестов модели GOMS для окна ввода 6.11; применение правила 0 для расстановки когнитивных операторов; удаление когнитивных операторов по правилам 1-5.

лового параметра с помощью ползунка займёт 2.4 секунды (PKPK): навести указатель, нажать клавишу мыши, навести на новое значение, отпустить клавишу. Ввод трёхзначного числа в поле клик по полю ввода займёт 2.3 секунды (PK-H-KKK). 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).

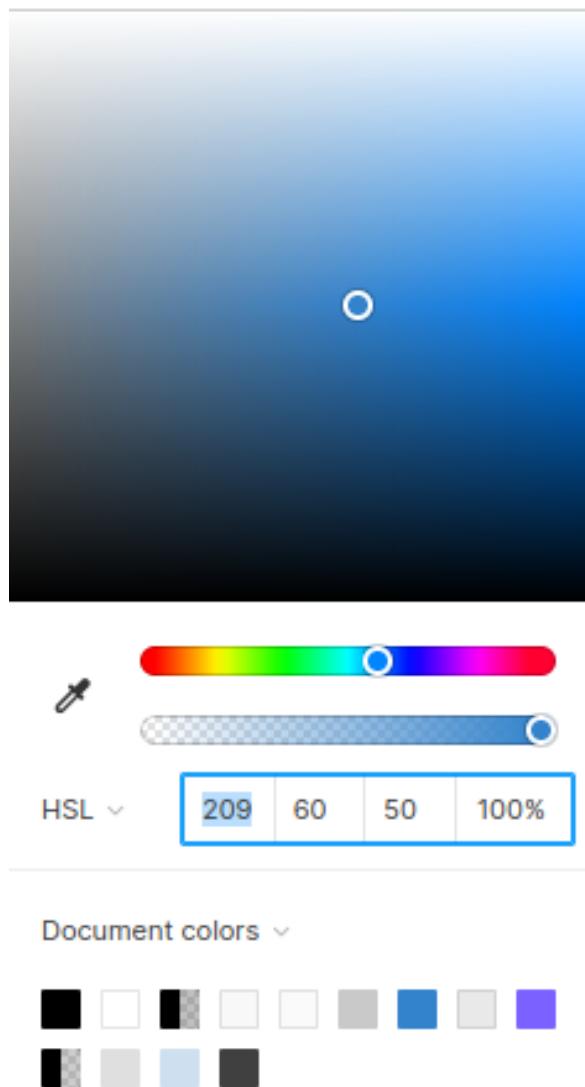


Рисунок 6.13. Выбор цвета в Figma. Пользователь может воспользоваться ползунком для быстрого выбора оттенка (Hue) и палитрой для выбора насыщенности и светлоты (saturation и value). Для точного задания этих параметров можно ввести значения в поля ввода. Список использованных ранее цветов (document colors) 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 совершение операций. Другая характеристика пользовательского интерфейса – количе-

ство 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 (рис. 6.14), создавая меньше "визуального мусора" и добавлять смысловую нагрузку существующим элементам интерфейса (рис. 6.15).

Один из приёмов повышения информативности — вынос за скобки. Повторяющуюся информацию (слова, пиктограммы и т.п.) в перечных приводят один раз, вынося на один уровень иерархии выше (рис. 6.16, 6.16). На рис. 3.14 первая колонка таблицы содержит повторяющиеся значения. В улучшенной версии таблицы (рис. 3.15) 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  $I_{\min}$ , необходимого для выполнения задачи, к количеству информации  $I_{\text{введ.}}$ , которое фактически требуется ввести от пользователя.

$$E = \frac{I_{\min}}{I_{\text{введ.}}} \quad (6.4)$$

Дополнительно рассмотрим пограничные случаи, не учитывающие 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).

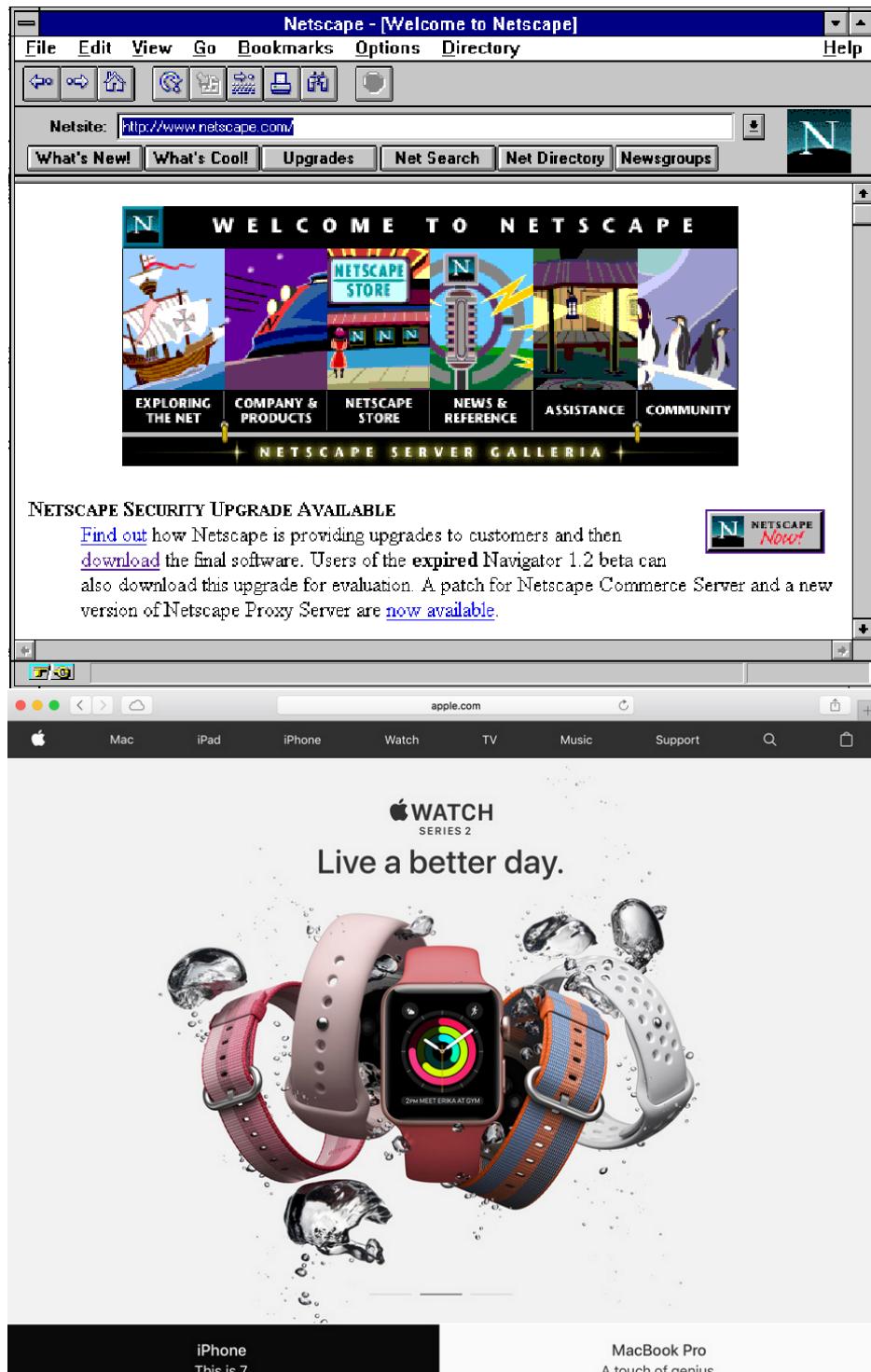


Рисунок 6.14. Интерфейс старых браузеров содержал множество видимых элементов интерфейса (вверху, браузер NetScape Navigator). Современные браузеры, при возросшем разрешении дисплеев, (внизу, Safari, 2017) скрывают меню, многие кнопки скрыты или расположены компактно вместе с адресной строкой, полоса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



Рисунок 6.15. Примеры повышение информативности интерфейса: краткое описание изменений в сообщении о новой версии программы. Объединение кнопки "Купить" и цены товара.

