Лабораторная работа #8
Новикова Дениса из группы 201-363
По теме «Анализ интерфейса»

Задание:

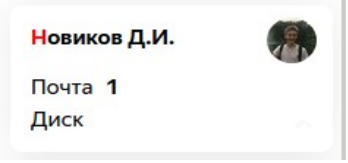
Возьмите любые существующие информационные системы (платформа не важна, анализируемых систем может быть несколько). Продемонстрируйте на основе их:

1. Не менее четырех применений правил теории близости. Чем больше правил вы продемонстрируете – тем лучше. Объясните, почему вы считаете, что правила теории близости в этих примерах применены верно.
2. Не менее двух нарушений правил теории близости. Объясните, почему вы считаете, что правила теории близости в этих примерах нарушены. Сделайте прототипы, которые покажут, как можно исправить эти нарушения. Объясните, почему вы считаете, что в ваших прототипах правила теории близости применены верно.
3. Не менее одного примера модального интерфейса. Объясните, почему вы считаете, что данный интерфейс является модальным. Предложите свой способ обхода и сделайте прототип, который покажет, как эту ситуацию можно исправить/обойти.
4. Не мене трех примеров разных типов обратной связи.
5. Не менее двух примеров нарушений следствий из закона Фиттса. Сделайте прототипы, которые покажут, как можно исправить эти нарушения. Объясните, почему вы считаете, что в ваших прототипах нарушения исправлены.

1. Четыре применения теории близости

Теория близости заключается в том, что элементы, находящиеся рядом друг с другом, воспринимаются как связанные по смыслу. Из этой теории следует совет: необходимо располагать связанные по смыслу элементы рядом, а также группировать связанные элементы. В дополнение идёт утверждение, что схожие предметы сложно различать. Из него следует нехитрый вывод: отличные по смыслу элементы должны различаться и графически. Особые элементы должны выделяться особенно.

Пример 1. Яндекс почта. На рисунке справа мы видим, что заголовок, представляющий имя и инициалы пользователя «Новиков Д.И.» отделён от перечня доступных сервисов (почта и диск) небольшим расстоянием, благодаря чему подчёркивается однотипность этих сервисов, а также разграничивается заголовок от перечня, показывая отличие их природы.

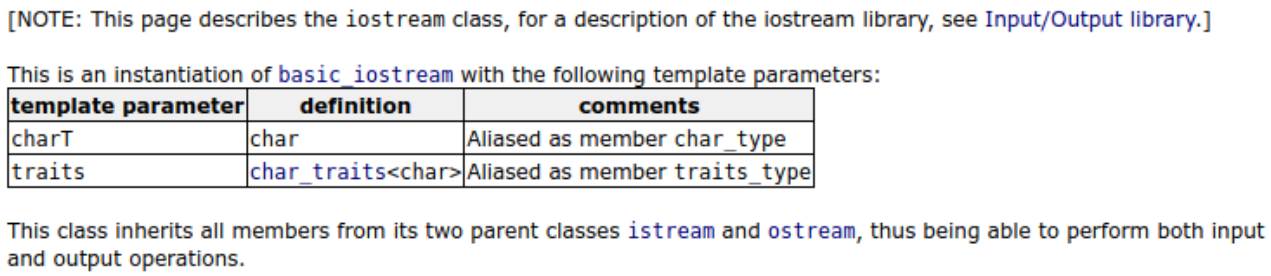


Пример 2. Страница работы с World-art. Обратим своё внимание на рисунок 1: заголовок (Берсерк [ТВ-1]) отделён от таблицы характеристик аниме (жанр, страна производства,

количество серий и т.д.) большим пробелом, а те в свою очередь отделены пустым пространством и в довесок горизонтальной чертой от характеристик другого рода (средний балл, человек проголосовало, место в рейтинге). Благодаря этому разделению образуется три разных группы объектов: заголовок, характеристика самого аниме и реакция пользователей сайта.

Рисунок 1 — Берсерк!!!

Пример 3. Описание класса в cplusplus.com. Посмотрев на рисунок 2 мы сразу увидим три отдельные группы, вторая из которых представляет собой таблицу с небольшой подписью сверху. Благодаря пустому пространству между абзацом сверху и снизу от таблицы, легко понять, какой текст непосредственно относится к таблице.


