**Практическая работа № 1.**

**Человеко-компьютерное взаимодействие. Человеко-компьютерный интерфейс**

**Цель работы**: изучение основ интерфейсов взаимодействия человека с компьютером.

**1. Теоретические сведения.**

Много книг написано о том, как устроен компьютер: о его "мозгах" и "сердце" (памяти и процессоре), о его "нервных волокнах" (шине), о его "органах чувств" (остальных устройств). Однако не следует забывать, что компьютер – это машина, а не биологическое существо. Как любое техническое устройство, компьютер обменивается информацией с человеком посредством набора определенных правил, обязательных как для машины, так и для человека. Эти правила в компьютерной литературе называются интерфейсом. Интерфейс может быть понятным и непонятным, дружественным и нет. К нему подходят многие прилагательные. Но в одном он постоянен: он есть, и никуда от него не денешься.

Интерфейс, по определению – это правила взаимодействия операционной системы с пользователями, а также соседних уровней в сети ЭВМ. От интерфейса зависит технология общения человека с компьютером.

**Классификация интерфейсов**

Как уже указывалось выше, интерфейс – это, прежде всего, набор правил. Как любые правила, их можно обобщить, собрать в "кодекс", сгруппировать по общему признаку. Таким образом, мы пришли к понятию "вид интерфейса" как объединение по схожести способов взаимодействия человека и компьютеров. Вкратце можно предложить следующую схематическую классификацию различных интерфейсов общения человека и компьютера.

Современными видами интерфейсов являются:

1. *Командный интерфейс.* Командный интерфейс называется так по тому, что в этом виде интерфейса человек подает "команды" компьютеру, а компьютер их выполняет и выдает результат человеку. Командный интерфейс реализован в виде пакетной технологии и технологии командной строки.

2. *WIMP - интерфейс* (Window - окно, Image - образ, Menu - меню, Pointer - указатель). Характерной особенностью этого вида интерфейса является то, что диалог с пользователем ведется не с помощью команд, а с помощью графических образов - меню, окон, других элементов. Хотя и в этом интерфейсе подаются команды машине, но это делается "опосредственно", через графические образы. Этот вид интерфейса реализован на двух уровнях технологий: простой графический интерфейс и "чистый" WIMP - интерфейс.

3. *SILK - интерфейс* (Speech - речь, Image - образ, Language - язык, Knowlege - знание). Этот вид интерфейса наиболее приближен к обычной, человеческой форме общения. В рамках этого интерфейса идет обычный "разговор" человека и компьютера. При этом компьютер находит для себя команды, анализируя человеческую речь и находя в ней ключевые фразы. Результат выполнения команд он также преобразует в понятную человеку форму. Этот вид интерфейса наиболее требователен к аппаратным ресурсам компьютера, и поэтому его применяют в основном для военных целей.

4.*"Мимический интерфейс"* (Биометрическая технология).

5.*Общественный интерфейс* - основан на семантических сетях.

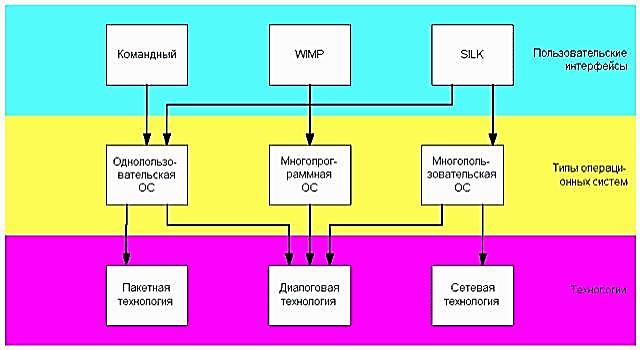


Рис. 1 Взаимодействие типов операционных систем, пользовательских интерфейсов и технологий их реализации

**1. Командный интерфейс**

**1.1. Пакетная технология**

Исторически этот вид технологии появился первым. Она существовала уже на релейных машинах Зюса и Цюзе (Германия, 1937 год). Идея ее проста: на вход компьютера подается последовательность символов, в которых по определенным правилам указывается последовательность запущенных на выполнение программ. После выполнения очередной программы запускается следующая и т.д. Машина по определенным правилам находит для себя команды и данные. В качестве этой последовательности может выступать, например, перфолента, стопка перфокарт, последовательность нажатия клавиш электрической пишущей машинки (типа CONSUL). Машина также выдает свои сообщения на перфоратор, алфавитно-цифровое печатающее устройство (АЦПУ), ленту пишущей машинки.

Такая машина представляет собой "черный ящик" (точнее "белый шкаф"), в который постоянно подается информация и которая также постоянно "информирует" мир о своем состоянии (см. рисунок 2.) Человек здесь имеет малое влияние на работу машины - он может лишь приостановить работу машины, сменить программу и вновь запустить ЭВМ. Впоследствии, когда машины стали помощнее и могли обслуживать сразу нескольких пользователей, вечное ожидание пользователей типа: "Я послал данные машине. Жду, что она ответит. И ответит ли вообще?" - стало, мягко говоря, надоедать. К тому же вычислительные центры, вслед за газетами, стали вторым крупным "производителем" макулатуры. Поэтому с появлением алфавитно-цифровых дисплеев началась эра по-настоящему пользовательской технологии - командной строки.

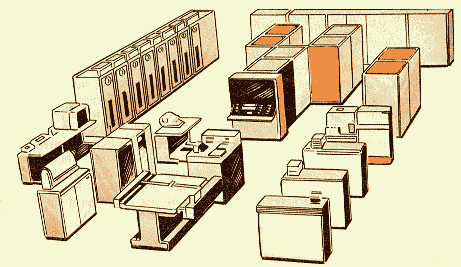
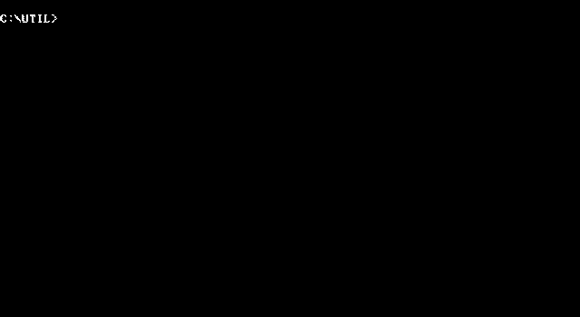


Рис. 2. Вид большой ЭВМ серии ЕС ЭВМ.

**1.2. Технология командной строки**

При этой технологии в качестве единственного способа ввода информации от человека к компьютеру служит клавиатура, а компьютер выводит информацию человеку с помощью алфавитно-цифрового дисплея (монитора). Эту комбинацию (монитор + клавиатура) стали называть терминалом, или консолью.

Команды набираются в командной строке. Командная строка представляет собой символ приглашения и мигающий прямоугольник - курсор (см. рисунок 3.) При нажатии клавиши на месте курсора появляются символы, а сам курсор смещается вправо. Это очень похоже на набор команды на пишущей машинке. Однако, в отличие от нее, буквы отображаются на дисплее, а не на бумаге, и неправильно набранный символ можно стереть. Команда заканчивается нажатием клавиши Enter (или Return.) После этого осуществляется переход в начало следующей строки. Именно с этой позиции компьютер выдает на монитор результаты своей работы. Затем процесс повторяется.

 а)

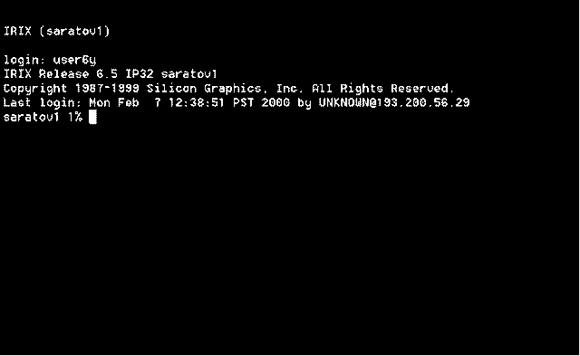
 б)

Рис.3. Приглашения командной строки в различных операционных системах:

а) MS-DOS, б) IRIX.

Технология командной строки уже работала на монохромных алфавитно-цифровых дисплеях. Поскольку вводить позволялось только буквы, цифры и знаки препинания, то технические характеристики дисплея были не существенны. В качестве монитора можно было использовать телевизионный приемник и даже трубку осциллографа.

Обе эти технологии реализуются в виде командного интерфейса - машине подаются на вход команды, а она как бы "отвечает" на них.

Преобладающим видом файлов при работе с командным интерфейсом стали текстовые файлы - их и только их можно было создать при помощи клавиатуры. На время наиболее широкого использования интерфейса командной строки приходится появление операционной системы UNIX и появление первых восьмиразрядных персональных компьютеров с многоплатформенной операционной системой CP/M.

**2. Графический интерфейс**

Как и когда появился графический интерфейс?

Его идея зародилась в середине 70-х годов, когда в исследовательском центре Xerox Palo Alto Research Center (PARC) была разработана концепция визуального интерфейса. Предпосылкой графического интерфейса явилось уменьшение времени реакции компьютера на команду, увеличение объема оперативной памяти, а также развитие технической базы компьютеров. Аппаратным основанием концепции, конечно же, явилось появление алфавитно-цифровых дисплеев на компьютерах, причем на этих дисплеях уже имелись такие эффекты, как "мерцание" символов, инверсия цвета (смена начертания белых символов на черном фоне обратным, то есть черных символов на белом фоне), подчеркивание символов. Эти эффекты распространились не на весь экран, а только на один или более символов. Следующим шагом явилось создание цветного дисплея, позволяющего выводить, вместе с этими эффектами, символы в 16 цветах на фоне с палитрой (то есть цветовым набором) из 8 цветов. После появления графических дисплеев, с возможностью вывода любых графических изображений в виде множества точек на экране различного цвета, фантазии в использовании экрана вообще не стало границ! Первая система с графическим интерфейсом 8010 Star Information System группы PARC, таким образом, появилась за четыре месяца до выхода в свет первого компьютера фирмы IBM в 1981 году. Первоначально визуальный интерфейс использовался только в программах. Постепенно он стал переходить и на операционные системы, используемых сначала на компьютерах Atari и Apple Macintosh, а затем и на IBM -- совместимых компьютерах.

С более раннего времени, и под влиянием также и этих концепций, проходил процесс по унификации в использовании клавиатуры и мыши прикладными программами. Слияние этих двух тенденций и привело к созданию того пользовательского интерфейса, с помощью которого, при минимальных затратах времени и средств на переучивание персонала, можно работать с любыми программным продуктом. Описание этого интерфейса, общего для всех приложений и операционных систем, и посвящена данная часть.

Графический интерфейс пользователя за время своего развития прошел две стадии. Об эволюции графического интерфейса с 1974 по настоящее время будет рассказано ниже.

**2.1. Простой графический интерфейс**

На первом этапе графический интерфейс очень походил на технологию командной строки. Отличия от технологии командной строки заключались в следующем.

a) При отображении символов допускалось выделение части символов цветом, инверсным изображением, подчеркиванием и мерцанием. Благодаря этому повысилась выразительность изображения.

b) В зависимости от конкретной реализации графического интерфейса курсор может представляться не только мерцающим прямоугольником, но и некоторой областью, охватывающей несколько символов и даже часть экрана. Эта выделенная область отличается от других, невыделенных частей (обычно цветом).

c) Нажатие клавиши Enter не всегда приводит к выполнению команды и переходу к следующей строке. Реакция на нажатие любой клавиши во многом зависит от того, в какой части экрана находился курсор.

d) Кроме клавиши Enter, на клавиатуре все чаще стали использоваться "серые" клавиши управления курсором.

e) Уже в этой редакции графического интерфейса стали использоваться манипуляторы (типа мыши, трекбола и т.п. - см. рисунок 4.) Они позволяли быстро выделять нужную часть экрана и перемещать курсор.

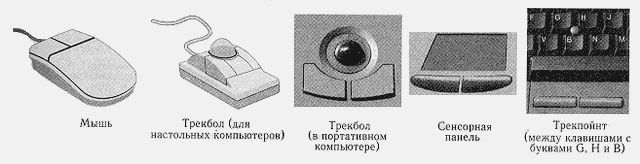


Рис. 4.Манипуляторы

Подводя итоги, можно привести следующие отличительные особенности этого интерфейса.

1) Выделение областей экрана.

2) Переопределение клавиш клавиатуры в зависимости от контекста.

3) Использование манипуляторов и серых клавиш клавиатуры для управления курсором.

4) Широкое использование цветных мониторов.

Появление этого типа интерфейса совпадает с широким распространением операционной системы MS-DOS. Именно она внедрила этот интерфейс в массы, благодаря чему 80-е годы прошли под знаком совершенствования этого типа интерфейса, улучшения характеристик отображения символов и других параметров монитора.