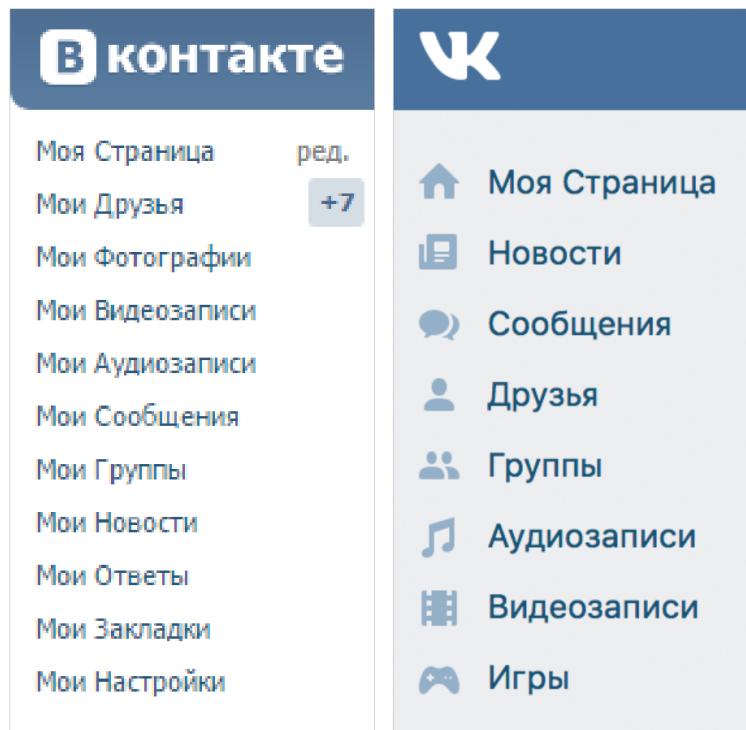


Рисунок 6.16. Слово “Мои“ можно вынести за скобки.

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](#)

Рисунок 6.17. Улучшения предложения продолжить поиск (верхний вариант, такой блок приводился ниже результатов поиска на странице yandex.ru) в два этапа: вынос за скобки “Поискать”, замечена обозначения данных “то же самое” на сам поисковый запрос.

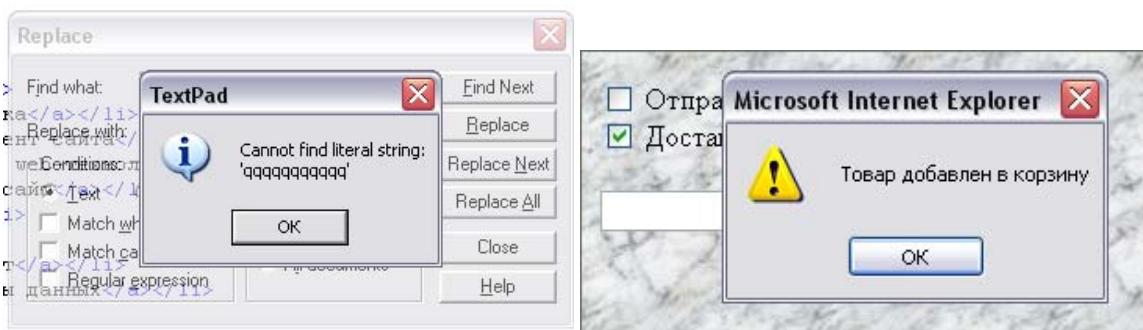


Рисунок 6.18. Нулевая информационная эффективность: данных для ввода не требуется, но пользователь должен ввести один бит информации нажав на кнопку OK.

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 и тексты

## 7.1 Понятие типографики

Текст – это около 90% всей визуальной информации, которую получает человек. Большая часть информации в пользовательских интерфейсах – тоже текст (рис. 7.1). Поэтому важно понимать, как доставлять 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), неудачная комбинация большого количества шрифтов, концентрация на

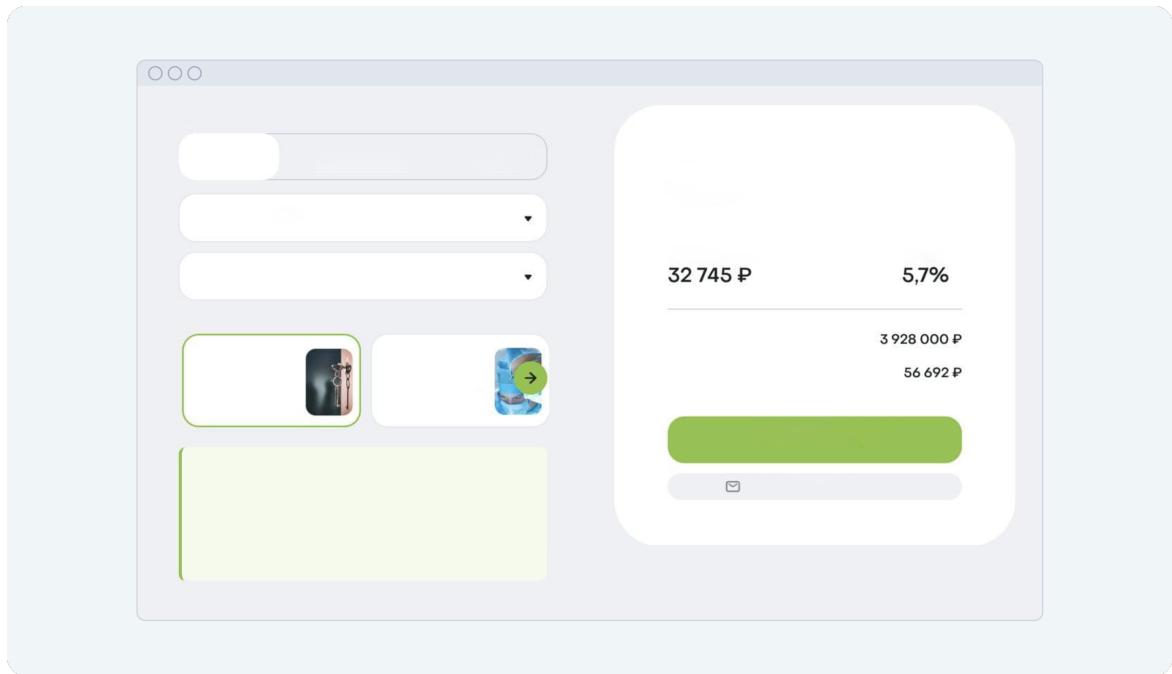


Рисунок 7.1. Графический пользовательский интерфейс без слов чаще всего невозможен.

