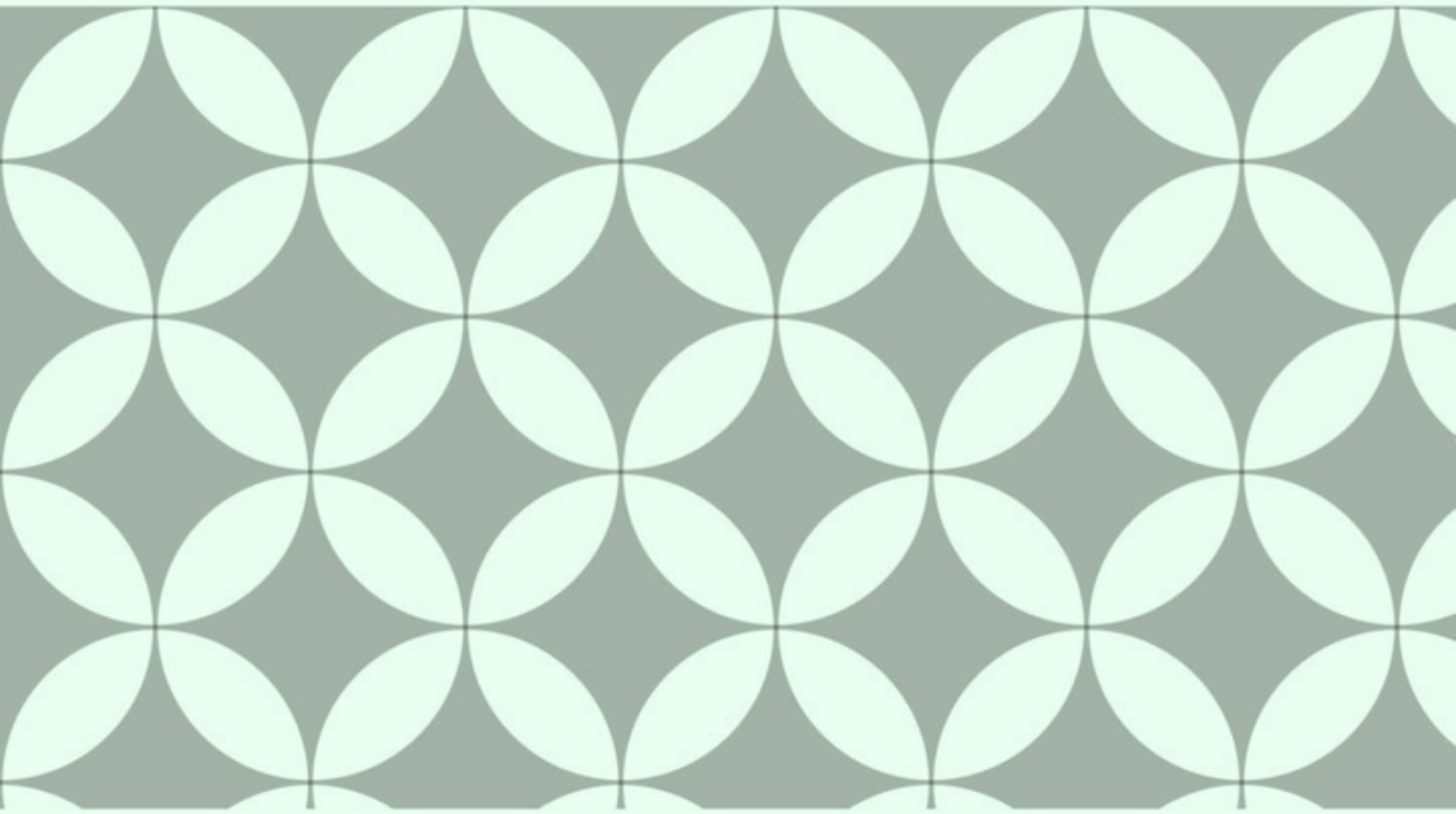
Иван Трещев



ПРОГРАММИРОВАНИЕ ДЛЯ МОБИЛЬНЫХ ПЛАТФОРМ

IOS

Иван Трещев

Программирование для мобильных платформ. IOS

«Издательские решения»

Трещев И.

Программирование для мобильных платформ. IOS / И. Трещев — «Издательские решения»,

ISBN 978-5-44-939973-1

Данная книга обобщает опыт работы лаборатории мобильных приложений на базе ФГБОУ ВО КнАГУ, где автор был ее руководителем. Приложения, разработанные в книге, были успешно выложены в магазин приложений. В книге вы найдете описание основных моментов для разработки приложений.

ISBN 978-5-44-939973-1

Содержание

Введение 6
Основы работы с Xcode 7
Основы графического дизайна интерфейсов приложений 18
Функции устройства 29
Конец ознакомительного фрагмента. 30

Программирование для мобильных платформ IOS

Иван Трещев

Участие в тестировании разработанных приложений Владислав Андреевич Чусов
Участие в работе над иллюстрациями Мария Сергеевна Аксютина
Участие в верстке и корректуре Анастасия Сергеевна Ватолина

ISBN 978-5-4493-9973-1
Создано в интеллектуальной издательской системе Ridero

Введение

Разработка мобильных приложений может приносить и стабильный доход и стать точкой роста для профессионала. Современный рынок мобильных устройств полон различными аппаратами всевозможных форм-факторов, если говорить про IOS то это различные телефоны и планшеты фирмы Apple, умные часы и плееры. Конечно по сравнению с другими платформами (скажем Windows Phone или Android) читатель, для того чтобы начать разрабатывать приложения для проприетарной операционной системы для мобильных устройств Apple, должен по крайней мере приобрести ПЭВМ Mac производства фирмы Apple, поскольку Xcode – среда программирования доступна только для MacOS (что является не дешевым удовольствием) и желательно иметь под рукой для тестирования минимальный набор аппаратных устройств – телефон и планшет, хотя в среде программирования и предусмотрены различные эмуляторы, но опыт показал, что все же желательно тестирование проводить на реальных устройствах. Итого инвестиции в процесс разработки приложений порядка 1500€. Помимо всего вышеперечисленного ежегодно необходимо оплачивать 100 долларов США как сбор для разработчиков (к примеру для платформы Android взнос единоразовый), что осуществить так же не просто, особенно для жителей России, поскольку номер факса для связи с Apple именно из США.

Лаборатория которой руководил автор на протяжении 5 лет занималась разработкой различных приложений для самых популярных за последнее пятилетие операционных систем носимых устройств – Android, IOS, Windows Phone. Сегодня платформа IOS насчитывает многомиллионную аудиторию и располагает одним из самых удобных и эргономичных способов для авторов (будь то песни, книги или приложения) для монетизации своих творений при этом не неся затрат на тиражирование, продажу, экспозицию и другие накладные расходы.

По сравнению с другими платформами IOS выделяют несколько немаловажных аспектов. Во-первых эмуляторы есть для всех устройств (не просто для популярных как Windows Phone и просто эмуляторы Android). Во-вторых это конечно то, что ваше приложение действительно будут тестировать специалисты Apple. Причем тестировать будут весьма притязательно. Не стоит надеяться что только разработав приложение и отправив его на модерацию оно тут же будет размещено в магазине (как в случае с Android). Для этого потребуется от двух рабочих дней. В-третьих все же есть единый стиль, единые стандарты программирования, единообразные интерфейсы и механизмы их создания. Member Center, ICloud, синхронизация со всей инфраструктурой Apple. Этот список довольно велик. Причем благодаря постоянным обновлениям операционной системы и «экосистемы» ежегодно появляются новые технологии компании, которые интегрируются в приложения.