Типичным примером использования этого вида интерфейса является файловая оболочка Nortron Commander (о файловых оболочках смотри ниже) и текстовый редактор Multi-Edit. А текстовые редакторы Лексикон, ChiWriter и текстовый процессор Microsoft Word for Dos являются примером, как этот интерфейс превзошел сам себя.

**2.2. WIMP - интерфейс**

Вторым этапом в развитии графического интерфейса стал "чистый" интерфейс WIMP, Этот подвид интерфейса характеризуется следующими особенностями.

1. Вся работа с программами, файлами и документами происходит в окнах - определенных очерченных рамкой частях экрана.

2. Все программы, файлы, документы, устройства и другие объекты представляются в виде значков - иконок. При открытии иконки превращаются в окна.

3. Все действия с объектами осуществляются с помощью меню. Хотя меню появилось на первом этапе становления графического интерфейса, оно не имело в нем главенствующего значения, а служило лишь дополнением к командной строке. В чистом WIMP - интерфейсе меню становится основным элементом управления.

4. Широкое использование манипуляторов для указания на объекты. Манипулятор перестает быть просто игрушкой - дополнением к клавиатуре, а становится основным элементом управления. С помощью манипулятора УКАЗЫВАЮТ на любую область экрана, окна или иконки, ВЫДЕЛЯЮТ ее, а уже потом через меню или с использованием других технологий осуществляют управление ими.

Следует отметить, что WIMP требует для своей реализации цветной растровый дисплей с высоким разрешением и манипулятор. Также программы, ориентированные на этот вид интерфейса, предъявляют повышенные требования к производительности компьютера, объему его памяти, пропускной способности шины и т.п. Однако этот вид интерфейса наиболее прост в усвоении и интуитивно понятен. Поэтому сейчас WIMP - интерфейс стал стандартом де-факто.

Ярким примером программ с графическим интерфейсом является операционная система Microsoft Windows.

**3. Речевая технология**

С середины 90-х годов, после появления недорогих звуковых карт и широкого распространения технологий распознавания речи, появился так называемый "речевая технология" SILK - интерфейса. При этой технологии команды подаются голосом путем произнесения специальных зарезервированных слов - команд. Основными такими командами (по правилам системы "Горыныч") являются:

- "Проснись" - включение голосового интерфейса.

- "Отдыхай" - выключение речевого интерфейса.

- "Открыть" - переход в режим вызова той или иной программы. Имя программы называется в следующем слове.

- "Буду диктовать" - переход из режима команд в режим набора текста голосом.

- "Режим команд" - возврат в режим подачи команд голосом.

- и некоторые другие.

Слова должны выговариваться четко, в одном темпе. Между словами обязательна пауза. Из-за неразвитости алгоритма распознавания речи такие системы требует индивидуальной предварительной настройки на каждого конкретного пользователя.

"Речевая" технология является простейшей реализацией SILK - интерфейса.

**4. Биометрическая технология ("Мимический интерфейс")**

Эта технология возникла в конце 90-х годов XX века и еще разрабатывается. Для управления компьютером используется выражение лица человека, направление его взгляда, размер зрачка и другие признаки. Для идентификации пользователя используется рисунок радужной оболочки его глаз, отпечатки пальцев и другая уникальная информация. Изображения считываются с цифровой видеокамеры, а затем с помощью специальных программ распознавания образов из этого изображения выделяются команды. Эта технология, по-видимому, займет свое место в программных продуктах и приложениях, где важно точно идентифицировать пользователя компьютера.

**5. Семантический (Общественный) интерфейс**

Этот вид интерфейса возник в конце 70-х годов XX века, с развитием искусственного интеллекта. Его трудно назвать самостоятельным видом интерфейса - он включает в себя и интерфейс командной строки, и графический, и речевой, и мимический интерфейс. Основная его отличительная черта - это отсутствие команд при общении с компьютером. Запрос формируется на естественном языке, в виде связанного текста и образов. По своей сути это трудно называть интерфейсом - это уже моделирование "общения" человека с компьютером.

С середины 90-х годов XX века, похоже, что в связи с важным военным значением этих разработок (например, для автономного ведения современного боя машинами - роботами, для "семантической" криптографии) эти направления были засекречены. Информация, что эти исследования продолжаются, иногда появляется в периодической печати (обычно в разделах компьютерных новостей).

**2. Порядок выполнения работы**

1. Изучение теоретического материала

2. Ответы на контрольные вопросы

3. Оформление отчета

Согласно своему варианту выполнить задание практической работы:

**Задание 1.** Определить тематику для разработки пользовательского интерфейса.

|  |  |
| --- | --- |
| **№ варианта** | **Тема** |
|  | Заказ и доставка еды |
|  | Туристическое агентство |
|  | Аэропорт (Ж/д станция). Пассажирское расписание и перевозки |
|  | Автосалон |
|  | Ветеринарная клиника (больница) |
|  | Доставка продуктов питания |
|  | Салон красоты (услуги, обслуживающий персонал и пр.) |
|  | Рекламное агентство |
|  | Санаторий |
|  | Риэлтерская компания: аренда; продажа первичного и вторичного жилья |
|  | Складская логистика |
|  | Магазин спорт товаров |
|  | Производство мебели (прием индивидуальных и типовых заказов и изготовление) |
|  | Такси |
|  | Транспортная логистика |
|  | Агентство по сдаче автомобилей в аренду |
|  | Страховая компания |
|  | Учебный центр |
|  | Разработка приложения на регистрацию абитуриентов в DTM |
|  | Игровой фан-клуб |
|  | Гостиница |

**Задание 2.** Подготовить презентацию (см. вариант)

|  |  |
| --- | --- |
| **№ варианта** | **Тема для Задания 2** |
| *Вариант 1.* | Познакомиться с основными понятиями. Привести примеры пользовательских интерфейсов. Распечатать отчет и защитить теоретическую часть. |
| *Вариант 2.* | Перечислить виды интерфейсов и взаимодействий. Распечатать отчет и защитить теоретическую часть. |
| *Вариант 3.* | Определить основные поколения интерфейсов. Распечатать отчет и защитить теоретическую часть. |
| *Вариант 4.* | Показать работу с командным интерфейсом, привести примеры. Распечатать отчет и защитить теоретическую часть. |
| *Вариант 5.* | Показать работу с WIMP интерфейсом, привести примеры. Распечатать отчет и защитить теоретическую часть. |
| *Вариант 6.* | Показать работу с SILK интерфейсом, привести примеры. Распечатать отчет и защитить теоретическую часть. |
| *Вариант 7.* | Показать работу с "Мимическим интерфейсом", привести примеры. Распечатать отчет и защитить теоретическую часть. |
| *Вариант 8.* | Показать работу с семантическим (общественным) интерфейсом, привести примеры. Распечатать отчет и защитить теоретическую часть. |

**3. Содержание отчета**

1. Титульный лист

2. Тема и цель работы

3. Выполнение задания, согласно своего варианта

4. Ответы на контрольные вопросы

5. Выводы

**Контрольные вопросы:**

1. Классификация интерфейсов.

2. Описать технологии командного интерфейса.

3. Каковы этапы развития графического интерфейса? Опишите их.

4. Какова суть SILK - интерфейса.

5. В каких программных продуктах и приложениях наиболее целесообразно использование биометрической технологии, почему?

6. Что представляет собой семантический интерфейс?

7. Приведите примеры программ, относящихся к различным видам интерфейса.

**Практическая работа № 2.**

**Дизайн дисплея. Требования к построению прототипа. Этапы построения прототипа: создание версий.**

**Цель работы:** освоить создание динамических прототипов в среде Flash, опробовать процесс тестирования прототипов и их итерационного улучшения.

**1.Теоретические сведения.**

**Прототипирование**

Прототипирование (создание прототипов) и итеративное улучшение проектов интерфейса признаются одними из наиболее мощных методов в проектировании взаимодействия человека и компьютера. Использование прототипов несёт, в частности, следующие преимущества:

* Стимулирование коммуникации между дизайнером и пользователями (заказчиком).
* Возможность уточнить проектные требования и цели проектирования, ещё не затратив много сил и времени на создание реального продукта.
* Возможность проверить идеи, заложенные в решении, на ранних стадиях разработки, избежать дорогостоящих ошибок.

В прототипировании иногда выделяют 4 стадии:

* Концептуальный дизайн: для исследования различных метафор в интерфейсе и подходов к дизайну.
* Дизайн взаимодействия: для организации структуры экранов или страниц и переходов между ними.
* Дизайн экранов: разработка внешнего вида каждого из конкретных экранов или страниц.
* Тестирование: оценка прототипов для их последующего итерационного улучшения (при помощи небольшого тестирования с представителями реальных пользователей или с использованием «эвристического анализа» экспертами).

**Виды прототипов и технологии их создания**

**Статичные прототипы:** включают в себя такую популярную методику как бумажное прототипирование, создание «скелетов» (набросков, эскизов интерфейсов – см. Рис. 1), создание прототипов в виде статичных изображений (при помощи графических редакторов).

**Динамические прототипы** могут создаваться в специализированных инструментах и средах, в редакторах анимации (прежде всего, Flash), на базе веб-редакторов (редакторов HTML-кода) или систем управления содержимым сайтов (CMS).

Пример разработки прототипа интерфейса с использованием технологий Flash **приводится в Приложении** к работе.

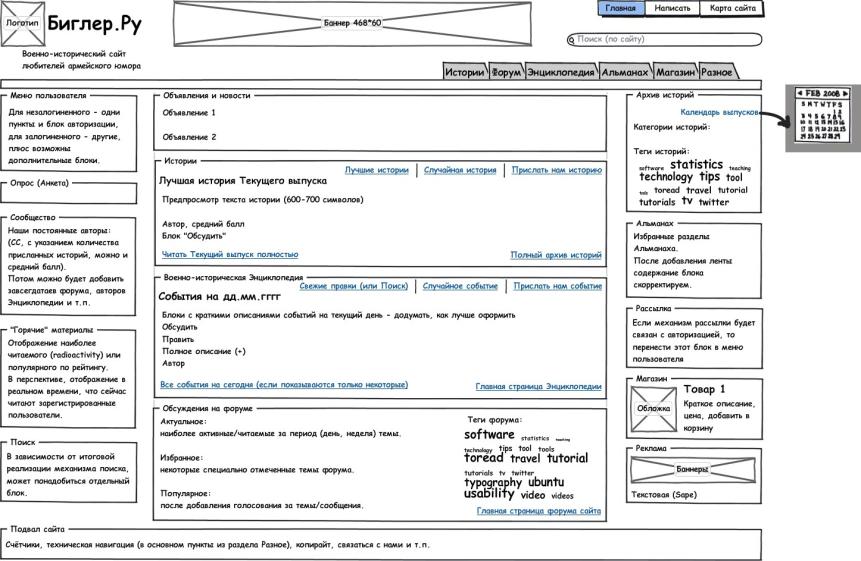


Рис. 1. Пример статичного прототипа («скелета»), созданного в программе Balsamiq Mockups.

***Версии прототипов.*** Первая версия – *бумажная,* вторая – *презентационная,* третья – *псевдореальная,* четвертая – *реальная.*

***Бумажная версия прототипа.*** Достоинствами ее являются исключительная простота и скорость рисования. Польза начального прототипирования на бумаге заключается, во‑первых, в исключительной простоте модификации по результатам тестирования, а во‑вторых, в относительной простоте привлечения представителей целевой аудитории.

Бумажная версия прототипа рисуется на этапах постановки задачи и высокоуровневого проектирования. Бумажный прототип рисуется быстро, но медленно перерисовывается. Бумажный прототип многое искажает. В компьютерном прототипе таких проблем нет – его можно нарисовать совпадающим по размерам с реальным. Таким образом, бумажные прототипы не могут полностью заменить прототипы экранные.

Примерно в таком виде может выглядеть бумажная версия интерфейса (рис. 2).

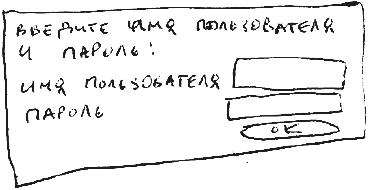


Рис. 2. Вариант бумажной версии интерфейса

***Презентационная версия прототипа.*** После того как исчерпаны возможности бумажной версии прототипа, стоит создать новую версию. Для этого точно так же рисуется интерфейс, но уже не на бумаге, а в какой‑либо презентационной программе. С этой версией прототипа можно тестировать значительно более сложное взаимодействие человека с системой, нежели с бумажной. В то же время исправлять ошибки значительно труднее. Такая версия прототипа может создаваться уже на этапе высокоуровневого проектирования, но на этапе низкоуровневого проектирования ПИ она обязательна.