эстетической составляющей в ущерб читаемости.

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

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.

## 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, готеск).

**Legibility**  
is how well you  
**see the letters.**

**Readability**

is how easily you read the words, as in long passages of text. There are very different requirements in each case, depending on the visibility of the text and the level of experience of the reader.

Рисунок 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.

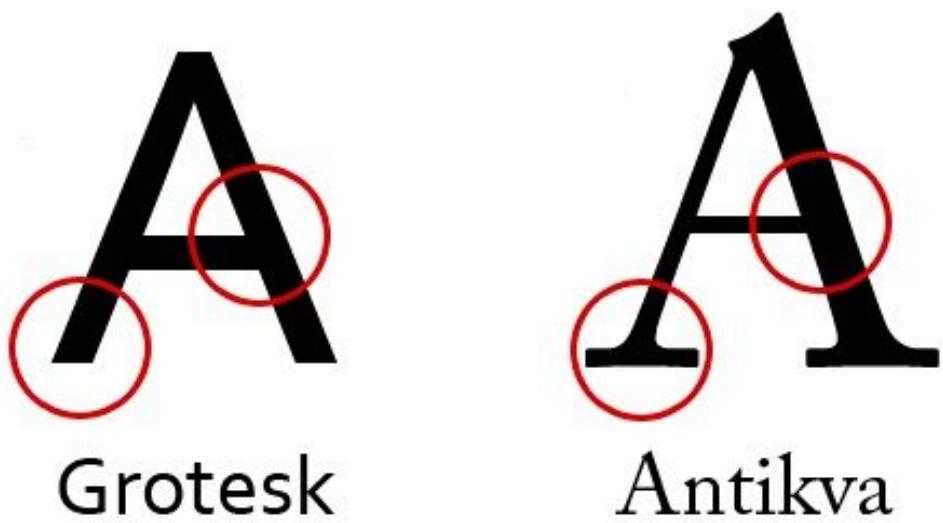


Рисунок 7.3. Гротеск – sans serif – шрифт без засечек. Антиква – serif – шрифт с засечками. Антиковенные шрифты, как правило, имеют разную толщину основного и дополнительного штриха, в отличие от гротеска.

Helvetica  
Arial  
Roboto  
San Francisco  
PT Sans

TRAJAN  
Baskerville  
Times New Roman  
Georgia  
PT Serif

Рисунок 7.4. Слева – шрифты с засечками, справа – без засечек.

Нет убедительных доказательств того, что антиква или готеск 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 антику для печатных текстов, готеск – 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.

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 предназна-

# Handgloves

# Handgloves

# Handgloves

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



Рисунок 7.6. Акцентные шрифты своим характерным видом часто призваны подчеркнуть общий стиль дизайна.

чены 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.

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

- Пропорциональные;
- Монотипичные.

# Proportional Monospaced

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

Roboto  
1 1 | 0 0 o  
Source code pro  
1 I l | 0 0 o

Рисунок 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.

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 3.15).

Наконец приведём наименее формальную классификацию шрифтов: нейтральные и характерные. 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,

```
[1..100]
|> List.map (fun x ->
  match (x % 3, x % 5) with
  | (0, 0) -> "FizzBuzz"
  | (0, _) -> "Fizz"
  | (_, 0) -> "Buzz"
  | (_, _) -> x |> string
)

```

```
[1..100]
▷ List.map (fun x →
  match (x % 3, x % 5) with
  | (0, 0) → "FizzBuzz"
  | (0, _) → "Fizz"
  | (_, 0) → "Buzz"
  | (_, _) → x ▷ string
)

```

Рисунок 7.9. Моноширические шрифты используются в редакторах кода, где важно выравнивание кода. Некоторые моноширические шрифты имеют лигатуры – отдельные начертания, обозначающие популярные сочетания символов, например `->`. Слева – шрифт Consolas, справа – шрифт Fira Code

## Typeface vs Font

Typeface: [Helvetica](#)

Font: [Helvetica Light](#)  
[Helvetica Regular](#)  
[Helvetica Oblique](#)  
[Helvetica Bold](#)  
[Helvetica Bold Oblique](#)

Рисунок 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).

Thin  
Extra-Light  
Light  
Regular  
Medium  
**Semi-Bold**  
**Bold**  
**Extra-Bold**  
**Black**

Рисунок 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 разработчи-

*Normal - Sphinx of black quartz, judge my vow.*

*Italic - Sphinx of black quartz, judge my vow.*

*Oblique - Sphinx of black quartz, judge my vow.*

Рисунок 7.12. прямое начертание (roman), наклонное (oblique), кур-  
сив (italic).

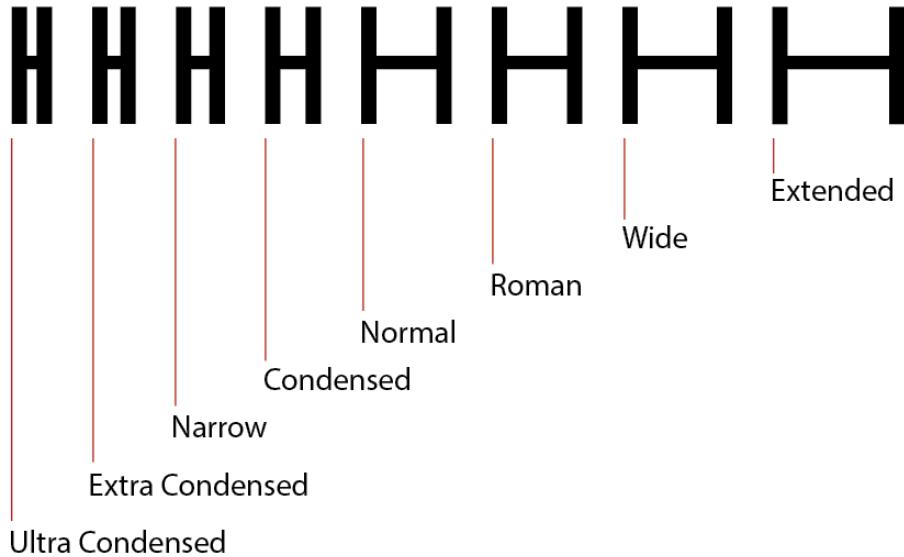


Рисунок 7.13. Различные плотности начертания. Вариации плотно-  
сти встречается чаще всего в интерфейсных и переменных шриф-  
тах.

AV Wa

Текст без кернинга

AV Wa

Текст с кернингом

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

ки ограничиваются малым числом начертаний, исключают курсив. Например, шрифт Roboto, разработанный компанией Google, имеет 12 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 ис-



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

пользоваться 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 веб-страницы или приложения. Нейтральные шрифты могут быть использованы практически везде. Шрифт может быть описан в руководстве по дизайну (Human Interface Guidelines [32, 33, 35, 36, 37, 38]) 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.