В работе не было попытки дать детальное объяснение всех возможных аспектов программирования под IOS. А скорее данная книга будет полезна тем кто все же открыл Xcode приобрел аккаунт разработчика и планирует начать пробираться через тернии и хитросплетения современных технологий программирования.

Данная книга посвящена разработке приложений именно под платформу от Apple и является второй в цикле, над которыми автор работает в настоящее время.

У читателя предполагается опыт программирования на объектно-ориентированном языке, желательно опыт на C#.

Автор хотел бы выразить огромную благодарность Чусову В. А. и Аксютиной М. С. без которых невозможно было работать.

Основы работы с Xcode

Сегодня мы разберемся, как создать свой проект, познакомимся с основными объектами Фреймворка UIKit.

И так, первым шагом мы запустим Xcode, и создадим проект, после этих действий, должно появиться окно, где можно выбрать уже подготовленный для чего-либо проект, например, выберем «Single View Application» (рис. 1.1). С остальными типами познакомимся позже.

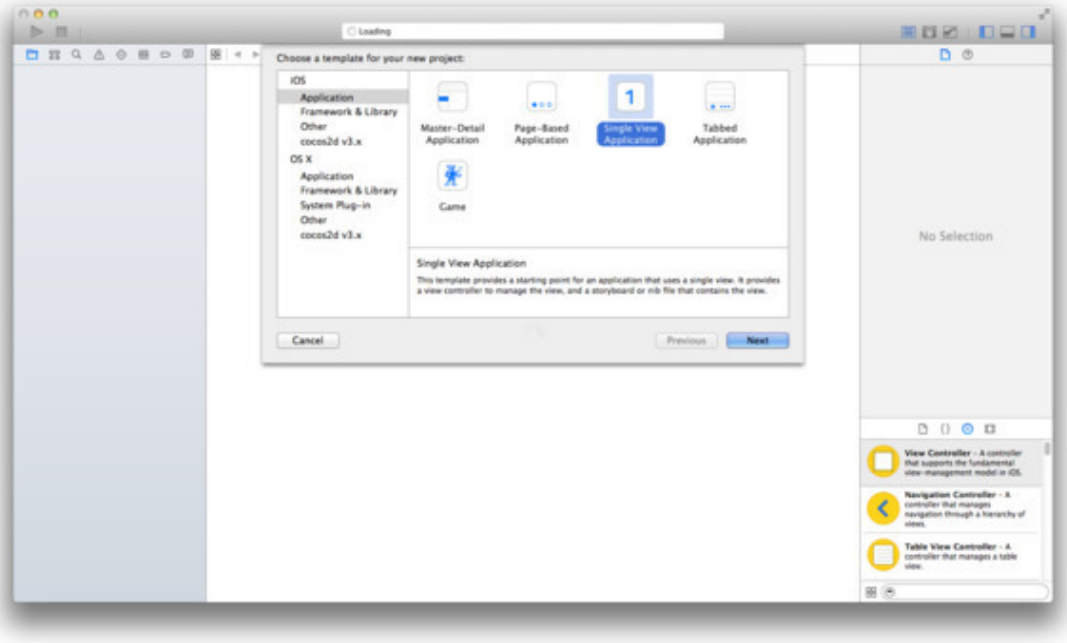


Рисунок 1.1 – Выбор типа приложения

Далее вводим данные, название проекта, название компании, уникальный идентификатор компании, язык на котором мы будем программировать, выбираем Objective-C, далее выбираем для какого типа устройств создаем приложением, и последний пункт – Core data, это некая оболочка/Фреймворк для обработки данных. Пока что, вы можете вводить любые данные, т.к. этот всего лишь тестовый проект.

После проделанных манипуляций, перед нами должно появится окно с настройками нашего проекта (рис. 1.2).

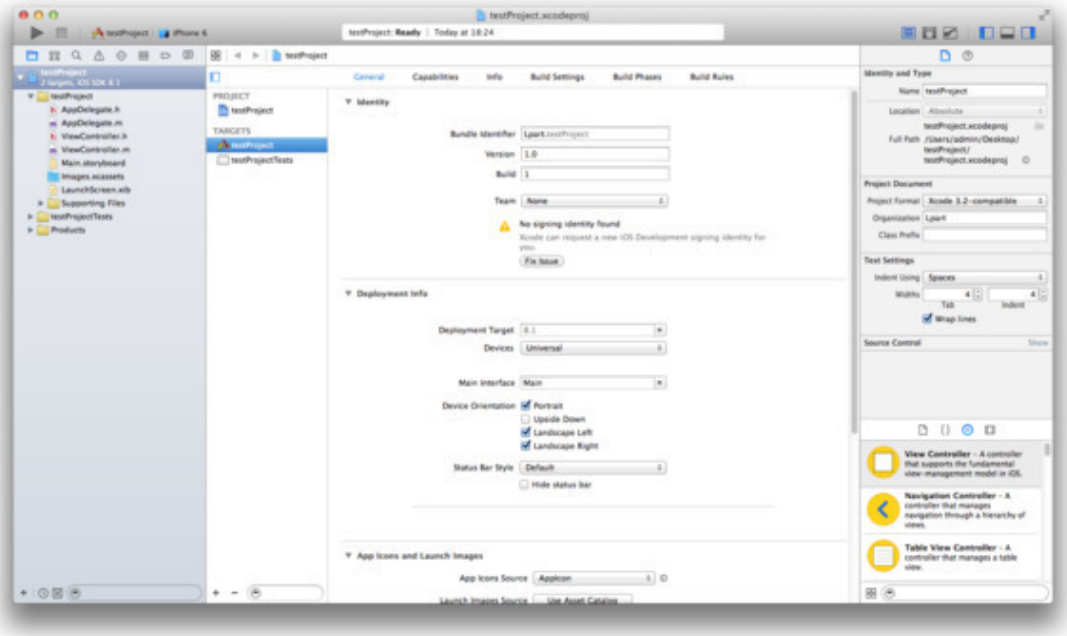


Рисунок 1.2 – Настройки приложения

Разберемся с targets. Грубо говоря, один target – один вид нашего проекта. В каждом target мы можем добавлять или убирать какие-либо объекты, данные. К примеру, мы создаем приложение для двух компаний, но с разными картинками, мы в одном target вставляем одни картинки, а в другом иные картинки. Так же есть еще такое понятие Scheme – это схема, которую мы можем изменять, создавать свою новую или создать автоматическую Она позволяет нам настраивает запуск приложения, его сборку и т. д.

Далее идут основные параметры проекта (рис. 1.3). Identity. В нем предоставляется информация о проекте и команде разработчиков. Deployment info – здесь мы указываем, какие версии iOS будет поддерживать наше приложение, для какого устройства разработка ведется, какие допустимы ориентации устройства. Status Bar – это строка состояние устройства.