Достаточно распространенными инструментами для создания прототипов этого вида являются на сегодняшний день MS PowerPoint и MS Visio.

Конечно, такие пакеты, как MS FrontPage или Macromedia Dreamweaver, обладают широкими возможностями для создания интерактивных прототипов, они способны, в частности, оперировать не только линейными, но и иерархическими структурами. Пакеты Macromedia Flash и Director имеют возможность для продолжительной и достаточно сложной анимации с большим количеством объектов. Но все такие пакеты намного дороже PowerPoint и сложнее в освоении.

MS Visio превосходит по возможностям привычный MS PowerPoint, но уступает Macromedia FreeHand. А вообще, подойдет любой иллюстративный пакет, обладающий способностью работать с несколькими страницами. Возможности MS PowerPoint для такой работы слишком малы, а возможности FreeHand, напротив, слишком велики.

Некоторые достаточно продвинутые разработчики не могут избежать соблазна воспользоваться для создания прототипа средствами быстрой разработки приложений (Delphi, Visual Basic и т.д.). В этом случае прототип интерфейса воспринимается лишь как вспомогательная оболочка над программным кодом. Как следствие возникают следующие недостатки.

1. Для тестирования прототипа на пользователях крайне желательно, чтобы он хоть немного работал, т.е. нажатие на кнопку вызывало другое окно или из выпадающего списка можно было выбрать значение. Но в данном случае любое взаимодействие как между отдельными элементами интерфейса, так и между различными формами реализуется только с помощью написания программного кода.

2. Чрезвычайно сложно создать принципиально новый элемент интерфейса либо модифицировать уже имеющийся. Подобные задачи часто встречаются на практике, так как для хорошего интерфейса стандартных элементов, как правило, не хватает.

3. Каждый элемент управления здесь имеет несколько десятков различных свойств, тогда как для прототипирования требуются лишь настройки внешнего вида – шрифт, цвет, текст, размер.

4. Естественный недостаток: нельзя разрабатывать web‑интерфейс. Фактически для большинства случаев этой презентационной версии прототипа оказывается вполне достаточно.

***Псевдореальная версия прототипа.*** В тех случаях, когда в интерфейсе появляются нестандартные элементы или необходимо проверить реальную скорость взаимодействия пользователя с системой, создается еще одна версия прототипа – реально выглядящая, но лишенная каких‑либо алгоритмов и, соответственно, не показывающая реальных данных. Делать этот вариант можно как в средах разработки, так и в редакторах изображений, что обычно быстрее. Фактически при этом создаются фальшивые снимки экрана, на которых и производят тестирование. Понятно, что как‑то модифицировать эти экраны затруднительно, так что лучше не увлекаться такой работой, не получив каких‑либо гарантий ее правильности. Эта версия прототипа больше соответствует этапу низкоуровневой разработки ПИ, но может применяться и на этапе высокоуровневой разработки.

Вместе с тем появились средства разработки прототипов, предельно близких к реальности. Сейчас имеется множество вполне доступных инструментальных средств, которые позволяют быстро создавать псевдореальные прототипы и моделировать разные ситуации. Эти средства значительно ускоряют разработку качественного ПО.

***Реальная версия прототипа.*** Иногда необходимо тестировать взаимодействие пользователя не только с интерфейсом системы, но и с обрабатываемыми системой данными. Например, работая с графической программой, пользователь не только нажимает на экранные кнопки, но также создает и модифицирует изображения мышью. Область же редактирования данных зачастую вообще не содержит каких‑либо визуальных интерфейсных элементов, из чего вовсе не следует, что интерфейса в ней нет, его, наоборот, много. Просто счет в нем идет не на кнопки и переключатели, а на пиксели и миллисекунды.

Понятно, что прототип в таких условиях практически не будет отличаться от готового ПИ. Поэтому лучше всего написать нужные участки программы до создания всего остального и проводить юзабилити‑тестирование на реальной версии прототипа ПИ. Разумеется, прототип такой версии, если он все‑таки разрабатывается, возможен только на этапе низкоуровневого проектирования.

**2. Порядок выполнения работы**

Согласно своему варианту выполнить задание практической работы:

*1.* Описать бумажную версию прототипа интерфейса. Распечатать отчет и защитить теоретическую часть.

*2.* Описать презентационную версию прототипа интерфейса. Распечатать отчет и защитить теоретическую часть.

*3.* Описать псевдореальную версию прототипа интерфейса. Распечатать отчет и защитить теоретическую часть.

*4.* Описать реальную версию прототипа интерфейса. Распечатать отчет и защитить теоретическую часть.

**Задание на работу**

1. Выбрать интерфейс материального продукта (DVD-плеера, стиральной машины и т.п.) или веб-сайта из числа тех, которые были проанализированы в ходе лабораторной работы №1.
2. С учётом результатов анализа конкурентов, сильных и слабых сторон интерфейса, изученных в ходе лабораторной работы №1, спроектировать новую, улучшенную версию интерфейса для выбранного продукта или веб-сайта.
3. Реализовать динамический прототип создаваемого интерфейса с использованием средств Adobe Flash.
4. На основе анализа задач пользователя, проведённого в ходе лабораторной работы №1, разработать задания для мини-тестирования прототипа с пользователем (представителем будущих целевых пользователей). В качестве пользователя, участвующего в тестировании, может выступать один из студентов вашей учебной группы. Продолжительность тестирования (исходя из объема заданий) должна составлять в районе 5-10 минут.
5. На основе результатов, полученных в ходе тестирования прототипа с пользователем, спроектировать и реализовать с использованием Adobe Flash новую, ещё более улучшенную версию прототипа интерфейса (итерация 2).
6. Составить отчёт по лабораторной работе.

**Приложение**

**Пример динамического прототипа интерфейса**

В качестве рассматриваемого примера выступает динамический прототип интерфейса стиральной машины (реальный продукт изображен на Рис. 3).



Рис. 3. Рассматриваемый продукт (стиральная машина) и его интерфейс.

В ходе анализа контекста и наблюдения за пользователем были отмечены следующие моменты:

* Пользователи в целом удовлетворены набором функций продукта и в основном используют только самые простые из них. Однако они хотели бы, чтобы эти основные функции выполнялись с б*о*льшим качеством.
* Нынешний интерфейс продукта сложно освоить, требуется обучение или продолжительная практика.
* Пользователям не кажется, что взаимодействие с продуктом сопровождается значительным количеством серьезных ошибок, но проблемы зачастую возникают – прежде всего потому, что несколько кнопок используется для переключения режимов или между различными значениями, однако полный перечень возможных значений нигде не отображается.

Также, с текущим интерфейсом были отмечены следующие проблемы:

* Необходимость нажимать кнопки по нескольку раз ведёт к ошибкам и требует от пользователей высокой концентрации внимания.
* Все режимы и варианты отображаются на экранчике, который становится «замусоренным». Часто затруднительно понять, какая кнопка и каким образом изменяет что-то на экране (и отсутствует перечень всех возможных значений и вариантов).
* Кнопки (элементы управления) расположены произвольным образом, без группировки.
* Варианты «по умолчанию» не всегда понятны, а также неясно, какие действия являются обязательными для начала стирки, а какие – нет.

**Статичный прототип**

Исходя из отмеченных недостатков интерфейса, был разработан статичный прототип новой версии интерфейса, представленный на Рис. 4.

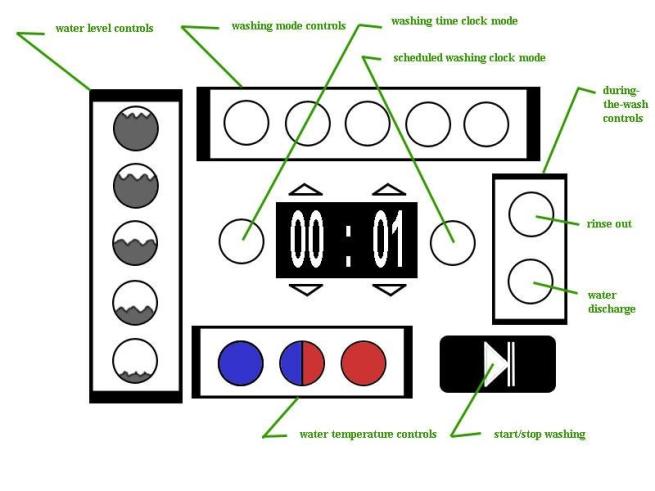


Рис. 4. Статичный прототип новой версии интерфейса продукта.

На Рис. 4. используются следующие обозначения:

* Water level controls: управление уровнем (объемом) воды.
* Washing mode controls: управление режимами стирки.
* Washing time clock mode: режим часов для отображения времени стирки.
* Scheduled washing clock mode: режим часов для программирования стирки по времени.
* During-the-wash controls: управляющие элементы, используемые во время стирки:

o Rinse out: отжим o Water discharge: спуск воды

* Water temperature controls: управление температурой воды.  Start/stop washing: начало/остановка стирки.

**Динамический прототип (версия 1)**

В соответствии со статическим прототипом и с использованием технологий Flash был разработан динамический прототип интерфейса (изображение представлено на Рис. 5).

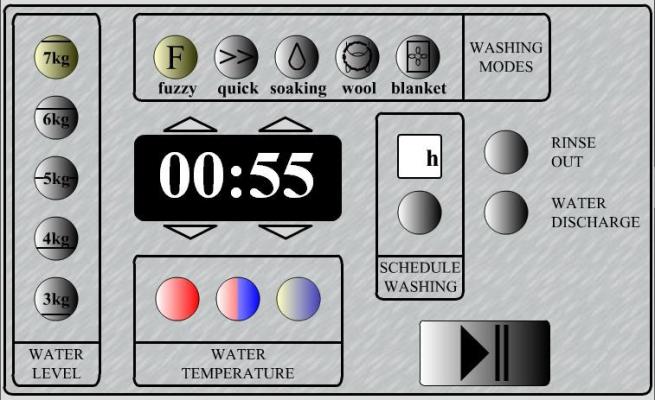


Рис. 5. Динамический прототип интерфейса продукта (версия 1).

Достоинства предлагаемой новой версии дизайна следующие:

* Решение не требует использования экрана. Все опции и значения по умолчанию для уровня воды, режимы стирки и т.д. могут быть отображены подсвеченными кнопками. К тому же, такое решение позволит пользователю видеть, какие установки являются обязательными для начала стирки, а также видеть все доступные установки (режимы и значения).
* Поскольку все кнопки (кроме стрелочек, используемых для установки часов) теперь имеют только два возможных состояния (нажата или не нажата), то ошибок, связанных с необходимостью множественных нажатий на кнопки, возможно избежать.
* Кнопки имеют более естественное расположение (особенно те, которые относятся к управлению уровнем воды – расположены от большего к меньшему по вертикали) и более правильно сгруппированы.

Для оценки прототипа было проведено тестирование с реальными пользователями, которое позволило сделать следующие выводы:

* Новый интерфейс легок в освоении и использовании, даже для пользователей, для которых английский язык (на котором сделаны подписи) не является родным.
* Общее впечатление у пользователей положительное, но они отмечают желательность изменения расположения кнопок. В частности, пользователи воспринимают «программирование» стирки как некоторый процесс (пошаговый). Таким образом, разделение интерфейса на группы кнопок, специально расположенные ассиметрично, чтобы показать их независимость, по-видимому является не лучшей идеей.
* Согласно пожеланиями пользователей, кнопка начала стирки могла бы быть подписана как Start, а не использовать символы начала и паузы. Однако такое решение имело бы и свои недостатки, т.к. пользователи не могли бы догадаться, что в принципе существует возможность паузы в процессе стирки.

**Динамический прототип (версия 2)**

Исходя из результатов тестирования прототипа с реальными пользователями, в него были внесены определённые корректировки и новая версия прототипа выглядит так, как показано на Рис. 6.

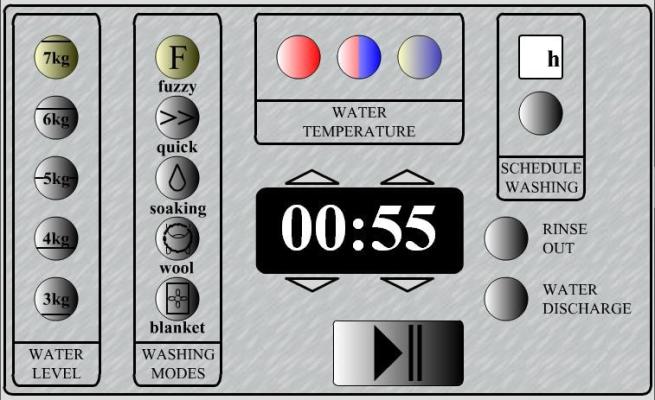


Рис. 6. Динамический прототип интерфейса продукта (версия 2, улучшенная).