# Headline 1

## Headline 2

### Headline 3

#### Headline 4

##### Headline 5

###### Headline 6

Subtitle 1

Subtitle 2

Body 1

Body 2

**BUTTON**

Font	Weight	Size	Letter spacing
Roboto	Medium	14px	1.25px

Caption

OVERLINE

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

**КОФЕ**  
**30 руб.**

К О Ф Е  
300 ₽

Рисунок 7.17. 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.

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»*

Рисунок 7.18. Шрифт Пушкин (внизу) – слишком характерный 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.

### **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<sup>1</sup>:

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

Исследования [70] 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. 7.21).

Иерархию можно показывать используя:

- Размер шрифта;
- Отступы;
- Цвет текста;
- Толщину, начертание (курсив, наклонный, обычный);
- Прописные, строчные буквы, капиталь;
- Трекинг;
- Гарнитуру.

### **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 тон-

---

<sup>1</sup>Элементы иерархии этого перечисления: заголовок и перечисляемые элементы. Отношение показано с помощью отступов и маркеров списка

Президентство Джейфтерсона, находившегося в Белом доме с 4 марта 1801 по 4 марта 1809, было отмечено покупкой в 1803 у Франции за 15 млн. долларов Луизианы, увеличившей почти вдвое территорию США; организацией экспедиции Льюиса и Кларка, которая в 1804-05 вышла на тихоокеанское побережье и вернулась в Сент-Луис, пройдя более 8 тыс. миль; установлением в 1808-09 дипломатических отношений с Россией. Выборы 1804 принесли новый успех Джейфтерсону и его партии. Однако второй срок президентства был омрачен внутренними и внешними неурядицами. Расколом страны и международными осложнениями грозил заговор. Нейтралитет США был поколеблен возобновившимися наполеоновскими войнами. Стремясь избежать вовлечения в европейский конфликт и поддержать суверенитет страны, Джейфтерсон в декабре 1807 подписал Акт об эмбарго, предусматривавший прекращение всей внешней торговли.



## Томас Джейфтерсон

результаты деятельности  
1801-1809

**1803:** Покупка Луизианы за \$15 млн у Франции.

**1804:** Организация экспедиции Льюиса и Кларка

**1808:** Установление дипломатических отношений с Россией



Рисунок 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.

кая 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 длина строки 40–70 знаков. При этом чем длиннее строка, тем больше должен быть интерлиньяж – вертикальное расстояние между соседними строками. При больших объемах текста у пользователя может рябить в глазах. В этом случае помогает снижение контрастности между текстом и фоном. Чаще всего можно сменить черный цвет текста на тёмно-серый.

## 7.4 Текст и синтаксис интерфейса

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



Пример текста, который вряд ли прочитает кто-то, кроме директора и пиарщика

Пример статьи, которая соберет десятки тысяч просмотров, если ее правильно распространять

Рисунок 7.20. 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 «Медуза», 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.

Рисунок 7.21. Фрагмент страницы с сайта meduza.io, где приведена большая статья о новом штамме COVID-19. 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.

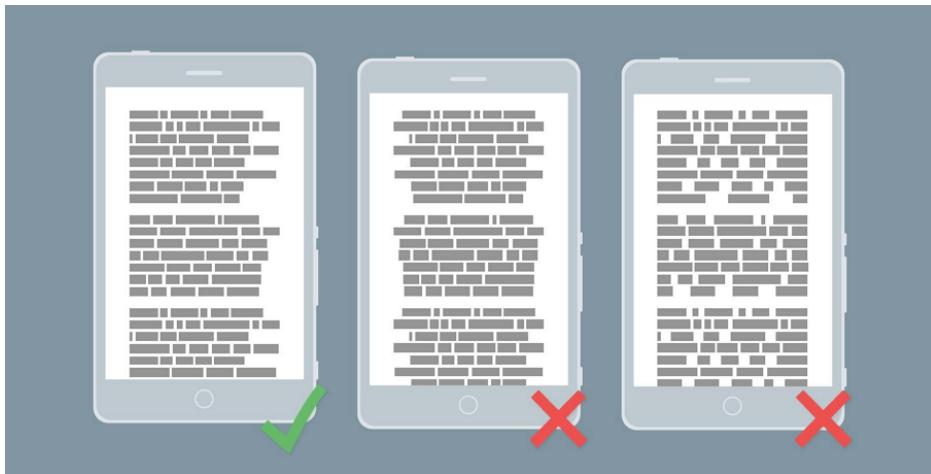


Рисунок 7.22. На сайтах уместнее использовать выравнивание больших блоков текста по левому краю, избегая непропорциональных пробелов.

#### Печатный текст [править | править код]

Традиционные исследования длины строки, ограничивавшиеся печатным текстом, сообщали о разных результатах, но обычно для печатного текста принятая длина строки от 45 до 75 знаков на строку<sup>[en]</sup> (англ. characters per line, cpl), хотя идеальным вариантом является 66 cpl, включая символы и пробелы<sup>[1]</sup>. В книжном наборе длина строки обычно равна 30 размерам набора, но считается допустимым набор в пределах 20–40 раз (то есть,  $30 \times 10\text{-пунктный набор} = 300\text{-пунктная строка}$ )<sup>[1]</sup>. Ранние исследования считали длину строки в 59–97 мм (около 57 cpl) оптимальной для 10-пунктного набора<sup>[2]</sup>. Для печатных работ с несколькими колонками часто лучше 40–50 cpl<sup>[1]</sup>. Для текста на английском языке с выключкой по формату минимальное число знаков на строку — 40, длина строки менее 38–40 символов часто приводит к появлению пробелов (или «коридоров») или слишком большого числа переносов слов в блоке текста<sup>[1]</sup>. Длинные строки (85–90 cpl) могут быть допустимы для несвязанного текста, например, в библиографиях или сносках, но для непрерывного текста строки в более чем 80 символов могут быть слишком длинными. Короткий текст, например, рваные магинации, может быть и длиной в 12–15 знаков на строку<sup>[1]</sup>. Исследования показали, что более краткие строки часто предпочтительны длинным, вероятно, потому что участникам более привычен такой формат<sup>[3]</sup>.

#### Электронный текст [править | править код]

Чтение с экрана создаёт дополнительные проблемы, делая использование традиционных исследований длины текста для электронных форматов проблематичным<sup>[4]</sup>. В отличие от печатного текста, дизайн для цифровых носителей должен учитывать такие факторы, как блики, мерцание и прокрутка или перелистывание<sup>[5]</sup>.

Исследования читаемости цифрового текста показали, что, как и для печатного текста, длина строки может влиять на скорость чтения. Если строки слишком длинные, читателям трудно вернуться к началу следующей строки (*сажада*), в то время как если они слишком короткие, им требуется больше прокрутки или листания<sup>[6]</sup>. Исследователи полагают, что длинные строки лучше подходят для быстрого сканирования, а короткие строки — для точности чтения<sup>[3]</sup>. Более длинные строки, в таком случае, лучше подходят для случаев, когда информация будет быстро считываться читателями, а короткие — когда информация предназначена для более тщательного чтения<sup>[3]</sup>. Одно из предложений для наилучшего компромисса между скоростью чтения и пониманием — использовать около 55 знаков на строку<sup>[6]</sup>. В то же время были и исследования, указавшие, что цифровой текст длиной в 100 знаков может читаться быстрее, чем строки в 25 знаков, сохранив тот же уровень понимания текста<sup>[4]</sup>.