[NOTE: This page describes the iostream class, for a description of the iostream library, see Input/Output library.]

This is an instantiation of basic_iostream with the following template parameters:

template parameter	definition	comments
charT	char	Aliased as member char_type
traits	char_traits<char>	Aliased as member traits_type

This class inherits all members from its two parent classes istream and ostream, thus being able to perform both input and output operations.

Рисунок 2 — cplusplus.com

Пример 4. Список новостей на Хабре. На рисунке 3 можно увидеть список новостей, в котором каждая новость представлена выделенным полужирным шрифтом заголовком и небольшой подписью внизу, показывающей время публикации новости и количество комментариев. Заголовок и подпись находятся очень близко друг от друга и поэтому воспринимаются как одна группа, тогда как разные новости отделены приличным отступом, из-за чего легко понять, что они представляют из себя разные объекты.

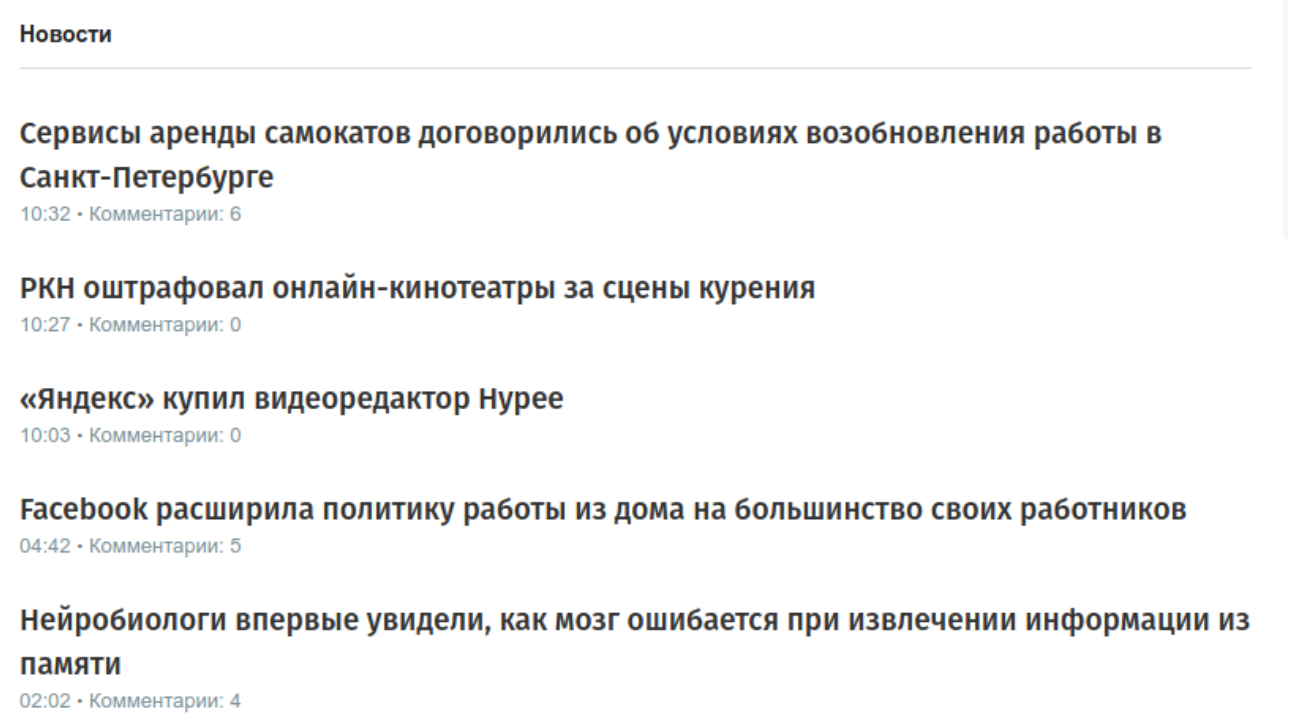


Рисунок 3 — Список новостей на Хабре

2. Два нарушения теории близости

Пример 1. Вход на RuTracker. Обратите внимание на рисунок 4: не кажется ли вам странным, что ссылка «Забыли имя или пароль?» отделена от ссылок «Вход» и «Регистрация» строкой поиска, ведь они должны образовывать одну группу? Здесь явно нарушен принцип теории близости, который может быть исправлен простым переупорядочиванием элементом и добавление небольшого пустого пространства, чтобы отделить две группы; элементы должны располагаться следующим образом: Строка поиска, кнопка «Поиск», пробел, «Регистрация», «Вход», «Забыли имя или пароль?».