**Задание.** Согласносвоей темы (см. практическая работа 1) создать 2 разные версии прототипа интерфейса.

**3. Содержание отчета**

1. Титульный лист

2. Тема и цель работы

3. Выполнение задания, согласно своего варианта

4. Ответы на контрольные вопросы

5. Выводы

**Контрольные вопросы**

1. Что такое прототипирование интерфейсов?
2. Какие вы знаете версии прототипов интерфейсов
3. Опишите бумажную версию прототипа интерфейса
4. Опишите презентационную версию прототипа интерфейса
5. Опишите псевдореальную версию прототипа интерфейса
6. Опишите реальную версию прототипа интерфейса
7. Какие прототипы называются статичные прототипы?
8. Какие прототипы называются динамические прототипы?

**Практическая работа № 3.**

**Принципы и правила использования компьютерной техники. Построение прототипа интерфейса.**

**Цель работы:** получение навыков проектирования прототипов прикладного программного обеспечения автоматизированных систем.

**Задачи работы**: освоение принципов применения программного продукта EaSynth ForeUI для проектирования прикладного программного обеспечения.

**1.Теоретические сведения.**

**Интерфейс пользователя (UI**) **–** это часть программы, которая находится на виду у пользователя и призвана обеспечивать отображение данных, управление или диалог с пользователем. Пр иразработке программного обеспечения программисты не должны оставлять дизайн интерфейса пользователя "на потом", считая, что в программе только код должен быть краеугольным камнем. Время от времени у пользователей возникает чувство раздраженности, например, даже из-за неудачно подобранных шрифтов, неудобного расположения элементов управления, отображения данных. Для удобства пользователей должно быть затрачено n-ое количество часов, так как это серьезная работа. Это следует учитывать при озвучивании сроков разработки программы.

**Удобство пользовательского интерфейса** входит в такое понятие как "Эргономика", которое включается в процессы разработки и тестирования программного продукта как часть системы качества. Разработка пользовательского интерфейса должна вестись совместно с дизайном программного продукта в целом.

**Процесс разработки пользовательского интерфейса** разбивается на этапы жизненного цикла:

* Анализ трудовой деятельности пользователя, объединение бизнес-функций в роли.
* Построение пользовательской модели данных, привязка объектов к ролям и формирование рабочих мест.
* Формулировка требований к работе пользователя и выбор показателей оценки пользовательского интерфейса.
* Разработка обобщенного сценария взаимодействия пользователя с программным модулем (функциональной модели) и его предварительная оценка пользователями и Заказчиком.
* Корректировка и детализация сценария взаимодействия, выбор и дополнение стандарта (руководства) для построения прототипа.
* Разработка макетов и прототипов ПИ и их оценка в деловой игре, выбор окончательного варианта.
* Имплементация ПИ в коде, создание тестовой версии.
* Разработка средств поддержки пользователя (пользовательские словари, подсказки, сообщения, помощь и пр.) и их встраивание в программный код.
* Usability тестирование тестовой версии ПИ по набору раннее определенных показателей.
* Подготовка пользовательской документации и разработка программы обучения.

**Юзабилити**, удобство пользовательского интерфейса. Программа разрабатывается для обеспечения работы пользователя, т. е. для того, чтобы он, с помощью компьютерной программы, быстрее, надежнее и качественнее решал свои производственные задачи. С точки зрения удобства пользовательского интерфейса в программе важно создать такой пользовательский интерфейс, который сделает работу эффективной и производительной, а также обеспечит удовлетворенность пользователя от работы с программой.

Разработчику необходимо тщательно продумать и осознать сценарий взаимодействия программы с пользователем, приведя его к оптимальной системе выполнения задач, и реализовать пользовательский интерфейс в соответствии с этой системой.

Для того, чтобы разобраться в технологии решения задач пользователя, разработчику необходимо выяснить следующие моменты:

* Какая информация необходима пользователю для решения задачи?
* Какую информацию пользователь может игнорировать (не учитывать)?
* Совместно с пользователем разделить всю информацию на сигнальную, отображаемую, редактируемую, поисковую и результирующую.
* Какие решения пользователю необходимо принимать в процессе работы с программой?
* Может ли пользователь совершать несколько различных действий (решать несколько задач) одновременно?
* Какие типовые операции использует пользователь при решении задачи?
* Что произойдет, если пользователь будет действовать не по предписанному Вами алгоритму, пропуская те или иные шаги или обходя их?

Дизайн пользовательского интерфейса должен обеспечивать минимизацию усилий пользователя при выполнении работы и приводить к:

* сокращению длительности операций чтения, редактирования и поиска информации,
* уменьшению времени навигации и выбора команды,
* повышению общей продуктивности пользователя, заключающейся в объеме обработанных данных за определенный период времени.
* увеличению длительности устойчивой работы пользователя и др.

**2. Порядок выполнения работы**

1. Запустить программу ForeUI:

Пуск → Программы → EaSynth → ForeUI → окно программы EaSynth ForeUI

1. Создать новый проект:

ф. EaSynth ForeUI | File → New Plot → ф. Untitled-1;

File → Save Plot → ф. Save | выбрать расположение и ввести имя проекта; кн. Save → форма проекта

1. Ввести заголовок приложения:

ф. EaSynth ForeUI | выбр. вкл. Elements → выбр. Text Label и перетащить на форму проекта → элемент Text Label на форме пр. кн. мыши на элементе Text Label → Edit Text | ввести текст «Агентство путешествий»; кн. Ok

кн. Tools | ф. Tools Panel | Pos.: X ← 19; Y ← 20

Size: W ← 488; H ← 48

Text: Arial Narrow; 35; Цвет ← #808080 выбр. вкл. Elements → выбр. Text Label и перетащить на форму проекта → элемент Text Label на форме

пр. кн. мыши на элементе Text Label → Edit Text | ввести текст «SPACE TRAVEL»; кн. Ok

кн. Tools | ф. Tools Panel |

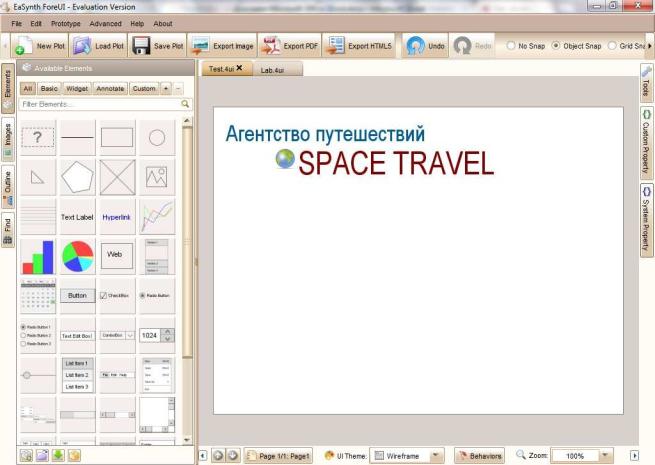
Pos.: X ← 132; Y ← 58

Size: W ← 488; H ← 48

Text: Arial Narrow; 50; Цвет ← #800000

1. Добавить логотип:

ф. EaSynth ForeUI | выбр. вкл.Images → кн. «+» (Add Image) | From Icon Library… PC.DE Essen (32×32); выбрать World; перенести логотип на форму кн. Tools | ф. Tools Panel | Pos.: X ← 95; Y ← 64



1. Запустить имитацию приложения:

ф. EaSynth ForeUI | кн. Save Plot; выбрать пункт меню Prototype → Run Simulation → запуск браузера и просмотр страницы

1. Добавить элемент Tabs на форму

ф. EaSynth ForeUI | выбр. вкл. Elements → выбр. Tabs и перетащить на форму проекта → элемент Tabs на форме пр. кн. мыши на элементе Tabs → Edit Tabs Titles | ввести текст «Бронирование,

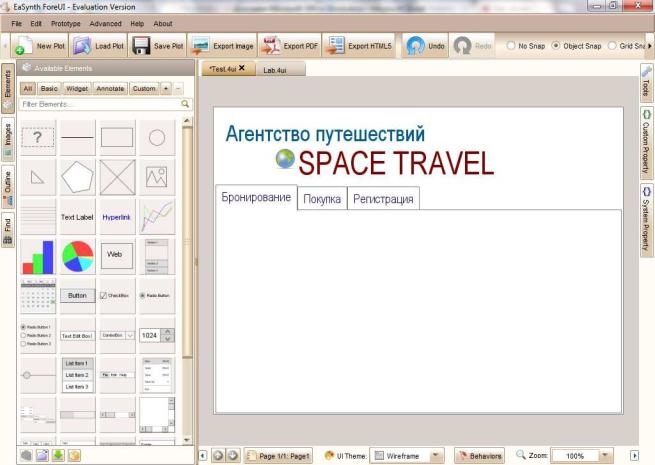
Покупка, Регистрация»; кн. Ok

кн. Tools | ф. Tools Panel |

Pos.: X ← 3; Y ←121

Size: W ← 635; H ← 358

Text: Arial Narrow; 19.5; Цвет ← #0F0078



1. Сформировать закладку «Бронирование»:
   1. Добавить на вкладку элемент Group:

ф. EaSynth ForeUI | выбр. вкл. Elements → выбр. Group Frame и правой кнопкой мыши перетащить на форму проекта (во вкладку «Бронирование») → элемент Group Frame на вкладке пр. кн. мыши на элементе Group Frame → Edit Group Title | ввести текст «Куда и Когда»; кн. Ok

кн. Tools | ф. Tools Panel |

Pos.: X ← 16; Y ←166

Size: W ← 447; H ← 131

* 1. Добавить радиокнопки («Туда и обратно», «В одну сторону»):

ф. EaSynth ForeUI | выбр. вкл. Elements → выбр. Radio Button Group и правой кнопкой мыши перетащить на форму проекта (во вкладку «Бронирование», в группу «Куда и Когда») → элемент Radio Button Group на вкладке пр. кн. мыши на элементе Radio Button Group → Edit Radio Buttons | ввести текст

«Туда и обратно В одну сторону»; кн. Ok

кн. Tools | ф. Tools Panel |

Pos.: X ← 29; Y ←199

List: Row Height ← 447; H ← 131

* 1. Добавить раскрывающийся список для пункта отправления:

ф. EaSynth ForeUI | выбр. вкл. Elements → выбр. Text Label и правой кнопкой мыши перетащить на форму проекта (во вкладку «Бронирование», в группу «Куда и Когда») → элемент Text Label на форме пр. кн. мыши на элементе Text Label → Edit Text | ввести текст «Откуда»; кн. Ok кн. Tools | ф. Tools Panel |

Pos.: X ← 174; Y ← 201

Size: W ← 55; H ← 12

ф. EaSynth ForeUI | выбр. вкл. Elements → выбр. ComboBox и правой кнопкой мыши перетащить на форму проекта (во вкладку «Бронирование», в группу «Куда и Когда») → элемент ComboBox на форме пр. кн. мыши на элементе ComboBox → Edit ComboBox Text | ввести текст

«Выбрать + три любых города»; кн. Ok кн. Tools | ф. Tools Panel |

Pos.: X ← 224; Y ← 195 Size: W ← 232; H ← 25

7.4. Аналогичным образом добавить раскрывающийся список для пункта назначения:

Параметры для TextBox:

Pos.: X ← 174; Y ← 234 Size: W ← 55; H ← 12 Параметры для ComboBox:

Pos.: X ← 224; Y ← 228 Size: W ← 232; H ← 25

7.5. Добавить поле для выбора даты отправления:

ф. EaSynth ForeUI | выбр. вкл. Elements → выбр. Text Label и правой кнопкой мыши перетащить на форму проекта (во вкладку «Бронирование», в группу «Куда и Когда») → элемент Text Label на форме пр. кн. мыши на элементе Text Label → Edit Text | ввести текст «Туда»; кн. Ok кн. Tools | ф. Tools Panel |

Pos.: X ← 30; Y ← 272

Size: W ← 55; H ← 12

ф. EaSynth ForeUI | выбр. вкл. Elements → выбр. Text Label и правой кнопкой мыши перетащить на форму проекта (во вкладку «Бронирование», в группу «Куда и Когда») → элемент Text Label на форме пр. кн. мыши на элементе Text Label → Edit Text | ввести текст «Туда»; кн. Ok кн. Tools | ф. Tools Panel |

Pos.: X ← 30; Y ← 272

Size: W ← 55; H ← 12

ф. EaSynth ForeUI | выбр. вкл. Elements → выбр. Text Edit Box и правой кнопкой мыши перетащить на форму проекта (во вкладку «Бронирование», в группу «Куда и Когда») → элемент Text Edit Box на вкладке

кн. Tools | ф. Tools Panel |

Pos.: X ← 80; Y ←270

Size: W ← 67; H ← 22

ф. EaSynth ForeUI | выбр. вкл. Elements → выбр. Calendar и правой кнопкой мыши перетащить на форму проекта (во вкладку «Бронирование», в группу «Куда и

Когда») → элемент Calendar на вкладке

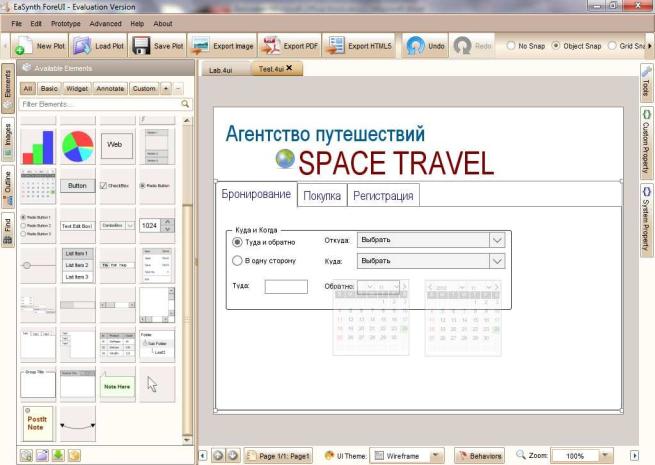
кн. Tools | ф. Tools Panel |

Pos.: X ← 186; Y ←269

Size: W ← 120; H ← 120

State: Visible (снять метку)

7.6. Аналогичным образом добавить поле для выбора даты прибытия.