Субъективные факторы также играют роль в выборе длины строки для цифрового текста. Одно исследование показало, что число знаков на строку имело незначительное влияние на читаемость, включая факторы скорости и понимания текста, однако при запросе о своих предпочтениях 60 % участников исследования отдали предпочтение самым коротким (35 cpl) или самым длинным (95 cpl) строкам, использованным в исследовании. В то же время 100 % участников выбрали один из этих вариантов как наименее предпочтительный<sup>[7]</sup>.

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



### В чём важность типографики?

Ян Чихольд и Роберт Бринхёртс не могли прийти к единому мнению в определении типографики, но они были полностью сходятся в том что является её целями. Они лишь использовали разные слова: "Суть новой типографики — ясность" — выделено жирным шрифтом в книге Чихольда. "Новая типография отличается от старой тем фактом, что ее первой задачей является разработка ее видимой формы из функций текста ... форма должна быть создана из функции" — уточняет Чихольд.



Бринхерст был более конкретным. Типография должна приглашать читателя в текст, раскрывать смысл и настроение текста, уточнять его структуру и порядок и связывать текст с другими существующими элементами.

Большинство людей думает, что типография касается шрифтов. Большинство дизайнеров считают, что типография — это шрифты.

Типография больше, чем это, она выражает язык через шрифт. Размещение, состав, выбор шрифта.

Марк Бултон

Типография — это не только шрифты. Речь идет не о размере шрифта, краях, интервалах, масштабах или цвете. Типография посвящена объективной организации информации. Речь идет о информации и о том, как мы передаем эту информацию читателю. Размер, цвета и шрифты — это просто инструменты, которые помогают нам превращать эту информацию в правильную форму. И, как мы увидим, все эти инструменты должны работать вместе в целом. Нет отдельных элементов.

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

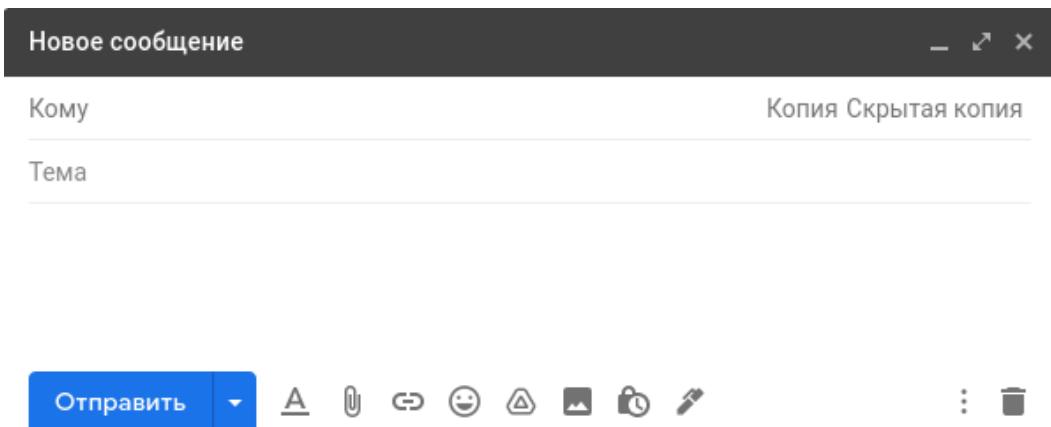


Рисунок 7.25. Форму отправки сообщения в Gmail можно воспринимать как отдельное предложение. Подлежащее – заголовок (существительное), сказуемое – кнопка отправить (глагол).

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

Многие диалоговые окна и формы можно рассматривать как предложения, которые образуют связный текст или предложение (рис. 7.25). 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<sup>2</sup> и сказуемом<sup>3</sup>.

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

<sup>2</sup>Подлежащее – главный член предложения, который обозначает предмет, действие которого выражается сказуемым

<sup>3</sup>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.

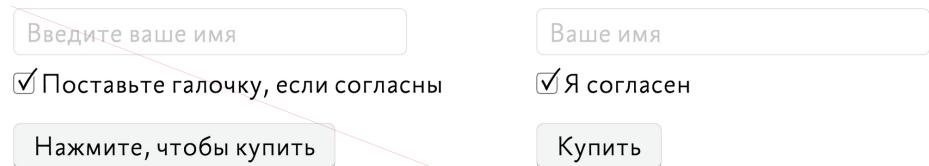


Рисунок 7.26. Синтаксис интерфейса составляют не только слова, но и элементы управления, поля ввода, флажки, переключатели и другие элементы. Поэтому не нужно добавлять к кнопке слово “нажмите”, к полю ввода слово “введите” и т.д.

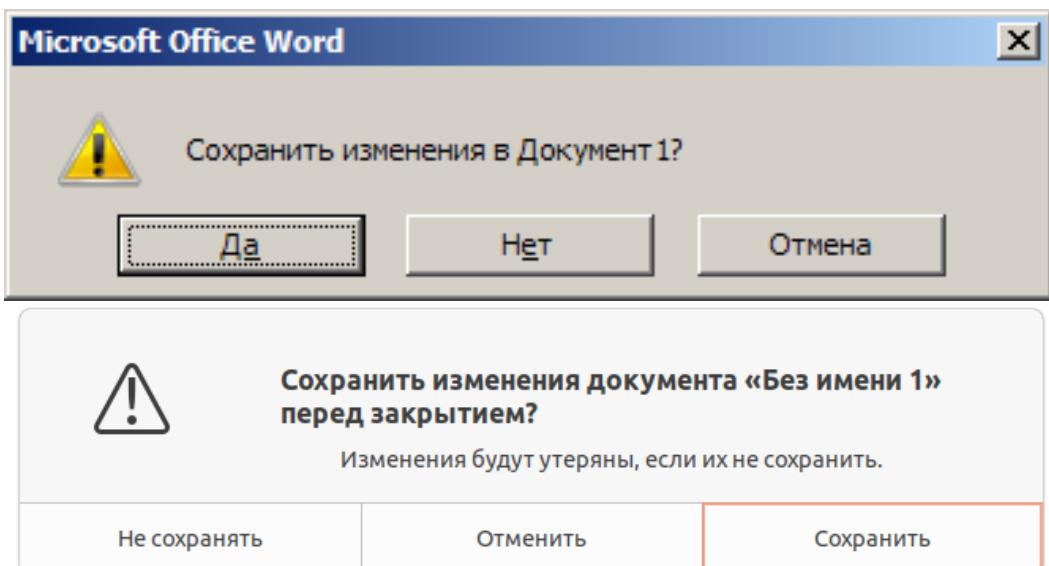


Рисунок 7.27. В старых версиях Microsoft Word из названий кнопок сразу нельзя понять о чем идёт речь. В современных версиях текстовых редакторов все кнопки подписаны глаголами, можно догадаться о чем идёт речь только прочитав названия кнопок.

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 [3] 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.

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

- быть понятным
- быть лаконичным
- быть полезным.