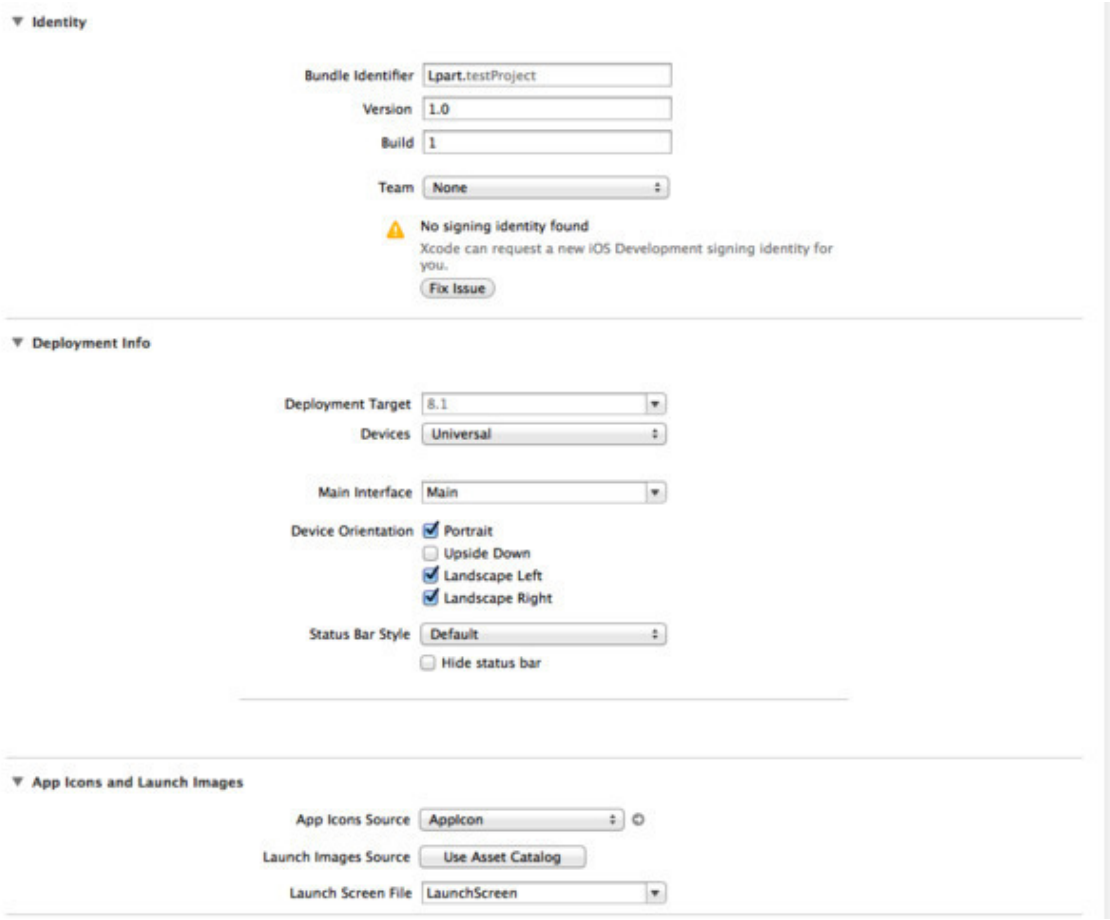


Рисунок 1.3 – Основные параметры проекта

Затем указывается галерея (рис. 1.4) иконок и загрузочных экранов приложения.

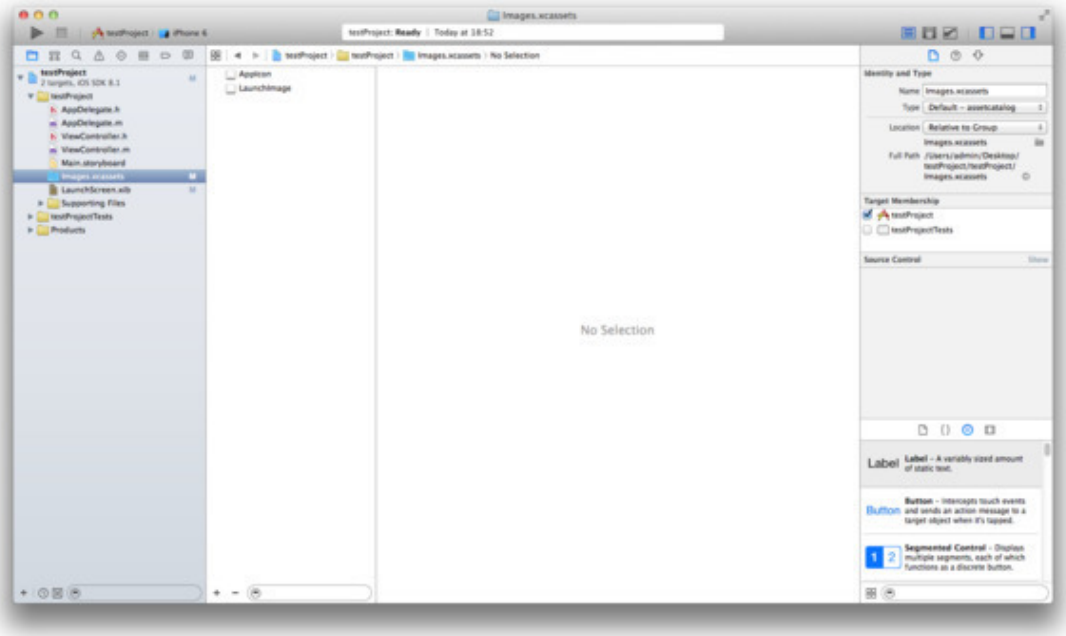


Рисунок 1.4 – Images. xcassets

Загрузочный экран указывается не изображение, а xib-файл (рис. 1.5), в котором у нас находится один объект UIKit – View, попозже разберемся что это такое. Какая разница между xib и storyboard, второе используется в версиях iOS 5 и позднее, xib является устаревшим механизмом разработки интерфейса программы, еще одно отличие – это то, что значительно упростилась работа по созданию интерфейса, различие в представление иерархии объектов, также значительно сократилось кол-во строчек xml-кода.

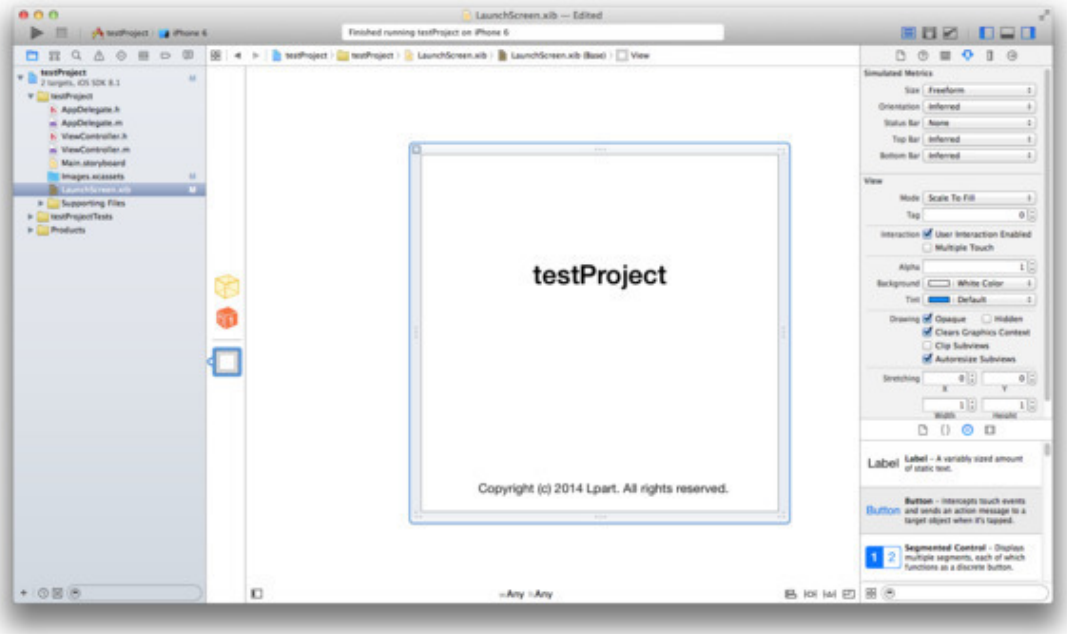


Рисунок 1.5 – Launch screen

Вернемся к настройкам проекта, рассмотрим еще параметр info (рис. 1.6). В нем содержаться основные данные о проекте, которые хранятся в файле "Info.plist», в нем мы можем указывать параметры нашего проекта. Вкладка Capabilities, содержит возможности приложения, такие как: Game Center, iCloud, Maps и т. д. В этой кладке мы подключаем дополнительные возможности. Далее Build Settings – настройка сборки приложения. Build Phases – позволяет нам настраивать сборку приложения для текущего target. Build Rules – позволяют нам задавать правила сборки приложения, к примеру, в момент сборки, сжимать текстовые файлы с помощью определенных скриптов.