7.7. Добавить правила обработки элементов (при нажатии на поле ввода для даты отправления открывается календарь, при выборе даты заполняется поле ввода, а сам календарь закрывается):

ф. EaSynth ForeUI | форма проекта → выбр. текстовое поле для даты отправления, нажать [Ctrl + D] → ф. Defined Behaviors

ф. Defined Behaviors | Add Behavior → For Elements…

ф. Select Elements as Behavior Owner… | выбр. TextEditBox\_1; кн. Select 1 Element

ф. Defined Behaviors | For Element: TextEditBox\_1; кн. Add Event → выбр. Element Clicked; кн. Add Action | выбр. Operate on Element… → ф. Manipulate Elements…:

Step 1: Choose elements to manipulate ← Calendar\_1

Step 2: Select the action to be created ← Change Visibility

Step 3: Specify parameters of the actions ← Make Element Visible;

кн. Ok

ф. EaSynth ForeUI | форма проекта → выбр. календарь для даты отправления, нажать [Ctrl + D] → ф. Defined Behaviors

ф. Defined Behaviors | Add Behavior → For Elements…

ф. Select Elements as Behavior Owner… | выбр. Calendar\_1; кн. Select 1 Element

ф. Defined Behaviors | For Element: Calendar\_1; кн. Add Event → выбр. Calendar Date Changed; кн. Add Action | выбр. Operate on Element… → ф. Manipulate Elements…:

Step 1: Choose elements to manipulate ← TextEditBox\_1

Step 2: Select the action to be created ← Set Value of TextEditBox

Step 3: Specify parameters of the actions | Please input the new value of

TextEditBox: кн. […] → Insert Element Property… →

ф. Select an Element to Provide the Property… ← выбр. Calendar\_1; кн.

Select 1 Element

ф. Choose Element Property to Insert… ← Current Selected Date in

Calendar…; кн. Ok →

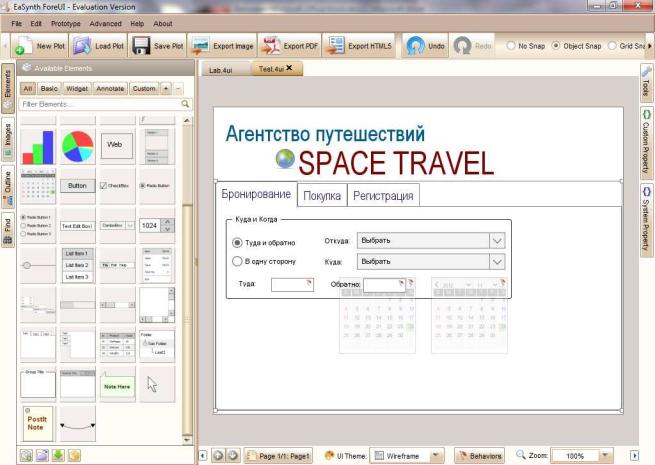
ф. Manipulate Elements… | кн. Ok кн. Add Action | выбр. Operate on Element… → ф. Manipulate Elements…:

Step 1: Choose elements to manipulate ← Calendar\_1

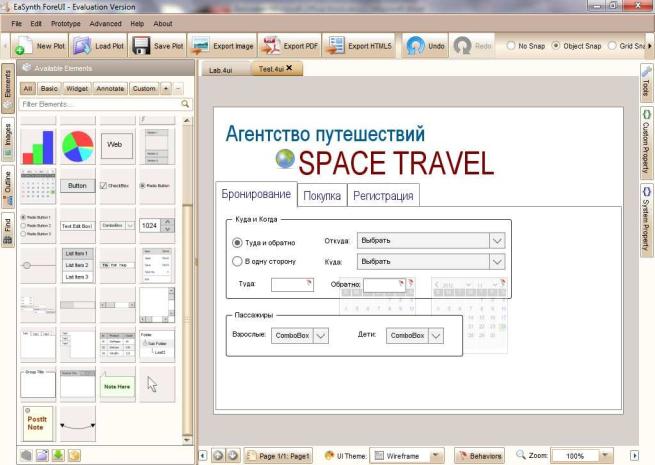
Step 2: Select the action to be created ← Change Visibility

Step 3: Specify parameters of the actions ← Make Element Hidden кн. Ok → ф. Manipulate Elements

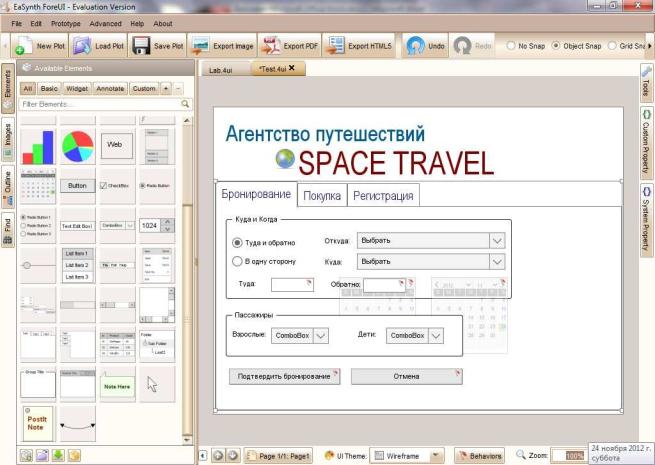
7.8. Аналогичным образом добавить правила обработки выбора даты прибытия.



7.9. Аналогичным образом добавить группу элементов «Пассажиры» для указания числа путешествующих.



7.10. Добавить кнопки «Подтвердить бронирование» и «Отмена». При нажатии на первую кнопку должно выводиться сообщение «Бронирование подтверждено». При нажатии на кнопку «Отмена» должны быть обнулены все выбранные значения.



1. Запустить имитацию приложения (см. п. 5).
2. Аналогичным образом сформировать закладки «Покупка» и «Регистрация».
3. Самостоятельно добавить еще 2 вкладки на основании предыдущих лабораторных работ.

**Задание.** Согласносвоей темы (см. практическая работа 1) построить прототип интерфейса.

**3. Содержание отчета**

1. Титульный лист

2. Тема и цель работы

3. Краткую теоретическую часть

4. Экранные формы работы с ForeUI

5. Выводы

**Контрольные вопросы:**

1. Что такое интерфейс пользователя (UI) – это
2. Что такое Эргономика?
3. На какие этапы ЖЦ разбивается процесс разработки пользовательского интерфейса?
4. Что такое юзабилити?
5. Для того, чтобы разобраться в технологии решения задач пользователя, разработчику
6. Какая информация необходима пользователю для решения задачи?
7. Какую информацию пользователь может игнорировать (не учитывать)?
8. Совместно с пользователем разделить всю информацию на сигнальную, отображаемую, редактируемую, поисковую и результирующую.
9. Какие решения пользователю необходимо принимать в процессе работы с программой?
10. Может ли пользователь совершать несколько различных действий (решать несколько задач) одновременно?
11. Какие типовые операции использует пользователь при решении задачи?
12. Что произойдет, если пользователь будет действовать не по предписанному Вами алгоритму, пропуская те или иные шаги или обходя их?
13. Дизайн пользовательского интерфейса должен обеспечивать минимизацию усилий

**Практическая работа № 4.**

**Аудит удобства, тест. Закон Хика. Использование закона Хика на практике.**

**Целью данной работы является**

1) экспериментальная проверка выполнения закона Хика в графическом интерфейсе пользователя;

2) изучение факторов и параметров графического интерфейса, влияющих на длительность интеллектуальных действий пользователя.

**1.Теоретические сведения.**

**Общие сведения**

Чем больше вариантов, тем больше времени уходит на выбор между ними. Эта аксиома, хорошо известная разработчикам интерфейсов как закон Хика, должна учитываться при написании программного обеспечения. Однако при работе со многими программными продуктами возникает ощущение, что их цель – отнять как можно больше времени у пользователя, или что закон Хика при их создании был забыт.

Скорость интеллектуальной работы пользователей является одним из основных критериев эргономичности интерфейса. Она напрямую зависит от функциональной сложности и структуры современных интерфейсов. Обилие элементов управления замедляет восприятие интерфейса пользователем и увеличивает вероятность его ошибки.

Длительность интеллектуальной работы состоит из длительности:

1) постановки задачи;

2) формирования алгоритма решения;

3) мыслительных действий по выполнению алгоритма;

4) восприятия информации (реакции);

5) интерпретации информации.

Значительная часть общего времени человеко-компьютерного взаимодействия расходуется на выполнение интеллектуальных действий по управлению интерфейсом. Скорость самого мышления пользователей повысить практически невозможно, но можно уменьшить влияние факторов, замедляющих процесс мышления.

Для повышения скорости интеллектуальных действий и снижения числа мыслительных ошибок в практике проектирования интерфейсов используются:

1) методы быстрого возвращения фокуса внимания пользователя после его отвлечения от выполняемой задачи;

2) непосредственное манипулирование объектами с помощью мыши;

3) методы ограничения принятия решений;

4) закон Хика.

Большинство из перечисленных методик выражены в виде рекомендаций и позволяют лишь качественно оценить проектируемый интерфейс с помощью привлечения соответствующих экспертов. Уникальность закона Хика заключается в том, что он, в отличие от других методов, позволяет формализовать зависимость скорости мыслительных действий (реакции) пользователя от параметров интерфейса. Данные, получаемые на его основе, позволяют более обоснованно принимать решения, связанные с компоновкой элементов меню, кнопок, полей ввода и других визуальных компонентов внутри экранных форм.

Этот закон назван именем английского психолога, доктора медицины, члена Британской академии психологических наук Хика (Hick) Вильяма Эдуарда (1912 – 1975). Он работал в Кембриджском университете по проблемам экспериментальной психологии и в 1952 г. совместно с Р. Хайманом вывел закон, связывающий *время реакции* выбора из альтернативных действий на равновероятные раздражители с числом альтернатив.

*Время реакции – интервал между предъявлением раздражителя и началом ответной реакции*, которая обычно фиксируется в двигательной сфере. Для раздражителей различного типа время реакции различно: наиболее быстрая реакция реализуется в ответ на слуховые раздражители, самая медленная – на обонятельные, вкусовые и температурные. Минимальное время реакции у здорового взрослого человека приблизительно равно *100* *мс*. Оно существенно зависит от сложности задачи, решаемой при *опознании* раздражителя.

*Опознание – процесс отнесения воспринимаемого предмета к какому-либо определенному классу*, за счет чего происходит построение осмысленного образа. В ходе практики каталог опознавательных признаков постоянно пополняется, что позволяет классифицировать все большее количество объектов.

**Закон Хика**

Закон Хика – подтвержденная экспериментально теория о том, что количество информации, которая поступает в мозг человека, влияет на время, которое она тратит на принятие решения.

Данную взаимосвязь описал немецкий психолог И. Меркель в конце XIX века, однако на тот момент эмпирическая база для ее доказательства отсутствовала. Этот пробел восполнил в 1952 году немецкий исследователь В.Е.Хик, представив эту закономерность в виде логарифмической функции. Согласно **закону Хика**, любое решение требует определенных временных затрат – чем сложнее решение, тем больше времени необходимо для его принятия. Иными словами, один элемент всегда лучше, чем два: например, один элемент с 8 составляющими будет быстрее воспринят, чем два элемента с 4 составляющими каждый.