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 сокращения текста демонстрирует Ольминский М. С.<sup>4</sup> на примере новостной заметки о демонстрации в Твери. Текст заканчивался словами: “Явившаяся на место происшествия местная поли-

<sup>4</sup>Ольминский Михаил Степанович – революционер, публицист, историк, литературный критик, литературовед и историк литературы

Original	<b>Failure</b> An authentication error has occurred <b>OK</b>
Clear	<b>Sign-in error</b> You entered an incorrect password <b>OK</b>
Clear, Concise	Wrong password <b>OK</b>
Clear, Concise, Useful	Wrong password <b>TRY AGAIN</b> <b>RECOVER PASSWORD</b>

Рисунок 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 (concise): Неправильный пароль. 3) Сделать полезным (useful), добавить кнопки “попробовать ещё раз“ и “восстановить пароль“.

ция 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“.



404. That's an error.

The requested URL /errorpage was not found on this server. That's all we know.

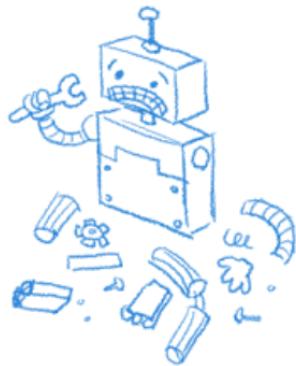


Рисунок 7.29. Гугл ничего не предлагает пользователю полезных действий на странице с ошибкой 404.



### Ошибка 404. Нет такой страницы

Если вы считаете, что страницы нет по нашей вине, [напишите нам](#).

- |                                  |                                    |
|----------------------------------|------------------------------------|
| Эфир — фильмы 1960-х годов       | Маркет — смартфоны Xiaomi          |
| Музыка — для полезной изоляции   | Игры — прямо в браузере            |
| Практикум — учёба с наставником  | Недвижимость — дизайнерский ремонт |
| Расписания — поезда и электрички |                                    |

Рисунок 7.30. Яндекс на странице с ошибкой 404 показывает строку поиска и предлагает обратную связь.

Упростить редактуру могут сервисы:

- orfogrammka.ru – проверка орфографии, пунктуации, стилистики и элементов типографики;
- glvrd.ru – Главред – оценка текста с точки зрения информационного стиля [6];
- 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 автори-

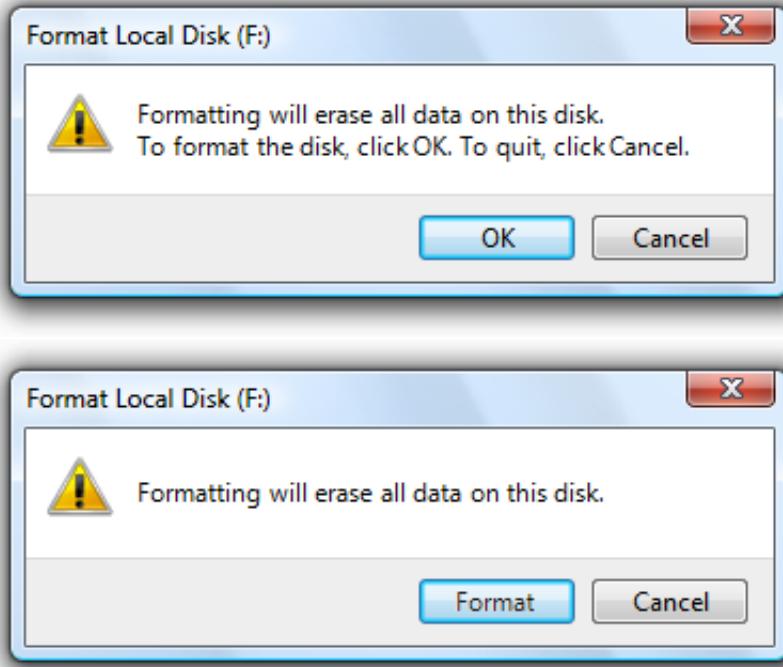


Рисунок 7.31. Глаголы предпочтительнее существительных там, где идёт речь о совершении действия.

зоваться, 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.

## Вопросы

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

ются?

3. Приведите классификации шрифтов.
4. Что такое антиква и готеск? 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 характерными
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. — :Дизайнбюро 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 пособии рассмотрены элементы системы "человек – машина". Дано описание важным с точки взаимодействия с пользовательским интерфейсом (ПИ) характеристикам человека: вниманию, памяти, способности формировать привычки. Описаны особенности восприятия визуальной информации человеком.

Приведена классификация ПИ; парадигмы, на основе которых строится понимание интерфейса пользователем. Рассмотрена методология проектирования ПИ продукта с точки зрения ПИ в частности и пользовательского опыта (UX) вообще.

Рассмотрена основная качественная характеристика ПИ – юзабилити, некоторые способы юзабилити-тестирования, количественной оценки интерфейсов. Метод GOMS.

Наконец приведены общие сведения о типографике и написанию интерфейсных текстов.

Все замечания по содержанию учебного издания можно направлять на почтовый ящик [vetrovsv@outlook.com](mailto:vetrovsv@outlook.com). Автор будет признателен за любые конструктивные предложения.



# Литература

1. Норман Д., Дизайн привычных вещей, обновлённое и дополненное издание — : Манн, Иванов и Фербер, 2021. — 384 с.
2. Канеман Д. Думай медленно... решай быстро — : ACT, 2013. — 710 с.
3. Раскин Д., Интерфейс: новые направления в проектировании компьютерных систем. – Пер. с англ. – СПб: СимволПлюс, 2007. – 272 с.
4. Д. Канеман. Внимание и усилие / под ред. А.Н. Гусева. — Москва: Смысл, 2006. — 287 с.
5. Мильчин А.Э., Методика редактирования теста. Изд. 3-е, перераб. и доп. - М.: Логос, 2005. - 524 с.
6. Ильяхов, М. Пиши, сокращай: Как создавать сильные тексты / М. Ильяхов, Л. Сарычева, . — : Альпина Паблишер, 2021. — 440 с
7. Бирман, И. Пользовательский интерфейс / И. Бирман. — :Дизайн-бюро Артёма Горбунова, 2017. — 419 с.
8. Пользовательский интерфейс: прицеливание. — Текст : электронный // Дизайн-бюро Артёма Горбунова : [сайт]. — URL: <https://bureau.ru/books/ui/demo/4> (дата обращения: 20.08.2021).
9. Гарретт Дж. Веб-дизайн: книга Джессса Гаррета. Элементы опыта взаимодействия». – Пер. с англ. – СПб.: Символ-Плюс, 2008. – 192 с.
10. Алан Купер, Рейманн Роберт М., Основы проектирования взаимодействия. – Пер. с англ. – СПб.: Символ-Плюс, 2018 – 720 с.