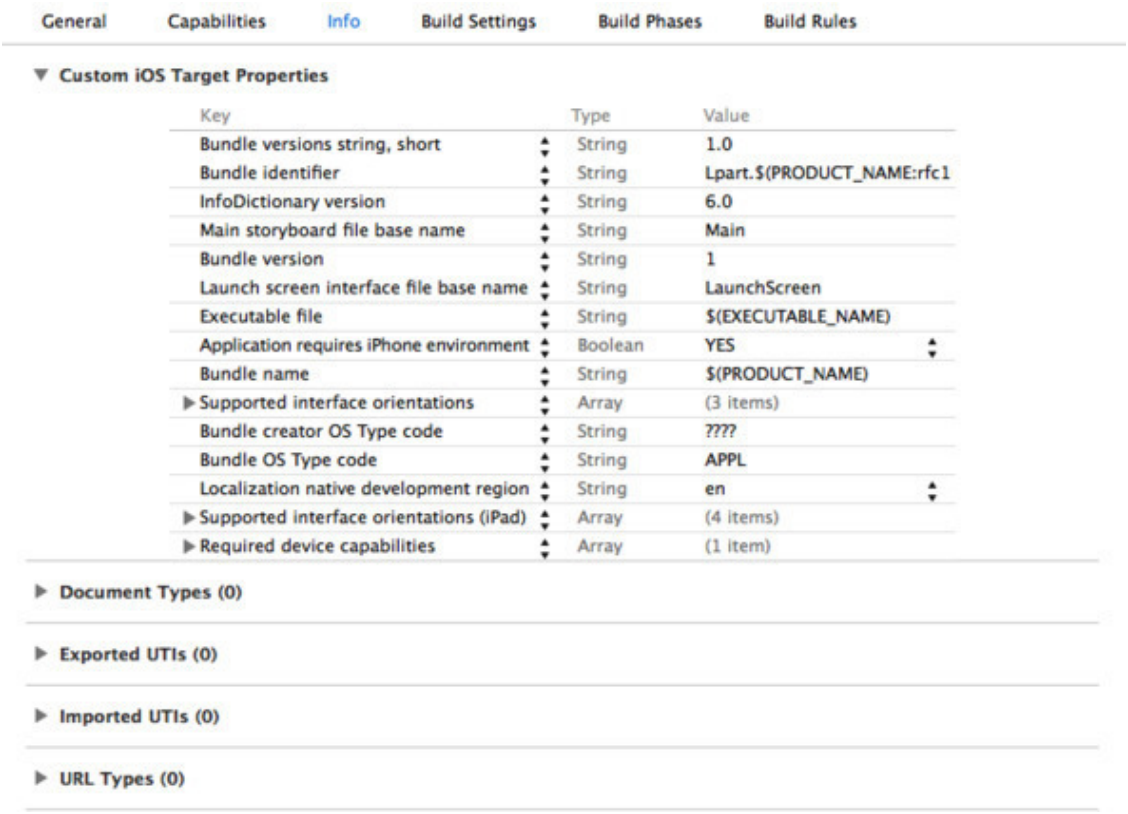


Рисунок 1.6 – Информация о проекте

Теперь перейдем в Storyboard. Как уже упоминалось ранее, это есть механизм разработки интерфейса приложения, также мы можем создавать интерфейса приложения в коде.

Перед нами должен появится View Controller (рис. 1.7) – это фундаментальный объект в UIKit, на котором можно отображать различные объекты UIKit. Сам View Controller добавлен на объект UIWindow, который поддерживает отображение графических элементов на экран. Как вы можете заметить у вью есть три элемента. Первый элемент – означает, что выбран сам вью. Второй элемент – открывает нам список готовых действий. Третий элемент – высвобождает из стека предыдущий вью, тем самым возвращает предыдущий «экран».

Рисунок 1.7 – Storyboard

Рассмотрим элементы Storyboard, в нижней части экрана (рис. 1.8) указываются размеры экранов, с которыми можно работать. Начиная с 3.5-дюймового телефона в портретной ориентации, и заканчивая 12.9-дюймовым планшетом в любой ориентации.

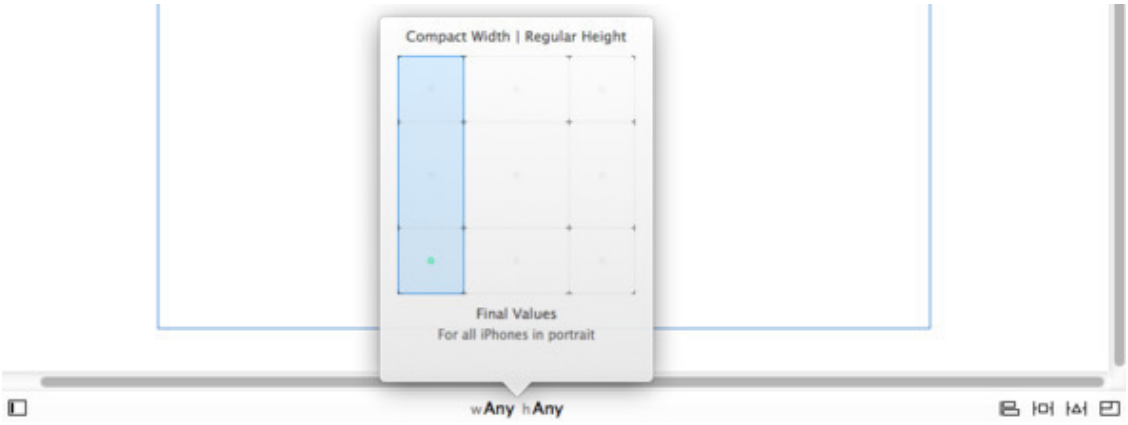


Рисунок 1.8 – Настройка представления

В левой нижней части есть кнопка, которая открывает нам иерархию объектов на View Controller, в правой нижней части, начиная от середины, первая кнопка позволяет выравнивать объекты на представление. Вторая кнопка позволяет нам закреплять объекты на определенной позиции экрана. То, что мы добавляем выравнивание или закрепляем объект – называется добавлением constraint. Третья кнопка обновляет constraint, либо изменяет положение объектов в соответствие с их constraints. Последняя кнопка обновляет constraints, если изменяются размеры экрана.

Теперь рассмотрим правый блок Xcode (рис. 1.9).

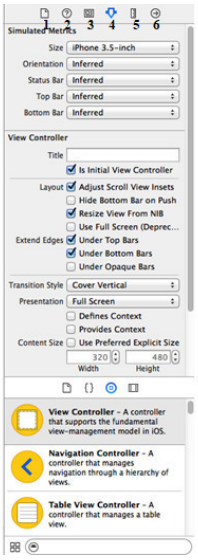


Рисунок 1.9 – Утилиты

Этот блок содержит сведения и параметры, выделенного объекта, снизу, находится библиотека объектов, список различных конструкций кода, файлов, добавляемых в проект, и медиа библиотека.

Разберем пронумерованные вкладки (рис 1.9). Первая – файловый инспектор, в котором указаны характеристики storyboard, такие как: где расположен файл, под какую версию разрабатывается и т. д. Вторая – быстрый помощник. Третья – идентификатор выбранного объекта, в нем задается идентификатор объекту, указывается класс. Четвертая вкладка – параметры объекта. В пятой вкладке, настраиваются размеры объекта. Шестая – показывается существующие связи между объектами.