Методы теории информации, применяемые при создании логарифмической функции, позволили использовать ее и в случаях с неравновероятными сигналами. Основная сфера реализации закона Хика – инженерная психология и эргономика. Данный закон позволяет анализировать деятельность оператора: рассчитывать время, необходимое ему для выполнения определенной задачи, определять скорость получения им информации с учетом его психофизиологических возможностей.

Также **закон Хика** в настоящее время широко применяется в информационно-технологической сфере: например, при разработке интерфейса сайта разработчики стремятся к минимализму, руководствуясь основным положением о том, что многочисленный выбор обуславливает долгий процесс принятия решения.

Закон Хика утверждает, что предоставление пользователю нескольких вариантов выбора одновременно является более эффективным, чем организация этих же вариантов в иерархические группы.

Формулировка закона Хика для выбора из n равновероятных вариантов:

**t (время выбора) (мс) = a+b\*log***2***(n+1)**

t — это общее время реакции,

a и b — константы, которые описывают индивидуальные особенности восприятия, такие как задержка перед выполнением задания и индивидуальный коэффициент скорости принятия решения,

n — количество равнозначных альтернативных вариантов, из которых нужно выбрать.

Для примера сравним время выбора из одного меню, включающего 8 пунктов, и время выбора из двух меню, состоящих из 4 элементов каждое. Пункты меню одни и те же.

• Для меню из 8 пунктов:

t (мс) = a+b\*log***2***(8+1) ≈ a+b\*log***2***(8) = a+3\*b

• Для двух меню из 4 пунктов:

t (мс) = 2\*(a+b\*log***2***(4+1)) ≈ 2\*(a+b\*log***2***(4)) = 2\*(a+2\*b)

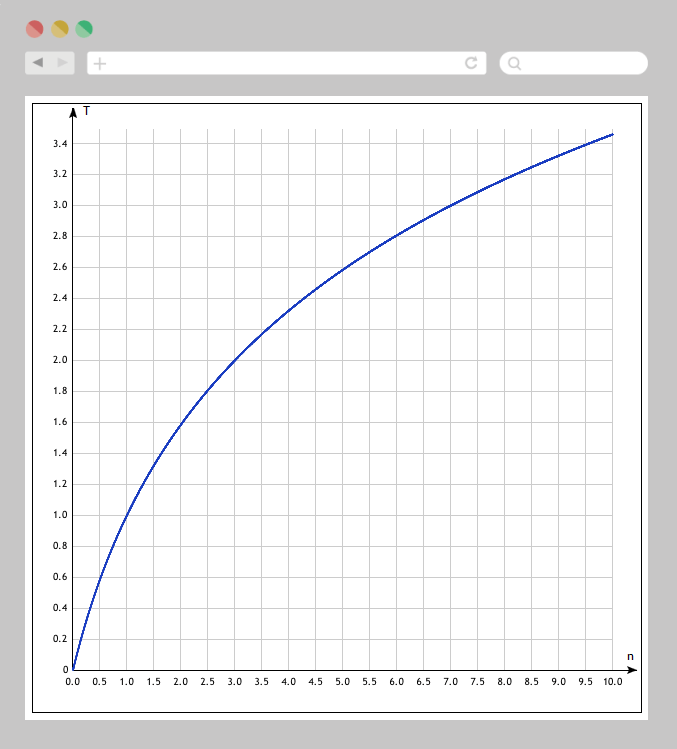
Здесь, пренебрегая единицей в выражении под знаком логарифма, во втором случае мы уменьшили время выбора значительнее, чем в первом.

Несмотря на это, неравенство a+3\*b < 2\*(a+2\*b) или, что то же, –a –b < 0 справедливо.

В этой весьма огрубленной оценке не учтено время на открытие вместо одного двух меню во втором случае. Кроме того, при этом изменяется локус внимания и увеличивается нагрузка на кратковременную память.

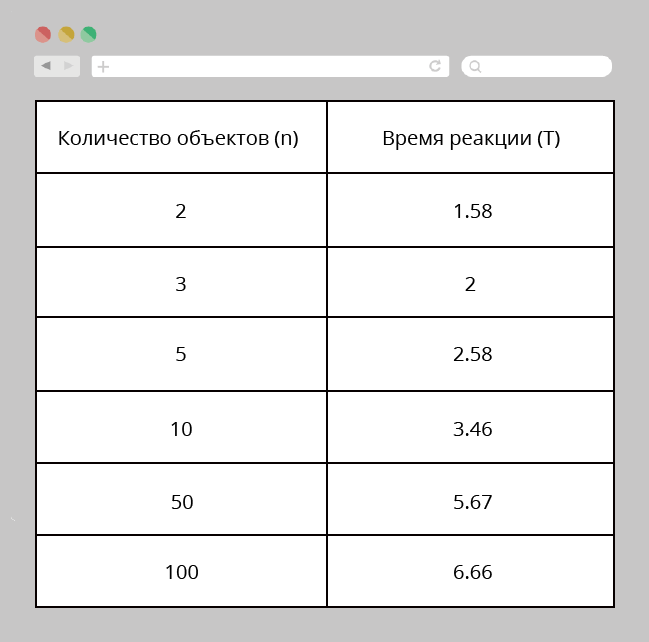
**Что это значит**

Для наглядности давайте построим график. Если не учитывать переменные a и b, которые зависят исключительно от индивидуальных особенностей человека, то он будет выглядеть так:



Вертикальная ось T — это время реакции, горизонтальная ось n — количество альтернативных объектов выбора.

Вот как меняется время реакции при увеличении количества объектов:



Мы видим, что время реакции при увеличении объектов с 2 до 5 выросло на 1 условную единицу. А теперь обратите внимание, что на эту же 1 условную единицу увеличилось и время реакции при увеличении объектов с 50 до 100. В этом и заключается логарифмическая зависимость.

Чем меньше объектов, тем быстрее и проще выбрать из них нужный. Но при увеличении количества объектов свыше определенного числа время реакции меняется незначительно.

**Общее применение закона Хика**



При проектировании интерфейсов закон Хика помогает определить оптимальное количество объектов в однородном массиве, например, в меню. Обычно он применяется в связке с [законом Фиттса](https://askusers.ru/blog/pravila/zakon-fittsa-v-proektirovanii-interfeysov/" \t "_blank), который помогает определить оптимальный с позиции скорости реакции размер элемента.

Закон Хика также тесно связан с другими принципами восприятия и психологическими особенностями принятия решений. Его можно одинаково эффективно рассматривать в контексте [теории близости](https://askusers.ru/blog/pravila/teoriya-blizosti-interfeysov/" \t "_blank) и [правила 7 +/- 2](https://askusers.ru/blog/pravila/7-2-vazhnoe-pravilo/" \t "_blank), а также других моделей поведения пользователя на сайте.

Вне интернет-среды принципы закона Хика из психологии реализованы практически в любом дизайне интерфейса, взаимодействующим с пользователем: начиная от панели управления микроволновкой и заканчивая расположением и количеством кнопок на пульте для телевизора.

**Особенности комплексного применении законов Хика и Фиттса в UX**



Обратите внимание, законы описывают действия, которые обычно следуют одно за другим.

1. Сразу нам нужно определиться с выбором (закон Хика).
2. А затем — попасть в нужный элемент (закон Фиттса).

Таким образом, общее время можно вычислить как сумму значений двух формул.

В контексте UX это значит следующее:

* Одно длинное меню (или расположение однородных элементов в одном блоке) удобнее для пользователя, чем два или несколько отдельных.
* При проектировании интерфейса нужно учитывать оба закона и стараться оптимизировать как размеры и положение блоков, так и количество элементов в каждом блоке.
* Ориентироваться на законы также можно при [составлении и оптимизации профилей задач.](https://askusers.ru/blog/pravila/profili-zadach-v-ux/" \t "_blank) Особенно показателен закон Хика при анализе процесса локализации и заполнения полей форм.

Закон Хика менее известен, чем закон Фиттса. Однако он его прекрасно дополняет и помогает проектировать взаимодействие с пользователем более осознанно и эффективно.



**Выводы и рекомендации**

* Чем больше объектов, тем больше времени нужно пользователю, чтобы выбрать из них нужный.
* Зависимость между временем реакции и количеством альтернатив выбора описывается логарифмической функцией.
* Закон Хика позволяет рассчитать оптимальное количество объектов в блоке.
* Применение закона Хика в дизайне в связке с законом Фиттса позволяет более точно спрогнозировать время реакции пользователя в ходе взаимодействия с интерфейсом.

Игнорировать законы Хика и Фиттса — все равно что стрелять с закрытыми глазами. Попасть можно только случайно. Если подходить к проектированию интерфейса серьезно, то оба принципа и описывающие их формулы помогут вам создать действительно эффективное решение.

**2. Порядок выполнения работы**

**Задание**

1. Согласно описанным методам произвести оценку быстродействия и эффективности разработанного интерфейса приложения: определить сколько времени затратит пользователь для достижения указанной в ТЗ функциональности, определить сколько лишней информации содержится в интерфейсе.
2. На основе полученных данных, определить, насколько эффективной является выбранная модель интерфейса. Придумать способы увеличения эффективности, если это возможно.
3. На основе проведенных исследований принять решение о выпуске программного продукта или отправке его на доработку. Обосновать свой выбор.
4. Подготовить отчет в свободной форме. При формировании отчета использовать полученные на занятии знания.

**3. Содержание отчета**

1. Титульный лист

2. Тема и цель работы

3. Выполнение задания

4. Ответы на контрольные вопросы

5. Выводы

**Контрольные вопросы**

1. Назовите составляющие длительности выполнения интеллектуальной работы.

2. Какие факторы наиболее существенно влияют на скорость интеллектуальной работы пользователя?

3. Что такое время реакции?

4. Что такое опознание объекта?

5. Сформулируйте закон Хика.

6. Запишите формулу для расчета времени выбора объекта по закону Хика (для объектов с равной вероятностью выбора).

7. Запишите формулу для расчета времени выбора объекта по закону Хика (для объектов с различной вероятностью выбора).

8. Когда начинается и когда заканчивается отчет времени по закону Хика?

9. Какие способы выделения объектов в интерфейсах наиболее эффективны?

**Практическая работа № 5.**

**Человеческие факторы. Закон Фитса. Факторы, влияющие на длительность физических действий пользователя.**

**Цель работы:**

1. Изучить закон Фитса
2. Привести примеры закона Фитса
3. Применение закона Фитса на практике

**1.Теоретические сведения.**

**Закон Фитса**

Закон Фиттса – закон, названный по имени его создателя, относительно связи между пространственным расстоянием до цели и скоростью осуществления действия по отношению к ней.

В 1954 году Пол Фиттс, американский психолог, сформулировал закон, который до сих пор считается лучшей математической моделью, описывающей движения человека при достижении цели в двумерном пространстве. Пол Фиттс сформулировал закон таким образом: время, которое необходимо для позиционирования на какой-то предмет, представляет собой функцию от расстояния до данного элемента и его размера. Иными словами, чем дальше цель от субъекта, тем больше времени потребуется на ее достижение и, соответственно, для этого нужно больше коррекции. В итоге большее количество времени затрачивается и на внесение самой коррекции.

На практике **закон Фиттса** можно продемонстрировать на примере человека, страдающего от никотиновой зависимости. Специалисты дадут ему рекомендацию не хранить сигареты дома. Если ему захочется курить ночью, то ему придется столкнуться с выбором: пересилить свое желание или пойти в магазин, который находится на темной улице и дорога к которому грозит обернуться встречей с сомнительными лицами. Цель (сигареты) в данном случае достаточно далека, анализ этапов ее достижения приводит к внесению корректив, которые, скорее всего, заставят человека все же остаться дома.

**Закон Фиттса** популярен не только в гештальт-психологии: в настоящее время он широко применяется при работе с дизайном сайтов. Здесь закон влияет на внешний вид кнопок, расположенных на сайте: самые необходимые должны быть в свободном доступе (располагаться в том месте, где пользователь скорее всего ожидает их найти и быть достаточно крупными для удобного визуального восприятия).

• Закон Фитса утверждает, что чем дальше находится объект от текущей позиции курсора или чем меньше размеры этого объекта, тем больше времени требуется для перемещения к нему курсора[[1]](#footnote-0).