Рисунок 4 — RuTracker

Пример 2. Opennet. На сайте opennet, где представлена в огромном количестве документация разнообразных инструментов, можно найти очень интересный перечень оши-

бок (рисунок 5), на который не очень приятно смотреть и не очень приятно читать, т.к. описание предыдущей ошибки никак не отделено от названия следующей ошибки и поэтому слипается с ним в одно целое. Следует делать небольшой отступ после описания каждой ошибки перед названием следующей, т.е. отделить строку «The other end closed the socket unexpectedly.» от «ETIMEDOUT» и «The other end didn't acknowledge retransmitted data after some time.» от «EAFNOTSUPPORT».

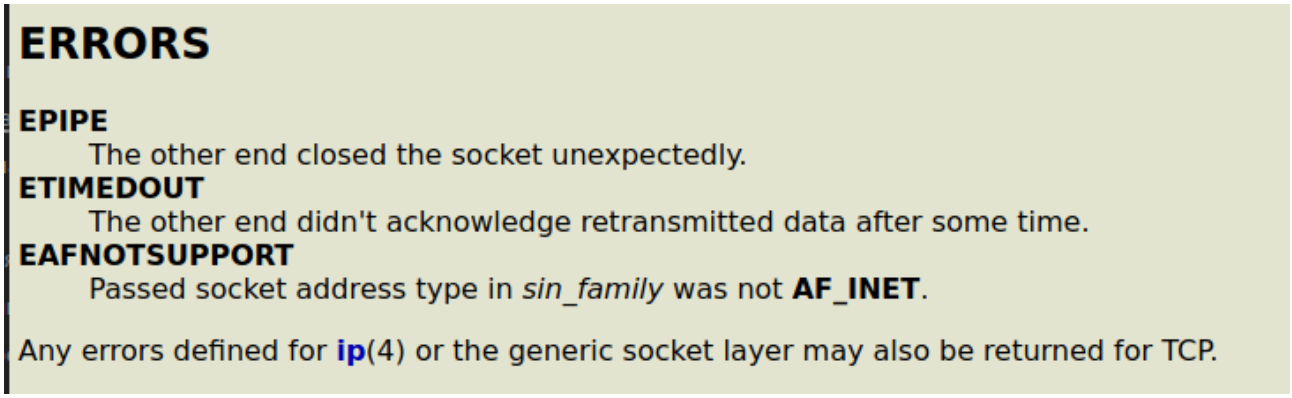


Рисунок 5 — Opennet

3. Модальный интерфейс

Чтобы определить, что такое модальность, сперва нужно разобраться с понятием жест.

Жест — это действие или последовательность действий, которые воспринимаются как одно целое и выполняются (как это воспринимается) одним движением. Например, для опытного пользователя компьютера ввести короткое слово — это один жест, тогда как для человека, только осваивающего клавиатуру, жестом будет каждое отдельное нажатие на клавишу.

Интерфейс называется модальным, если в нём присутствуют состояния, которые не осознаются человеком, но в которых один жест действует по-разному.

Пример модального интерфейса. Одним из самых лучших примеров модального интерфейса, думаю, может послужить Vim — мощнейший текстовый редактор, широко известный и имеющий неоднозначную славу в узких кругах программистов. Модальность в нём заложена в самой его основе; в Vim'е есть несколько режимов: режим вставки, замены, командный (или нормальный) и три визуальных (для выделения текста, строк и блоков текста). Чтобы показать модальность Vim'а достаточно рассмотреть два самых употребительных режима: вставки и командный.