Рассмотри навигацию проекта (рис. 1.10). Первая вкладка – файловый инспектор проекта. Вторая – показывает нам классы, а при их раскрытие, показывает методы классов. Третья – поисковая строка, ищет во всем проекте, также можно искать и в самом классе (cmd+f). Четвертая – показывает список предупреждений и ошибок, если таковые имеются. Пятая – список существующих тестов для приложения. Шестая вкладка – показывает нагрузку на процессор, память, сеть, файловую систему телефона. Седьмая – список точек остановок. Восьмая – показывается выполненные действия над проектом.

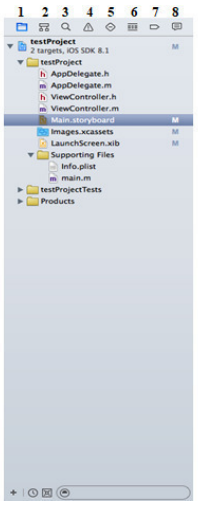


Рисунок 1.10 – Навигация проекта

Теперь добавим кнопку в наш проект. Для этого, нам необходимо найти библиотеку объектов, и в ней найти Button (рис 1.11).

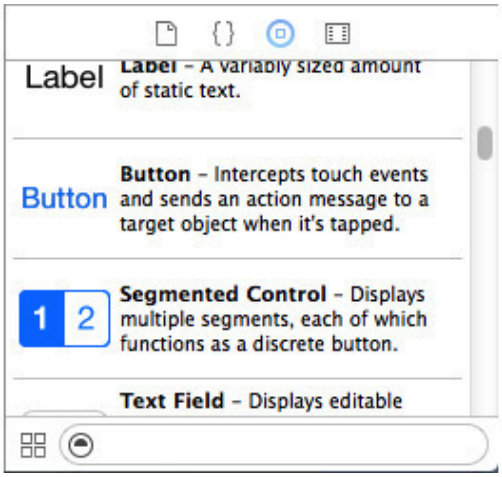


Рисунок 1.11 – Button

Затем, с помощью метода «drag and drop», добавляем на наш View Controller. Затем таким же способом добавим Label. Обоям объектам установить размер 200x40, в инспекторе размера. А нашему вью в инспекторе параметров, установите для size значение – iPhone 3.5 inch.

Теперь мы должны установить связи, между кнопкой и вью, и между надписью и вью. Нам нужно разделить экран Xcode на две части, нажмем на соответствующую кнопку в правом верхнем углу (рис. 1.12).

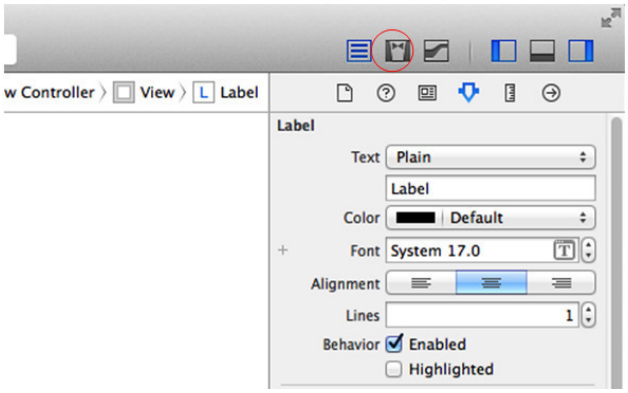


Рисунок 1.12 – Assistant Editor

Теперь у нас есть два окна, в левой части storyboard, в правой части какой-либо класс. Теперь в правую часть поместим ViewController. h, из файлового инспектора перетянем этот класс в правое окно (рис. 1.13).

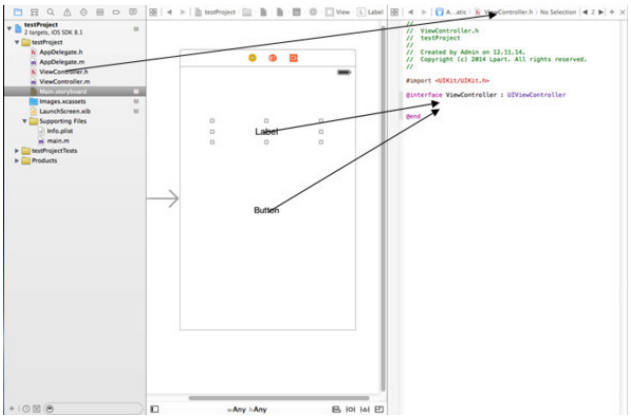


Рисунок 1.13 – Assistant Editor II

И добавим связи в представление вью. Правой кнопкой мыши перетяните стрелку с Label и Button в ViewController. h. Перед вами появится окно с параметрами (рис. 1.14), с ними мы разберемся позже, а пока что введите имя надписи.

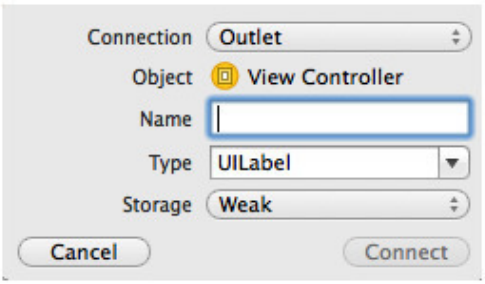


Рисунок 1.14 – Параметры свойства

Для Button нужно будет дважды проделать такое действие, первое мы добавляем свойство для кнопки, а второе мы добавляем для него метод, чтобы добавить метод, нужно указать в параметре connection – action (рис 1.15).

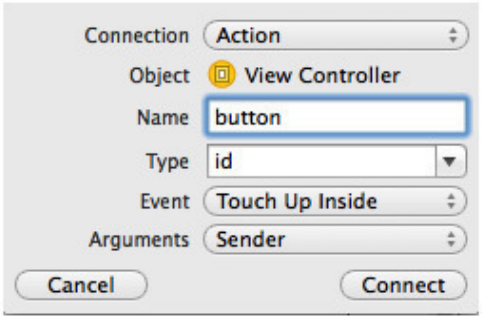


Рисунок 1.15 – Action button

Должно получится следующим образом (рис. 1.16).