11. Alan Cooper, About Face: The Essentials of Interaction Design, Fourth Edition, 2014 – 722 p.
12. Купер, А. Психбольница в руках пациентов. Алан Купер об интерфейсах / А. Купер. — :Питер, 2018. — 384 с.
13. Буч Г., Рамбо Д., Якобсон И. Язык UML. Руководство пользователя. 2-е изд.:Пер. с англ. Мухин Н. – М.:ДМК Пресс, 2006. – 496 с.: ил.
14. Выготский Л.С., Психология. – Москва : Эксмо-Пресс : Апрель Пресс, 2000. – 1008 с.
15. ISO/IEC/IEEE 24765-2010 – Systems and software engineering Vocabulary
16. Медведев, В.Ю. Роль дизайна в формировании культуры / В.Ю. Медведев. – СПб.: СПГУТД, 2004. – 108 с.
17. Горбунов А. с. Типографика и вёрстка. – М.: Изд-во Бюро Горбунова, 2015. - 175 с.
18. Шпикерманн, Э. О шрифте / Э. Шпикерманн. — : Манн, Иванов и Фербер (МИФ), 2016. – 280 с.
19. Google Fonts. — Текст : электронный // : [сайт]. — URL: <https://fonts.googleapis.com/> (дата обращения: 15.09.2021).
20. Variable Fonts. — Текст : электронный // : [сайт]. — URL: <https://variablefonts.com/> (дата обращения: 20.02.2022).
21. Edward R. Tufte, The Visual Display of Quantitative Information, Graphics Press, 2001. 200 p.
22. Edward R. Tufte Envisioning Information, Graphics Press, 1990. 126 p.
23. Edward R. Tufte, Visual Explanations: Images and Quantities, Evidence and Narrative Graphics Press, 1997, 156 p.
24. Edward R. Tufte, Beautiful Evidence, Graphics Press, 2006, 213 p.
25. Стив Круг. Не заставляйте меня думать. – Москва: Издательство “Э“, 2018. – 256 с.

26. Хьюбел, Д. Глаз, мозг, зрение / Д. Хьюбел. — : Мир, 2003. — 240 с. — Текст : непосредственный.
27. Унгер, Р. UX-дизайн. Практическое руководство по проектированию опыта взаимодействия / Р. Унгер, К. Чендлер. — : Символ-Плюс, 2011. — 336 с. — Текст : непосредственный.
28. Голомбински, К. Добавь воздуха! Основы визуального дизайна для графики, веба и мультимедиа / К. Голомбински, Р. Хаген. — : Питер, 2013. — 272 с. — Текст : непосредственный.
29. Орфограммка. — Текст : электронный // :[сайт]. — URL: <https://orfogrammka.ru/> (дата обращения: 23.01.2022).
30. Главред. — Текст : электронный // :[сайт]. — URL: <https://glvrd.ru/> (дата обращения: 23.01.2022).
31. Paradigm. — Текст : электронный // Дизайн-система Mail.ru Paradigm :[сайт]. — URL: <https://paradigm.mail.ru/> (дата обращения: 23.01.2022).
32. Human Interface Guidelines. — Текст : электронный // developer.apple.com : [сайт]. — URL: <https://developer.apple.com/design/human-interface-guidelines/> (дата обращения: 21.07.2021).
33. Material Design Guidelines. — Текст : электронный // material.io : [сайт]. — URL: <https://material.io/design/guidelines-overview> (дата обращения: 21.07.2021).
34. Mew K., Learning Material Design – Packt Publishing Ltd, 2015 – 186 p.
35. Design and code Windows apps. — Текст : электронный // microsoft.com : [сайт]. — URL: <https://docs.microsoft.com/en-us/windows/apps/design/> (дата обращения: 21.02.2022).
36. Fluent Design System. — Текст : электронный // microsoft.com : [сайт]. — URL: <https://www.microsoft.com/design/fluent/> (дата обращения: 21.07.2021).

37. GNOME Human Interface Guidelines. — Текст : электронный // gnome.org : [сайт]. — URL: <https://developer.gnome.org/hig/> (дата обращения: 21.07.2021).
38. KDE Human Interface Guidelines. — Текст : электронный // kde.org : [сайт]. — URL: <https://develop.kde.org/hig/> (дата обращения: 21.07.2021).
39. Дакетт, Д. HTML и CSS. Разработка и дизайн веб-сайтов / Д. Дакетт. — : Эксмо, 2013. — 480 с. — Текст : непосредственный.
40. Jake , Bootstrap: Responsive Web Development – O'Reilly Media, Inc., – 2013 – 128 р.
41. Benjamin Jakobus, Mastering Bootstrap 4: Master the latest version of Bootstrap 4 to build highly customized responsive web apps, 2nd Edition, Packt Publishing Ltd, 2018 – 354 р.
42. Машнин Т., Bootstrap: Быстрое создание современных сайтов – Литрес, 2021. – 210 с.
43. typographic system with modular scale & vertical rhythm. — Текст : электронный // Gridlover : [сайт]. — URL: <https://www.gridlover.net/try> (дата обращения: 11.10.2021).
44. Type Scale - A Visual Type Scale. — Текст : электронный // type-scale.com : [сайт]. — URL: <https://type-scale.com/> (дата обращения: 11.10.2021).
45. Кесенбери, Е. Сторителлинг в проектировании интерфейсов. Как создавать истории, улучшающие дизайн / Е. Кесенбери, К. Брукс, . — :Манн, Иванов и Фербер (МИФ), 2013. — 310 с
46. Зараменских, Е. П. Управление жизненным циклом информационных систем : учебник и практикум для вузов / Е. П. Зараменских. — 2-е изд. — Москва : Издательство Юрайт, 2022. — 497 с. — (Высшее образование). — ISBN 978-5-534-14023-1. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/489983> (дата обращения: 23.01.2022).

47. Одегов, Ю. Г. Эргономика : учебник и практикум для академического бакалавриата / Ю. Г. Одегов, М. Н. Кулапов, В. Н. Сидорова. — Москва : Издательство Юрайт, 2018. — 157 с. — (Бакалавр. Академический курс). — ISBN 978-5-9916-8258-9. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/413951> (дата обращения: 23.01.2022).
48. Климов Е.А. Инженерная психология и эргономика : учебник для академического бакалавриата / Е. А. Климов [и др.] ; под редакцией Е. А. Климова, О. Г. Носковой, Г. Н. Солнцевой. — Москва : Издательство Юрайт, 2018. — 178 с. — (Бакалавр. Академический курс. Модуль). — ISBN 978-5-534-00906-4. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/415260> (дата обращения: 23.02.2022).
49. Головач, В. В. Дизайн пользовательского интерфейса. Искусство мыть слона / В. В. Головач. — 2009 : , 2009. — 97 с.
50. Магазанник В. Д. Человеко-компьютерное взаимодействие : учеб. пособие для вузов / Магазанник В. Д. - 2-е изд., доп. и перераб. - М. : Университетская книга, 2016. - 406 с.
51. Мандел, Т. Разработка пользовательского интерфейса / Т. Мандел. — Москва : ДМК Пресс, 2007. — 418 с. — ISBN 5-94074-069-3. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/1227>
52. Сакулин, с. А. Основы интернет-технологий: HTML, CSS, JavaScript, XML : учебное пособие / с. А. Сакулин. — Москва : МГТУ им. Н.Э. Баумана, 2017. — 112 с. — ISBN 978-5-7038-4724-4. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/103525>
53. Винокуров, И. В. Использование библиотек классов Swing и MFC для разработки графического интерфейса пользователя : учебное пособие / И. В. Винокуров. — 2-е изд., стер. — Москва : МГТУ им. Н.Э. Баумана, 2011. — 396 с. — ISBN 978-5-7038-3560-9. —