В командном режиме каждая клавиша (не только буквы, но и спецсимволы, цифры, F1...F12; в дополнение идут заглавные буквы, т.е. Shift+Буква, и Ctrl+Буква) имеет своё собственное особенное значение и выполняет определённое действие: клавиша x, например, удаляет символ под курсором, последовательность клавиш gt переключают открытую вкладку; перечислить всё не получится, да и нет необходимости, так как команд в Vim'е больше сотни. В режиме вставки каждая символьная клавиша вставляет в текстовый файл соответствующий символ. В этом заключается модальность Vim'а.

В данном случае модальность позволяет невероятно расширить функционал текстового редактора, сделав использование горячих клавиш нормой, благодаря чему все операции от

элементарных до весьма сложных делаются парой нажатий на клавиши, что с опытом доходит до органического автоматизма, вселяя экзистенциальный ужас всем, кто не знаком с Vim'ом.

Однако в данной лабораторной работе следует предложить способ избавиться от модальности. Поэтому: чтобы избавить от модальности Vim, нужно сделать из него Emacs. Т.е. вынести все горячие клавиши из командного режима в режим вставки, добавив кучу клавиш Ctrl и Shift, отяжелив и убив всю их полезность. Такова цена следования букве закона проектировщиков интерфейсов!

4. Обратная связь

На действия пользователя интерфейс должен отвечать заметным сигналом, по которому можно понять, что действие прошло. Однако обратная связь должна быть адекватна действию и не вводить пользователя в заблуждение: если сообщение не отправлено, не должно быть сигналов, подтверждающих отправление.

Пример 1. Подсветка строки поиска. Firefox подсвечивает строку поиска, когда та получает фокус (рисунок 6).

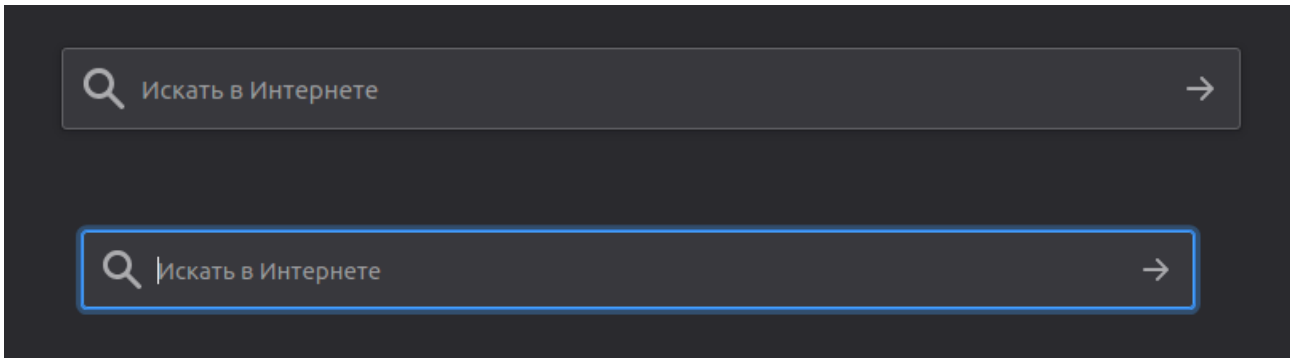


Рисунок 6 — Подсветка строки поиска

Пример 2. Изменение курсора при наведении на ссылку. Firefox меняет курсор, если под ним находится ссылка (русинок 7).

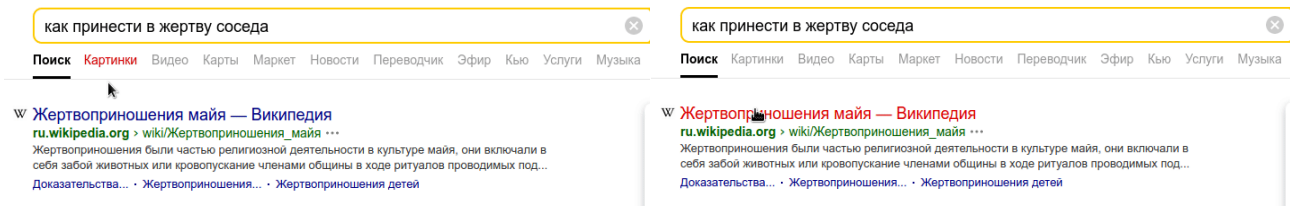


Рисунок 7 — Принесение соседа в жертву

Пример 3. Информирование о доставке и прочтении сообщения. WhatsApp помечает сообщение двумя серыми галочками, когда сообщение доставлено, и двумя синими галочками, когда сообщение прочитано (рисунок 8).