Пусть S – размер объекта вдоль линии перемещения курсора, D – расстояние от начальной позиции курсора до ближайшей точки целевого объекта (*дистанция*). Иллюстрация:

Цель

S

S

D

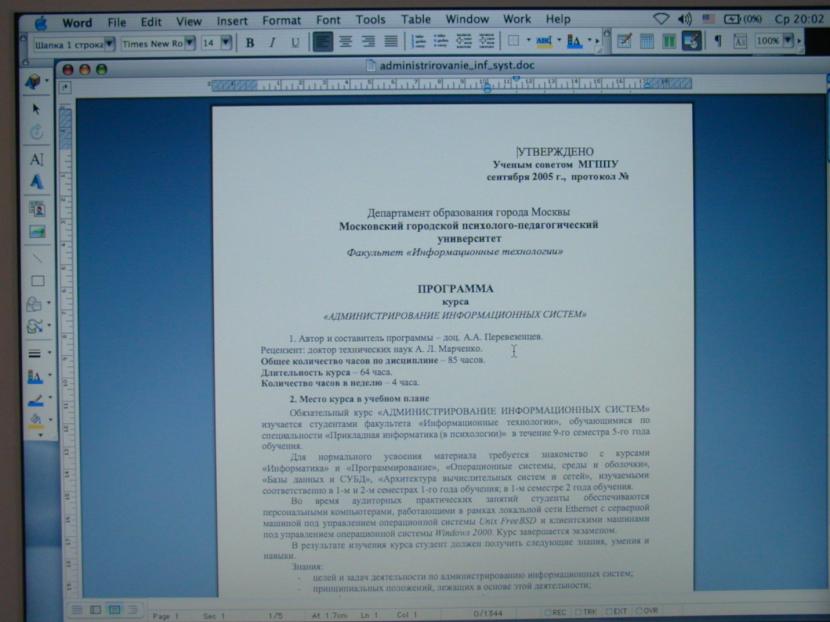
Начальная позиция курсора

Тогда закон Фитса формулируется следующим образом:

**время для перемещения курсора к объекту (мс) = a+b\*log*2*(D/S+1)**

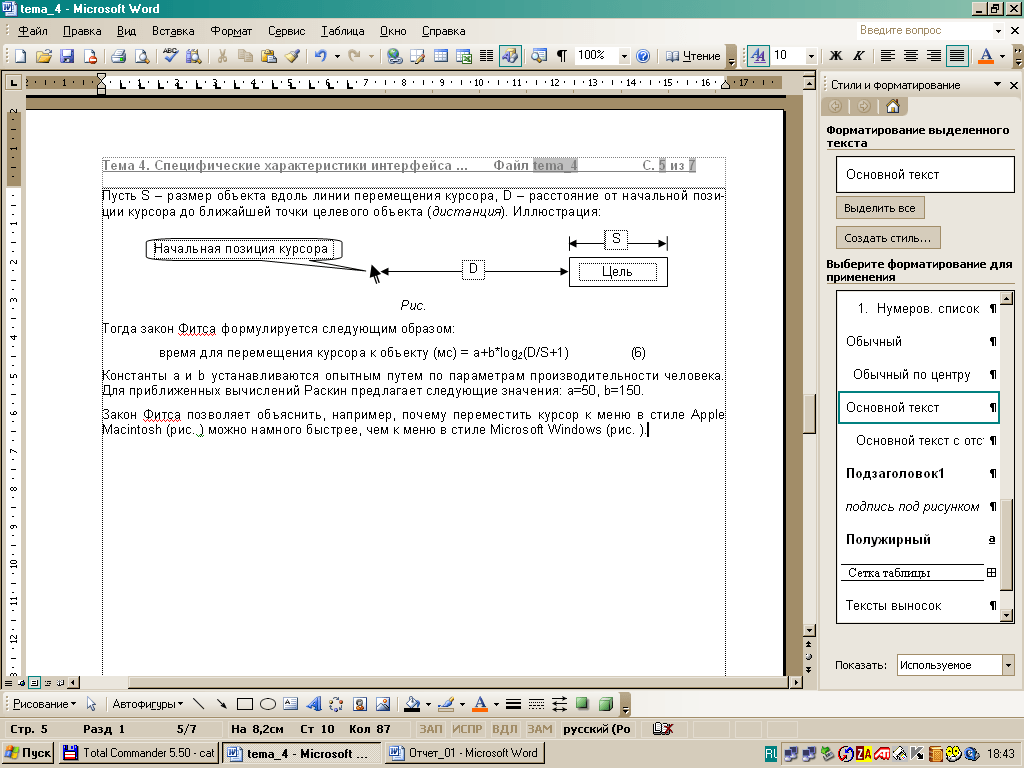
Константы a и b устанавливаются опытным путем по параметрам производительности человека. Для приближенных вычислений Раскин предлагает следующие значения: a=50, b=150.

• Закон Фитса позволяет объяснить, например, почему переместить курсор к меню в стиле Apple Macintosh (рис. 1) можно намного быстрее, чем к меню в стиле Microsoft Windows (рис. 2).



*Рис. 1. Меню в Macintosh находится непосредственно у верхней границы экрана.*

*Ширина полосы меню как бы не ограничена сверху. Достаточно подвести курсор выше нижней границы полосы меню*



Размер цели S

*Рис. 2. Меню в Windows находится ниже верхней границы экрана.   
Курсор надо подводить более точно, согласуясь с шириной полосы меню*

Размер S меню в Windows составляет 0.5 – 0.8 мм, и именно в этих пределах следует разместить курсор ГУВ, чтобы открыть подменю. В Macintosh достаточно превысить расстояние до нижней границы меню; вверх можно передвигаться сколько угодно («бесконечно большой элемент») – курсор в любом случае остановится на границе экрана.

Приведенные в кн. Раскина результаты расчетов и экспериментальные данные согласуются: в среднем пользователю требуется около 0.6 с для открытия Apple-меню и более 1 с – для открытия Windows –меню. Этот анализ объясняет, почему при разработке Macintosh меню были намеренно помещены на границе экрана.

• Закон Фитса предполагает две стратегии оптимизации:

* размещать объекты ближе друг к другу;
* делать далекие объекты больше.

Реализации этих стратегий различны, но достаточно просты и эффективны.

•• В отличие от стандартных меню, которые распахиваются только в одном направлении, можно сделать «меню-распашонку», распахивающееся в обоих направлениях. При этом среднее расстояние, преодолеваемо мышью для достижения целевого объекта, сокращается вдвое.

Исследования в области внедрения таких меню весьма обширны, и они (меню) могли бы быть стандартизированы очень давно.

•• Далекие объекты можно делать больше по размеру (рис. 3).

.

*Рис. 3. Далекие объекты больше по размеру*

Усовершенствовать этот вариант можно за счет «липких границ», не выпускающих указатель из фокуса элемента меню (на рис. 4 эти границы изображены черными жирными линиями).

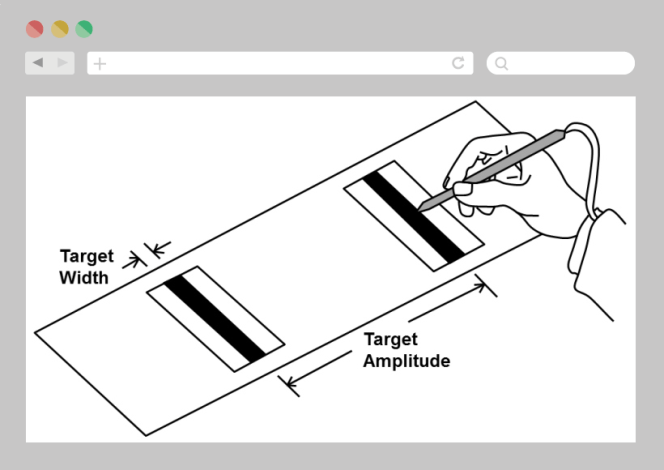
•• Можно расположить элементы меню на равных расстояниях от текущей позиции указателя («вокруг»). Вариант такого гипотетического меню с уменьшенным средним временем выборки приведен на рис.

*Рис. 4. Вариант гипотетического меню с уменьшенным средним временем выборки*

В результате эксперимента ученый выявил интересную закономерность: чем больше расстояние и чем меньше размер до цели, тем больше времени требуется для ее достижения.

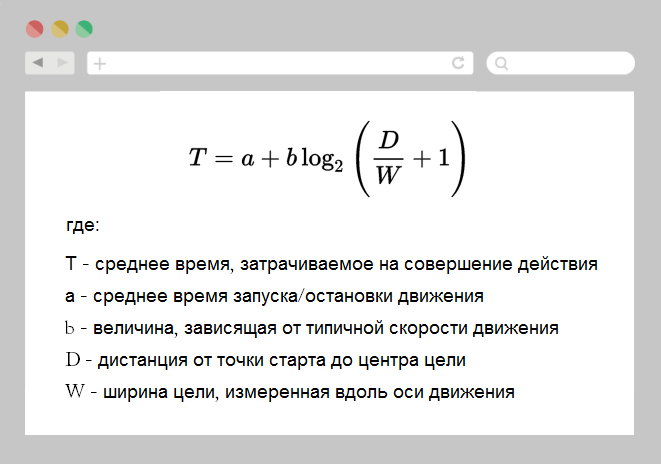
Именно этот принцип и лег в основу закона. И хотя с момента его открытия прошло уже более 60 лет, сегодня закон Фиттса активно применяется в проектировании и дизайне интерфейсов. Об этом мы сейчас и поговорим.

**Суть эксперимента**



На горизонтальной плоскости расположены две цели-полоски одинаковой ширины. Человек по очереди касается каждой полоски. Задача — попасть в каждую цель как можно быстрее.

Меняя ширину целей и расстояние между ними, Пол Фиттс замерял время, которое требовалось для точного попадания в цели. Он выявил закономерность, которую, в свою очередь, описал математической формулой.



*Формула и расшифровка по материалам Википедии.*

В упрощенном виде формулу можно объяснить так: общее время достижения цели зависит от скорости движения и ширины цели вдоль оси движения.

Коэффициенты*a*и *b* при этом будут постоянны в зависимости от индивидуальных особенностей человека и ситуации, в которой происходит взаимодействие с целью.

Обратите внимание: Фиттс исследовал однонаправленные движения, которые совершались вдоль одной оси. Именно по этой причине высота цели значения не имеет при перемещении между ними по горизонтальной оси.

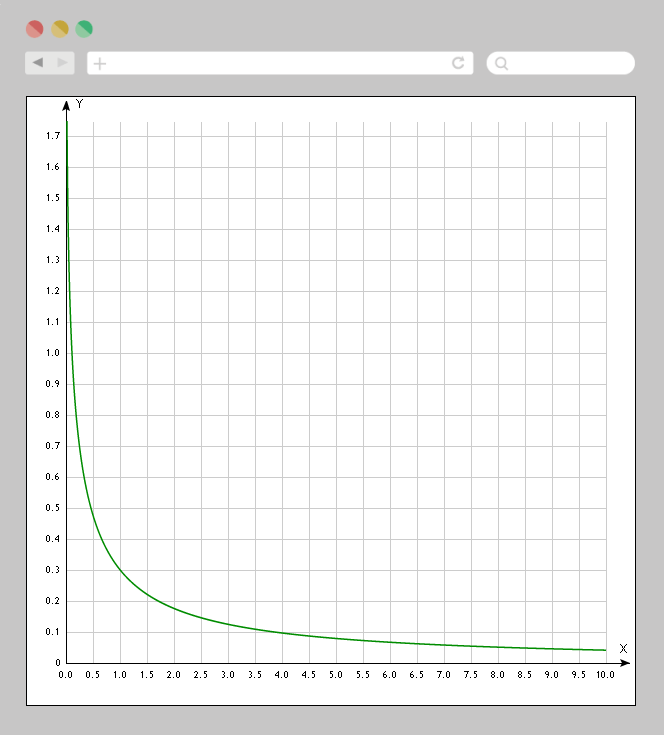
Сам процесс достижения цели при этом можно разделить на несколько этапов:

1. Локализация — время описывается коэффициентом a.
2. Движение — непосредственно перемещение к цели.
3. Торможение — снижение скорости движения перед достижением цели.
4. Достижение — касание.

**Что все это значит**

1. Чем больше цель — тем проще и быстрее можно ее достигнуть.
2. Даже небольшое увеличение маленькой цели существенно увеличить скорость ее достижения.
3. Но после определенного этапа увеличение цели практически не отражается на скорости ее достижения.

Объясняются эти выводы логарифмической зависимостью между переменными. Давайте этот момент немного поясним. Вот график нашей функции, который описывает зависимость между размером цели и временем ее достижения при неизменной дистанции:



Ось Х — это размер нашей цели. Ось У — время.

Мы видим, что до определенного момента даже небольшое увеличение размера дает существенное сокращение времени. Например, при увеличении размера с 0,5 до 1 время сокращается с 0,45 до 0,3, т.е. примерно на 30%.