Текст : электронный // Лань : электронно-библиотечная система.  
— URL: <https://e.lanbook.com/book/106519>

54. Алфимцев А. Н. Разработка пользовательского интерфейса [Электрон. ресурс] : метод. указания к лаб. раб. по дисциплине "Протоколы и интерфейсы информационных систем" / Алфимцев А. Н. ; МГТУ им. Н. Э. Баумана. - М. : МГТУ им. Н. Э. Баумана, 2010.
55. Алфимцев А.Н. 25 упражнений по юзабилити. Saarbrücken, Deutschland, LAP Lambert Academic Publishing. 2014. 108 с.
56. ГОСТ Р МЭК 60447-2000 Интерфейс человеко-машинный. Принципы приведения в действие.
57. ГОСТ Р МЭК 60073-2000 Интерфейс человекомашинный. Маркировка и обозначение органов управления и контрольных устройств. Правила кодирования информации.
58. Руководство по Figma. — Текст : электронный // /designer : [сайт]. — URL: <https://slashdesigner.ru/figma-guide> (дата обращения: 26.01.2022).
59. Squire, L. R., Knowlton, B. J. The medial temporal lobe, the hippocampus, and the memory systems of the brain. In M. Gazaniga (Ed.), The new cognitive neurosciences (2nd ed., pp. 765–780). Cambridge, MA: MIT Press., 2000
60. George A. Miller. The Magical Number Seven, Plus or Minus Two. // The Psychological Review, 1956, vol. 63, pp. 81–97.
61. The precipitous rise of Figma and fall of InVision. — Текст : электронный // : [сайт]. — URL: <https://uxdesign.cc/the-precipitous-rise-of-figma-and-fall-of-invision-435f07e8d1b6> (дата обращения: 27.07.2021).
62. The magical number seven, plus or minus two: Not relevant for design [Электронный ресурс]. – Режим доступа: [https://www.edwardtufte.com/bboard/q-and-a-fetch-msg?msg\\_id=0000U6](https://www.edwardtufte.com/bboard/q-and-a-fetch-msg?msg_id=0000U6) (дата обращения: 7.11.2021)

63. Schneider W., Shiffrin R.M. Controlled and automatic human information processing: Detection, search and attention // Psychol. Review. 1977. Vol. 84. N 1. p. 1–66.
64. Don, Norman; Jakob, Nielsen. "The Definition of User Experience (UX)". Nielsen Norman Group. Retrieved 2 March 2021.
65. Хабр: Принципы гештальта в дизайне пользовательского интерфейса [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/company/cloud4y/blog/347444/>. – Дата доступа: 13.09.2021.
66. Visualizing Fittss law. — Текст : электронный // Particletree : [сайт]. — URL: <http://particletree.com/features/visualizing-fittss-law/> (дата обращения: 20.08.2021).
67. MacKenzie, I. S.. Fitts' law. In K. L. Norman & J. Kirakowski (Eds.), Handbook of human-computer interaction, 2018, pp. 349-370.
68. Carroll, John Millar and Mary Beth Rosson. Paradox of the active user. 1987.
69. 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).
70. Applying Writing Guidelines to Web Pages. — Текст : электронный // Nielsen Norman Group : [сайт]. — URL: <https://www.nngroup.com/articles/applying-writing-guidelines-web-pages/> (дата обращения: 21.02.2022).
71. Csikszentmihalyi, Mihaly. Flow: The Psychology of Optimal Experience, 1990, 336 p.
72. Т. Демарко, Т. Листер «Человеческий фактор: успешные проекты и команды». – Пер. с англ. – СПб: СимволПлюс, 2005. 256 с.

# Предметный указатель

- A/B testing, 120
- А/В тестирование, 120
- GOMS, 135
- HSV, 58
- mockup, 90
- RGB, 54
- screen flow, 79
- task flow, 79, 80
- use case diagram, 74
- user experience, 65
- user flow, 79
- user interface, 31
  - wire flow, 79
  - wireframe, 83
- автоматичные задачи, 16
- архитектура информационная,
  - 76
- вайрфрейм, 83
- внимание, 11
- восприятие, 49
- гарнитура шрифтовая, 153
- гештальтпсихология, 49
- диаграмма
  - потока, 79
- потоков задач, 79, 80
- экранов, 79
- диаграмма прецедентов, 74
- дизайн, 45
- дизайн-система, 45
- закон
  - Фиттса, 125
  - Хика, 133
  - близости, 49
- интерфейс, 31
  - жестовый, 32
  - идеоматический, 39
  - командной строки, 36
  - материальный, 32
  - метафорический, 39
  - нейрокомпьютерный, 32
  - текстовый, 35
- интерфейс пользовательский, 31
- интерференция внимания, 16
- информационность интерфейса,
  - 140
- информационная производительность интерфейса, 141
- квазирежим, 42
- кернинг, 156
- локус внимания, 14
- макет аннотированный, 90

метод  
карточной сортировки, 122  
персонажей, 71

модальность, 41

модель  
ментальная, 22, 119  
пользователя, 74  
представления, 23  
реализации, 23, 77

мокап, 90

навигация, 77

опыт взаимодействия, 65

ошибки и промахи, 116

ощущение, 49

память кратковременная, 20

парадигмы интерфейса, 38

парадокс активного пользователя, 27

персонаж, 71, 122

пользовательская роль, 74

правило  
бесконечной границы, 127  
размера цели, 127

представление, 49

прерывание опыта, 13

принцип гештальта  
близость, 49  
замкнутость, 50  
общая зона, 50  
смежность, 50  
схожесть, 50  
целостность, 50

проклятие знания, 26

прототип пользовательского ин-

терфейса, 90, 100

режим, 41

сетка, 85

символьная эффективность интерфейса, 145

сканирующее чтение, 160, 162

скевоморфизм, 39

состояние потока, 15

тестирование  
коридорное, 122

типографика, 148

уровни опыта взаимодействия, 67

шрифт, 153  
аварийный, 150  
без засечек, 148  
брюсовый, 150  
моноширинный, 151  
переменный, 156  
пропорциональный, 151  
с засечками, 148

шрифтовая иерархия, 157

эвристики Нильсона, 111

эффект выскакивания, 13, 62

юзабилити, 111

юзабилити-тестирование, 119

*Учебное издание*

Человеко-машинное взаимодействие

Учебное пособие набрано и отвёрстано в системе TeX

Печатается с оригинал-макета автора при участии издательства

Подписано в печать \*\*.\*\*.2022 . Формат \*\* × \*\* '/ \*\*. Бумага  
ксерографическая.

Гарнитура PT Serif. Способ печати цифровой.

Усл. печ. л. \*\*. Уч.-изд. л. \*\*. Заказ № \*\*\*\*\*.

Тираж \*\*\* экз. (1-й з-д 1–28 экз.).????

ФГБОУ ВО «Забайкальский государственный университет» 672039,  
г. Чита, ул. Александро-Заводская, 30