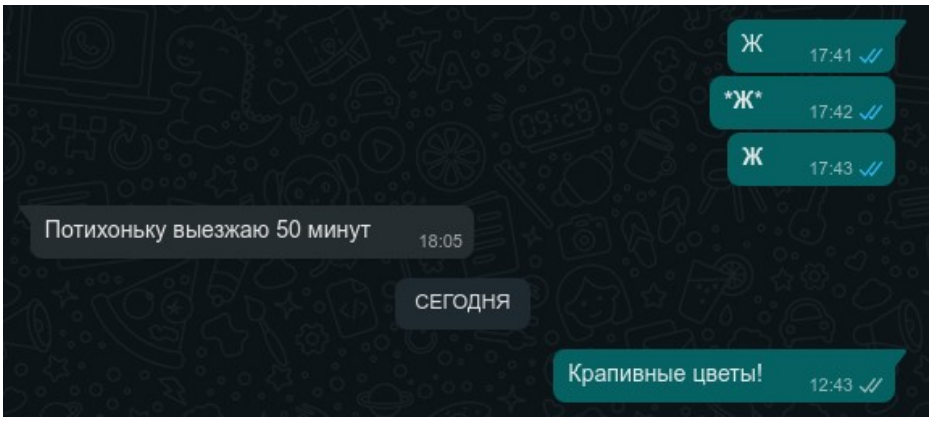


Рисунок 8 — Переписка в WhatsApp

5. Нарушение следствий из закон Фиттса

Основываясь на законе Фиттса (да, похоже, и не только), в отношении проектирования интерфейсов были сделаны следующие выводы:

1. Следует делать интерактивные объекты (любые места на экране, куда можно нажать; например, кнопки); достаточного, но не слишком большого размера. Следует отделять взаимно-нежелательные кнопки, такие как "сохранить" и "удалить";
2. Если невозможно сделать объект достаточно большого размера, то следует увеличивать не его самого, а область нажатию, которую можно выделить с помощью цвета заднего фона;
3. Точка под курсором — самая удобная точка для нажатия; на этом принципе делают всплывающие по нажатию правой кнопки мыши меню (в программах 3Д моделирования иногда делают всплывающие меню, размещающиеся по разным сторонам от курсора);
4. Углы экрана — тоже очень удобные точки нажатия; иногда используются так называемые горячие углы;
5. Необходимо, чтобы один объект (как его будет мыслить пользователь), если интерактивен, при клике по нему в разные места действовал одинаково, иначе возникает большая путаница; интерактивный объект не должен содержать дыр, где интерактивность не работает;
6. Движение мыши к цели и нажатие превращаются один жест, поэтому окна, которые могут всплыть по ходу движения мыши и закрыть цель, — зло;
7. В более удобные места необходимо ставить объекты, которыми часто пользуются.

Пример 1. Неудобное расположение кнопки закрытия. Посмотрите на рисунок 9: можно видеть, что кнопка закрытия браузера отстоит от верхнего правого угла на пару пикселей, из-за чего, если сдвинуть курсор максимально вправо-вверх, по этой кнопке попасть не

получится. Это очень неудобно, т.к. придётся дольше целится в эту кнопку. На рисунке 10 ситуация исправлена: теперь, если сдвинуть курсор в самый угол, нажатие на левую клавишу мыши всё равно нажмёт на кнопку закрытия и действие будет совершенно без лишних усилий.