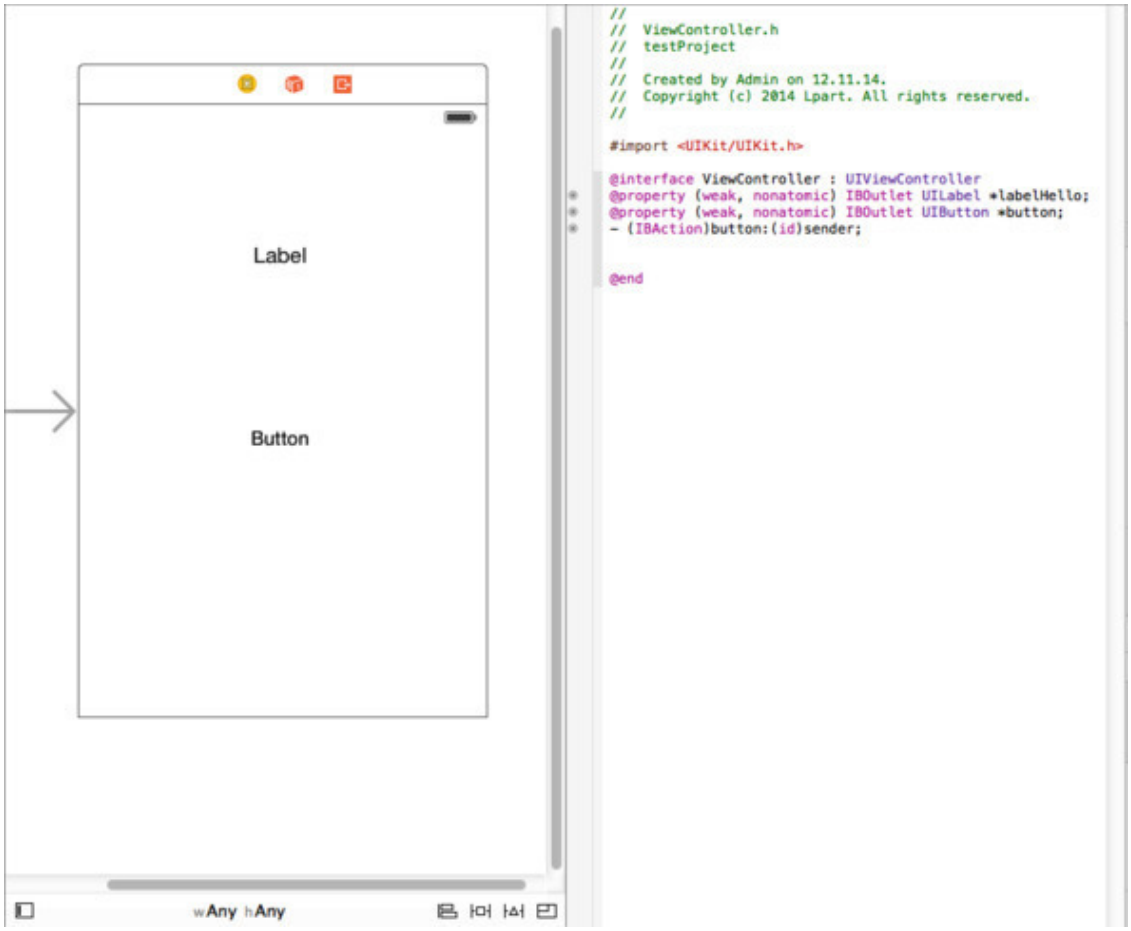


Рисунок 1.16 – Связь объектов и вью

Теперь закроем второе окно и перейдем в класс ViewController. m. Найдем в нем метод – (IBAction) button: (id) sender. Это метод кнопки, который вызывается каждый раз, когда мы нажимаем на кнопку. Пропишем в этом методе следующий код:

– (IBAction) button: (id) sender {
_labelHello. text = @«Hello World!»;
}

Теперь, когда будете нажимать на кнопку, надпись будет менять свой текст на «Hello World», результат работы – рисунок 1.17.

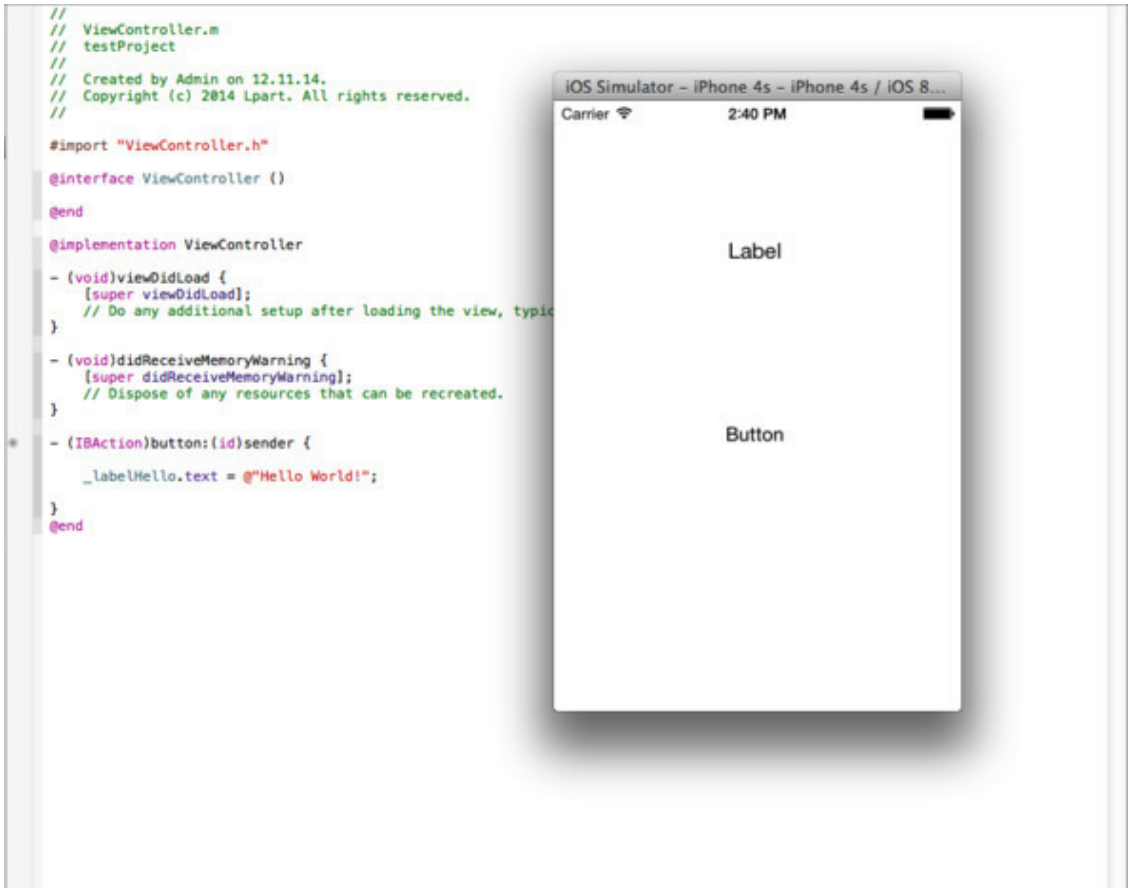


Рисунок 1.17 – Результат работы

И последнее, что осталось – отладчик. Чтобы поставить точку остановки, необходимо нажать в выделенной полосе слева от кода (рис. 1.18).

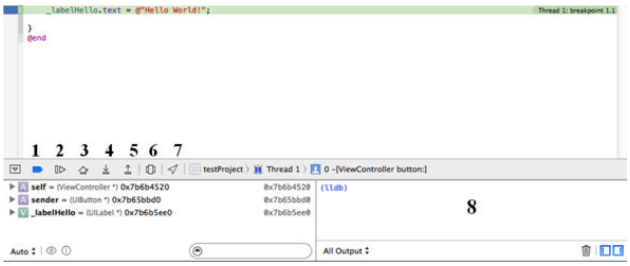


Рисунок 1.18 – Отладчик

Первая кнопка – отключает/включает точки остановки. Вторая – продолжить/приостановить процесс работы приложения. Третья – пошаговая отладка кода. Четвертая кнопка – заход в метод. Пятая – выход из метода. Шестая – позволяет нам просматривать наше приложение в виде 3D-модели. Седьмая кнопка – задает местонахождение. Окно под номером 8 – консоль, в которой выводятся системные сообщения.

Основы графического дизайна интерфейсов приложений

«Быть дизайнером – значит не просто собирать разрозненные элементы воедино, упорядочивать их или как-то изменять. Тут нужно и создавать некую ценность, и придавать смысл, и освещать, и упрощать, и трансформировать, и облагораживать, и сгущать краски, и убеждать, и даже в какой-то мере развлекать».

– Пол Рэнд (Paul Rand)

Перед тем как начать разрабатывать интерфейс приложения, следует разобраться с его основными задачами. Грамотно созданная графическая составляющая приложения должна способствовать эффективной работе, облегчать взаимодействие пользователя с ПО. Исходя из этого, сформулируем 20 основных тезисов, которые должен знать каждый дизайнер:

1.Обязанность интерфейса – обеспечение взаимодействия.