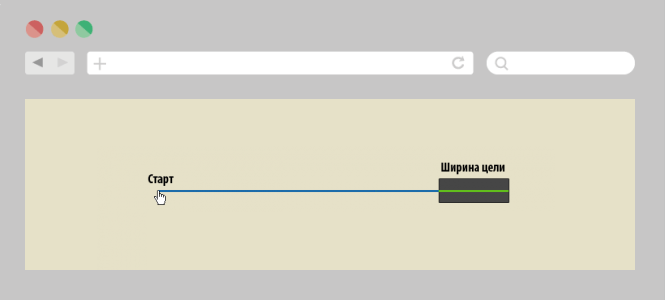
Но чем больше мы увеличиваем размер, тем незначительнее сокращается время достижения цели. Обратите внимание, при увеличении размера с 8 до 10 время достижения уже почти не меняется.

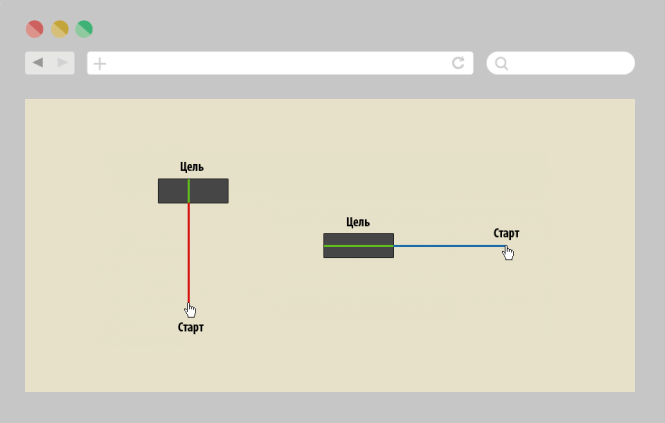
**Как закон Фиттса применяется в дизайне**

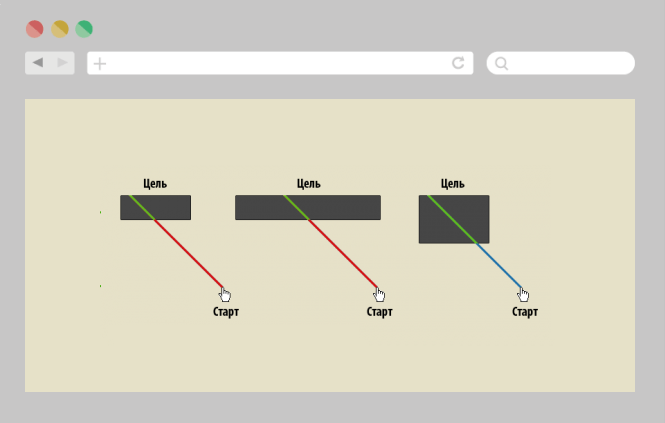
При создании дизайна и проектировании интерфейса закон Фиттса помогает определить оптимальные размеры и взаимное расположение кнопок, ссылок и других элементов на странице в зависимости от того, чего именно мы хотим добиться от пользователя.

Также закон позволяет задать оптимальные для пользователя зоны нажатия для всех кликабельных элементов интерфейса.

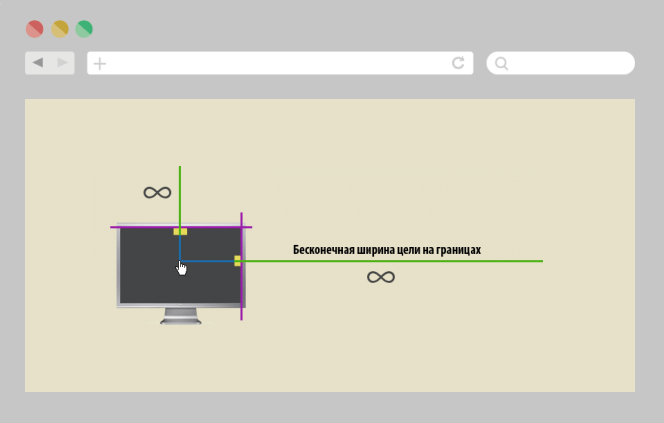
Но здесь нужно учитывать важную особенность: Фиттс проводил исследования для движения строго слева направо. Стартовая точка при этом находится над одной из целей. В реальных же условиях стартовая точка может быть где угодно, а оптимальная ширина в одном измерении может быть не совсем оптимальной в другом.

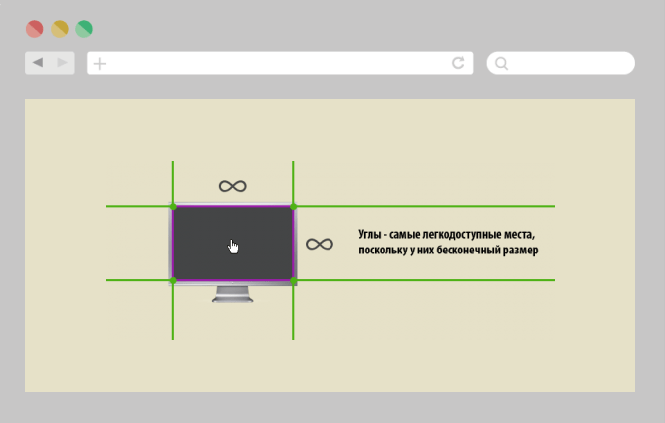






Еще одна особенность, характерная для применения закона Фиттса в веб-разработке и в разработке приложений, — это появлений зон бесконечной ширины.



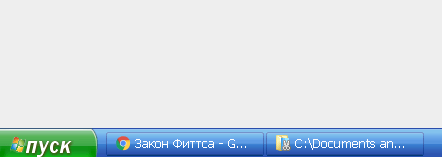


В силу того, что экран ограничен по размерам, элементы, которые находятся на границах имеют условно бесконечную ширину, потому что курсор при попытке в них попасть просто упрется в край экрана и нам не нужно дополнительное время на то, чтобы попасть в элемент по ширине.

Углы в вебе бесконечны — мы можем моментально передвинуть курсор в любой угол, не обращая внимания на реальные размеры элемента, и все равно в него попадем.

**Примеры**

Нижняя панель в старых версиях Windows, в нашем случае это XP, спроектирована в соответствии с законом Фиттса.

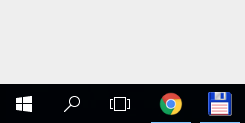


Кнопка «Пуск» расположена в нижнем левом углу экрана — это бесконечная зона, в которую легко попасть за минимальное время с любой стартовой позиции.

Вкладки приложений при этом по ширине практически совпадают с самой нижней панелью, что делает попадание в них также достаточно простым практически с любой позиции.

Обратите внимание, ширины вкладок достаточно, чтобы на них помещался логотип приложения и часть заголовка. Это сокращает время на определение направления движения и локализацию цели.

В новой версии, Windows 10, бесконечный угол остался, но вкладки стали не такими широкими.

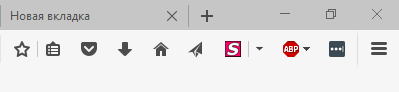


Если мы перейдем в правый верхний угол, можно увидеть еще один интересный пример воплощения закона.

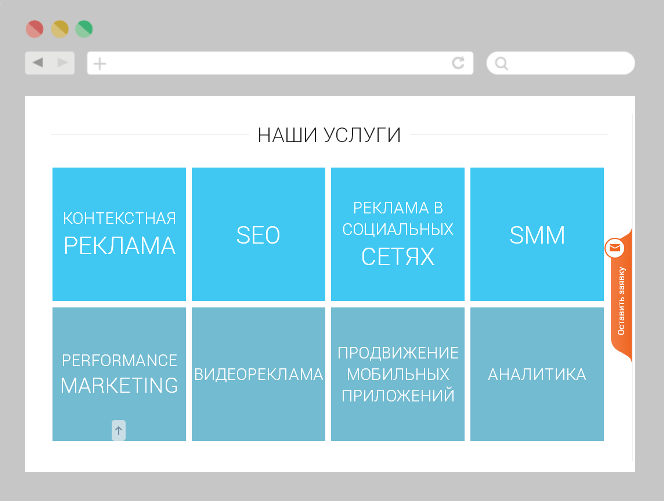


Хотя кнопка закрытия браузера достаточно маленькая и на несколько пикселей отстоит от края экрана, зона нажатия расширена до угла. Курсор точно так же легко, как и с «Пуском» можно, практически не целясь, поместить в верхний правый угол и закрыть приложение.

В новой версии операционной системы зона нажатия совпадает с границами элемента. Но при этом угол по-прежнему остается бесконечным.

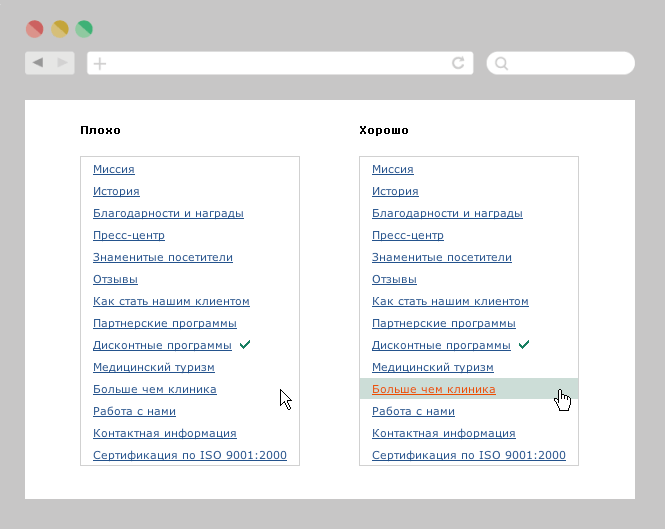


Пример использования бесконечной ширины по бокам экрана мы можем наблюдать на сайте практически любой компании.



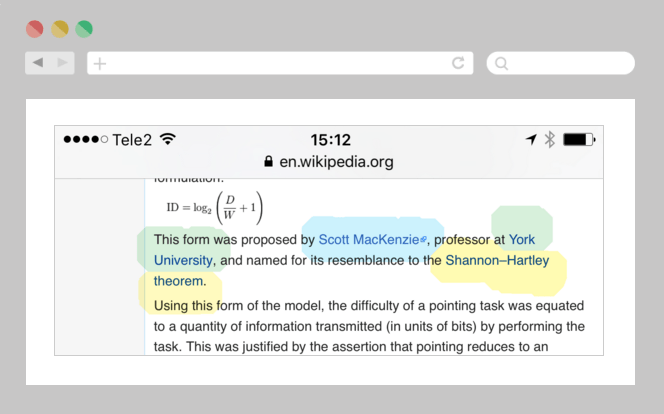
Вкладка «Оставить заявку», онлайн чат, иногда кнопки социальных сетей — размещение этих элементов по краям экрана позволяет сократить время, необходимое для достижения цели.

Илья Бирман [приводит примеры использования закона Фиттса](https://askusers.ru/blog/pravila/zakon-fittsa-v-proektirovanii-interfeysov/" \t "_blank) при проектировании зон нажатия в меню.



В плохом варианте кликнуть можно только по ссылке, в правильном — по всей ширине элемента вокруг ссылки. За счет расширения зоны кликабельности уменьшается время на достижение цели.

Еще один интересный пример — расширение зоны кликабельности для контекстных ссылок при просмотре сайта со смартфона.



Попасть в нужную ссылку становится намного проще, чем при использовании стандартного сценария, который предполагает нажатие только на текст ссылки.

**Выводы и рекомендации**

* Закон Фиттса одинаково применим при проектировании как веб, так и мобильных интерфейсов.
* Также закон показывает одинаковые результаты вне зависимости от того, курсором или пальцем производится движение в направлении цели.
* Закон помогает определить оптимальные размеры кнопок и других элементов, продумать наиболее удобные для пользователя варианты их размещения в интерфейсе.
* Грамотное использование бесконечных углов и зон бесконечной ширины по краю позволяет увеличивать кликабельность элементов без увеличения их фактической видимой ширины.

**2. Порядок выполнения работы**

**Задание**

1. Согласно описанным методам произвести оценку быстродействия и эффективности разработанного интерфейса приложения: определить сколько времени затратит пользователь для достижения указанной в ТЗ функциональности, определить сколько лишней информации содержится в интерфейсе.
2. На основе полученных данных определить насколько эффективной является выбранная модель интерфейса. Придумать способы увеличения эффективности, если это возможно.
3. На основе проведенных исследований принять решение о выпуске программного продукта или отправке его на доработку. Обосновать свой выбор.
4. Подготовить отчет в свободной форме. При формировании отчета использовать полученные на занятии знания.

**3. Содержание отчета**

1. Титульный лист

2. Тема и цель работы

3. Выполнение задания

4. Ответы на контрольные вопросы

5. Выводы

**Контрольные вопросы**

1. В каком году был озвучен закон Фитса?
2. Дайте понятие закона Фитса.
3. Что утверждает закон Фитса?
4. Привести пример использования закона Фитса.

1. Это простейшая формулировка, относящаяся к двумерным целям и перемещению к ним по прямой [↑](#footnote-ref-0)