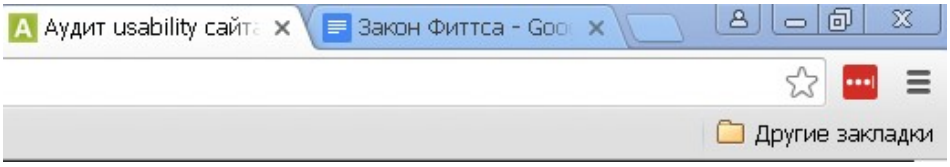


Рисунок 9 — Неудачное расположение кнопки закрытия

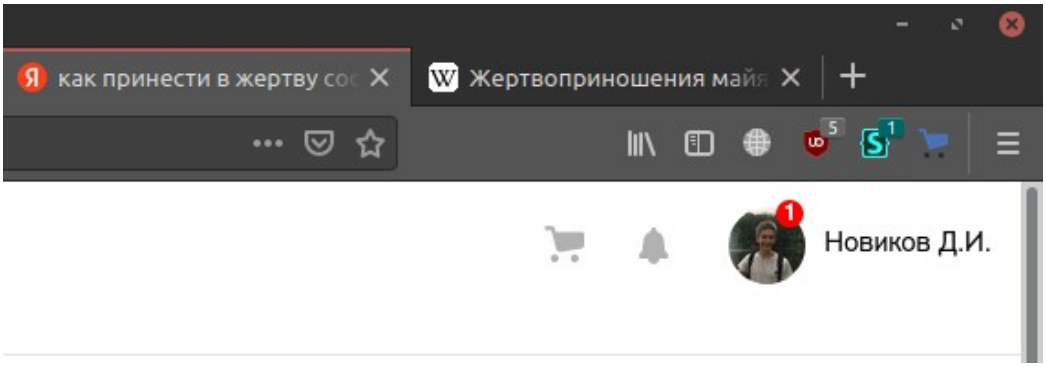


Рисунок 10 — Удачное расположение кнопки закрытия

Пример 2. Элемент списка должен нажиматься в любом месте. Рассмотрим список, представленный на рисунке 11. Его беда в том, что если отвести курсор чуть правее текста, описывающего элемент списка и нажать на левую кнопку мыши, элемент выбран не будет. Однако должен. На рисунке 12 изображена ситуация, исправленная путём расширения области нажатия на элемент.

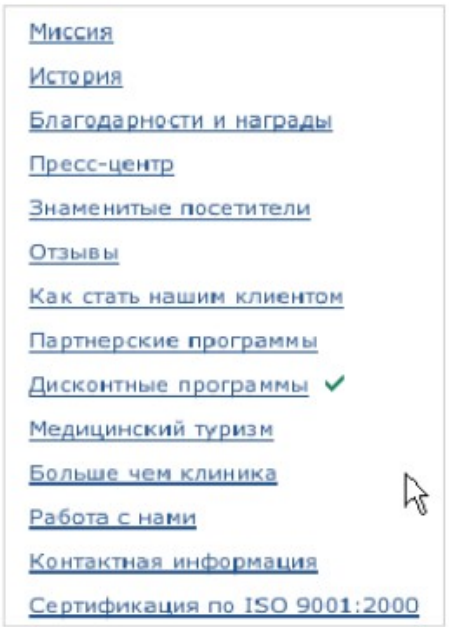


Рисунок 10 — Неправильный список

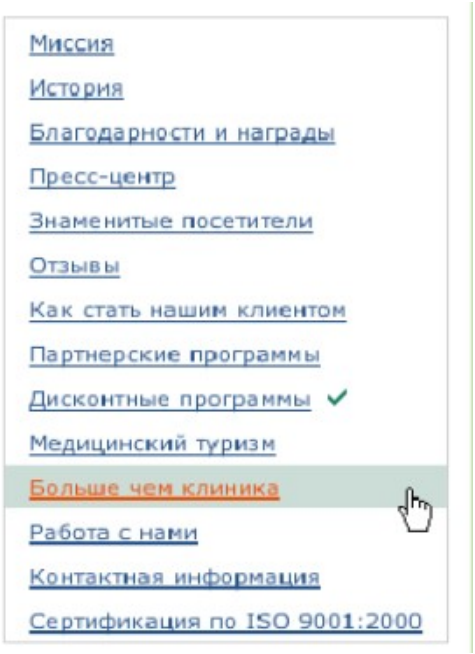


Рисунок 11 — Правильный список

Выводы

При проектировании интерфейса следует придерживаться следующих принимать во внимание следующее:

1. Теорию близости
2. Следует избегать модального интерфейса
3. Необходимо осуществлять обратную связь
4. Следствия из закон Фиттса.

Вышеперечисленные принципы помогут создать интерфейс, который будет наиболее удобен конечному пользователю.