Интерфейсы служат для обеспечения взаимодействия между людьми и окружающим миром. Они помогают нам прояснять, освещать, реализовывать и наблюдать взаимосвязи; они могут объединять и разъединять нас, влиять на наши ожидания; а кроме того, они дают нам доступ к различным услугам. Интерфейсы призваны выполнять определенные функции.

2. Ясность прежде всего.

Любой интерфейс в первую очередь должен быть понятным. Чтобы эффективно использовать разработанный вами интерфейс, люди должны понимать, что он из себя представляет, зачем им его использовать, какие задачи они смогут с его помощью выполнять, к чему приведет то или иное действие – и тогда они смогут успешно с ним взаимодействовать. Да, разобраться в интерфейсе с первого раза может быть непросто, но двусмысленностям в нем места нет. Понятному интерфейсу пользователи доверяют и поэтому, скорее всего, будут использовать его в дальнейшем. В общем, вместо того чтобы запихивать все на один экран и тем самым запутать пользователей, лучше сделайте сотню понятных экранов.

3. Внимание любой ценой.

Внимание пользователей бесценно. Дайте ему сосредоточиться, не засоряйте приложения элементами, отвлекающими внимание от главного назначения того или иного созданного вами экрана. Если пользователям нужно что-то прочитать, то дайте им достаточно времени для этого, а уж потом показывайте рекламу, если это необходимо. Цените внимание ваших пользователей: от этого в выигрыше будут и они, и вы. Для обеспечения использования продукта внимание пользователей – жизненно важный фактор. Старайтесь удерживать его любой ценой.

4. Под контролем пользователей.

Людям нравится чувствовать контроль над ситуацией. Многие разработчики об этом не задумываются, и в результате пользователи вопреки своему желанию вынуждены совершать операции, которые они не собирались совершать, причем направление их движения оказывается весьма неясным, а результаты действий – неожиданными. Дайте пользователям почувствовать, что ситуация под их контролем, периодически отображая состояние системы, описывая причинно-следственные связи (если вы сделаете это, случится то-то) и помогая им ясно понять, чего можно ожидать от каждой конкретной операции.

5. Лучшее управление – прямое управление.

Лучший интерфейс – никакого интерфейса: например, с объектами реального мира мы взаимодействуем напрямую. Но постепенно появляется все больше объектов цифровой природы, которыми управлять напрямую невозможно, и тут нам на помощь приходят интерфейсы. Очень легко сбиться с верного пути и в итоге перегрузить интерфейс кнопками, финтифлюшками, графикой, параметрами, настройками, окнами, дополнительными вставками

и прочим «мусором». В результате пользователи вынуждены вместо выполнения задач заниматься управлением элементами интерфейса. Чтобы этого избежать, возьмите за образец прямое управление: интерфейс должен быть максимально незаметным и способным распознавать естественные человеческие жесты. В идеале у пользователей должно появиться ощущение, будто они управляют объектом напрямую.

6. Один экран – одна основная задача.

Каждый экран приложения должен служить для выполнения какой-либо одной задачи, стоящей перед пользователями. Так пользователям приложения будет проще в нем разобраться и с ним работать, а разработчикам – расширять его функционал при необходимости. Экраны, поддерживающие выполнение нескольких основных задач, сбивают пользователей с толку. Очевидно, что, например, текст не может содержать две или три главные темы – главная тема может быть только одна. Так и дизайнерам следует закладывать в каждый экран возможность выполнения только одной ключевой задачи.

7. Второстепенная задача, знай свое место.

Кроме какой-то одной ключевой задачи экраны также могут служить для выполнения нескольких второстепенных задач. Но важно помнить, что главные и второстепенные задачи нельзя валить в одну кучу. Второстепенные задачи не должны выходить на первый план: к примеру, они могут быть оформлены менее заметным образом или отображаться только после того, как была выполнена ключевая задача.

8. Место для шага вперед.

Так как большинство операций пользователей не обрывают процесс взаимодействия, а последовательно переходят друг в друга, весьма благоразумно будет спроектировать для каждой операции какое-нибудь продолжение. Постарайтесь представить себе, каким будет следующий шаг пользователей в каждом конкретном случае, и выстраивайте интерфейс в соответствии с этим. Как и в обычно человеческом общении, здесь нужна отправная точка для дальнейшего взаимодействия. Если пользователи уже сделали все, что требовалось, не бросайте их: дайте им возможность естественным образом сделать следующий шаг на пути к достижению их целей.

9. Поведение определяет внешний вид.

Люди предпочитают иметь дело с тем, что ведет себя предсказуемым образом: это равным образом относится к другим людям, животным, объектам – и в том числе к программному обеспечению. Когда чье-то поведение совпадает с нашими ожиданиями, мы чувствуем себя на правильной волне. Соответственно, элементы дизайна интерфейса должны выглядеть соответственно своему поведению. На практике это означает, что пользователи должны понимать, как поведет себя тот или иной элемент интерфейса, едва взглянув на него. Элемент, похожий на кнопку, и вести себя должен как кнопка. Не стоит заигрывать с основополагающими аспектами взаимодействий пользователей и интерфейса – приберегите свою творческую энергию для задач другого порядка.

10. Как важно быть последовательным.

Из предыдущего пункта следует, что если поведение экранных элементов

различается, то и выглядеть они должны по-разному. Безусловно, элементы, схожие в поведении, и выглядеть должны схожим образом (последовательно). Но не менее важно, чтобы несхожие элементы были оформлены по-разному (т. е. непоследовательно). Дизайнеры-новички, стараясь сделать интерфейс логичным и последовательным, зачастую игнорируют существенные различия между элементами и используют для их оформления одни и те же приемы там, где следовало бы внести разнообразие.

11. Четкая иерархия.

Четкая визуальная иерархия достигается, когда элементы на экране расположены в определенном порядке. То есть одни и те же элементы отображаются в одном и том же порядке

каждый раз. Плохо проработанная визуальная иерархия не приносит никакой пользы и только сбивает пользователей с толку. В постоянно изменяющихся средах не так-то просто поддерживать четкую иерархию элементов, потому что визуальный вес становится относительной величиной: ведь если выделено все, то не выделено ничего. Если на экран нужно добавить заметный элемент, дизайнеру может понадобиться сделать все остальные элементы менее заметными, чтобы сохранить визуальную иерархию. Большинство пользователей не задумываются о визуальной иерархии при работе с интерфейсом, но при этом ее продуманное (или непродуманное) выстраивание – это один из самых легких способов улучшить (или ухудшить) дизайн.

12. Грамотная организация снижает когнитивную нагрузку.

Джон Маэда (John Maeda) в своей книге Simplicity пишет, что грамотная организация элементов интерфейса позволяет придать экрану менее загруженный вид. С помощью продуманной организации элементов вы сможете продемонстрировать связи между ними, и освоить такой интерфейс пользователям будет куда проще. Группируйте схожие элементы, располагайте их на экране таким образом, чтобы пользователям стало понятно, как они связаны между собой. Благодаря грамотной организации контента можно значительно снизить когнитивную нагрузку пользователей. Если в самом дизайне будут наглядно продемонстрированы связи между элементами, пользователям уже не придется мучительно в них разбираться. Не заставляйте пользователей лишний раз напрягаться – лучше просто покажите им все эти связи между элементами интерфейса с помощью вашего дизайна.

13. Подсказывай, а не указывай: роль цвета.

Цвет физических объектов меняется в зависимости от освещения. Одно и то же дерево, например, в полдень и на закате выглядит совершенно по-разному. В общем, в мире физических объектов цвет включает в себя множество оттенков и вообще довольно относителен, и в интерфейсах цвет также должен играть соответствующую роль. Цветом можно выделять объекты, привлекая к ним внимание пользователей, но при этом элементы нельзя разделять только по цвету. Если предполагается, что пользователи будут работать с каким-либо элементом продолжительное время, или же элемент содержит объемный текст, рекомендуется использовать для оформления бледные или приглушенные тона – а яркие оттенки приберегите для расстановки акцентов. Разумеется, можно использовать яркие цвета и для фоновой заливки, но только там, где это уместно.

14. Не все сразу.

На каждом экране должно отображаться только самое необходимое. Если пользователям требуется сделать выбор, предоставьте им ровно столько информации, сколько им для этого нужно. Подробностям можно посвятить последующие экраны. Не надо пытаться объяснить все от А до Я или выложить всю информацию разом. По возможности распределяйте рабочий процесс на несколько экранов, раскрывая информацию постепенно. Благодаря этому взаимодействие с интерфейсом остается ясным и понятным для пользователей.

15. Подсказывай с умом.

В идеальных интерфейсах подсказки не нужны вовсе, потому что такой интерфейс легко изучить и использовать. Но если спуститься с небес на землю, то в идеале подсказки должны быть контекстно-зависимыми и появляться только тогда и там, где они нужны, в остальное время оставаясь скрытыми. Заставляя людей открывать справку и искать ответы на возникшие у них вопросы, вы затрудняете их работу с интерфейсом, так как в этом случае им приходится формулировать, что именно они хотят найти. Лучше встраивать подсказки там, где они могут потребоваться. Только убедитесь, что они не будут лишний раз маячить перед носом тех пользователей, которые уже знакомы с интерфейсом.

16.Стартовая страница.

Дизайнеры часто упускают из вида такой важный момент, как первое знакомство с интерфейсом. Чтобы помочь пользователям как можно быстрее освоиться, дизайнер должен рабо-

тать с прицелом на нулевое состояние – тот момент, когда еще ничего не произошло. Первый экран, который видят пользователи, не должен быть пустым, аки чистый лист, – на нем должны содержаться указания и подсказки для быстрого начала работы. Большинство затруднений при работе с интерфейсом возникает на почве непродуманного нулевого состояния. Но стоит пользователям понять правила игры, как их задача сразу значительно упрощается.

17. Текущие проблемы – главные проблемы.

Пользователям нужно решать задачи, актуальные на данный момент, а не гипотетические вопросы, которые могут возникнуть в будущем. Таким образом, интерфейс, ориентированный на решение потенциальных проблем, никому не будет нужен: изучайте текущую ситуацию и разрабатывайте интерфейс в соответствии с актуальными проблемами. Витать в облаках и строить гипотезы, конечно, гораздо увлекательнее, но зато результаты вашего труда окажутся востребованы благодарными пользователями, а не отправлены на свалку бесполезных интерфейсов.

18. Лучший дизайн – невидимый дизайн

Интересный факт: действительно хорошие дизайны обычно никак не отмечаются пользователями, работавшими с ними. Причина заключается в том, что удачный дизайн позволяет пользователям сконцентрироваться на их задачах, а не на работе интерфейса. Пользователи, успешно выполнившие свои задачи, не станут задумываться над тем, как это так у них все хорошо получилось. Получается, что пользователи обращают внимание на дизайн только в том случае, если у них возникают какие-либо трудности. Да, дизайнеры не в восторге от того, что за удачные решения их никто не хвалит, но действительно хорошим специалистам вполне достаточно того, что их дизайном активно пользуются. Они понимают, что довольный пользователь – это молчаливый пользователь.

19. Расширяем кругозор

Визуальный и графический дизайн, полиграфия, копирайтинг, информационная архитектура и визуализация – все это входит в дизайн интерфейсов. С этими дисциплинами можно знакомиться вскользь, а можно углубиться в их изучение. Черпайте в них полезные знания – и вперед. Не брезгуйте и на первый взгляд абсолютно не связанными с дизайном интерфейсов сферами.

20. Неиспользуемый интерфейс – плохой интерфейс

Как и в других областях дизайна, в дизайне интерфейсов успешным считается тот результат дизайнерских трудов, который оказывается востребован пользователями. Люди не сядут даже в самое красивое кресло, если оно окажется неудобным, и этот предмет мебели не выполнит свою функцию, как и дизайн, который пользователи обходят вниманием. Таким образом, в дизайне интерфейсов важную роль играет создание не только самого объекта, но и некоей среды его использования. Дизайнер создает интерфейс не для услады собственных очей, а для того чтобы им пользовались.

Цвет в дизайне

Цвет – это метод создания баланса элементов.

Во время продумывания сценария взаимодействия и восприятия интерфейса следует на каждом этапе помнить, что пользователь видит экран рабочими областями (Рисунок 2.1).

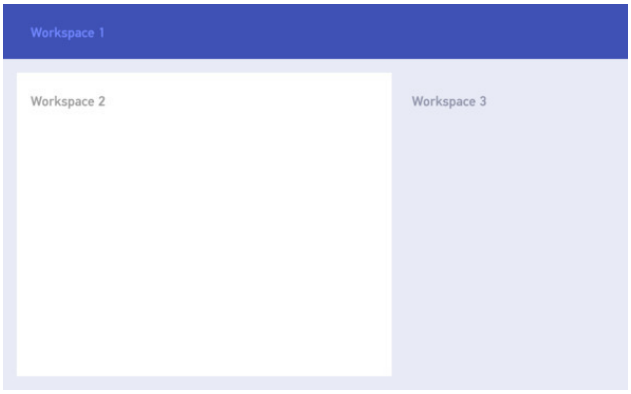


Рисунок 2.1. Рабочие области простого интерфейса

Характерно, что последовательность восприятия, у большинства пользователей, будет несколько отлична от нумерации рабочих областей, а именно: workspace 2 → workspace 1 → workspace 3, при этом workspace 2 и 3, очевидно, будут восприниматься как основная и второстепенная часть одного сценария.

Пользователь стремится игнорировать элементы, находящиеся в другой рабочей области, во время работы с текущей – подсознательно он представляет, что где-то там находится ряд функций, которые ему могут понадобится, но сознательно он их «не видит». Отсюда можно сделать вывод, что элементы внутри областей должны находиться на одном уровне интенсивности относительно других областей. Например на рисунке 2.2 очевидно, что в первом варианте сбивается изначально заданный баланс: элементы списка справа конфликтуют с горизонтальной шапкой. Во втором варианте он сохранен: внимание в первую очередь направленно на заголовок Workspace1.

Рисунок 2.2. Цветовое акцентирование на главное рабочей зоне

Следует использовать только сознательно подобранные цвета. Это основная ошибка очень большого количества дизайнеров. Часто можно видеть пример, когда дизайнер берет элемент, основываясь на абстрактных представлениях о том, что «подтвердить» должно быть зеленым, а «отменить» – красным. Помимо явной ограниченности такого подхода есть еще и проблема того, что не все понимают, что, если ты использовал определенный зеленый цвет, ты можешь к нему использовать только строго определенный красный. Случайных цветов не бывает вообще: человеческий глаз способен улавливать малейшие отличия и подсознательно всегда воспринимает цветовой диссонанс. Более того, следует помнить, что большинство современных экранов имеют весьма ограниченные цветовые пространства, и диссонансы на них представляются гораздо более грубыми, нежели, скажем, на холсте. Пара примеров подобного отношения (рисунок 2.3.):

Рисунок 2.3. Различные варианты использования цвета в одном и том же приложении.

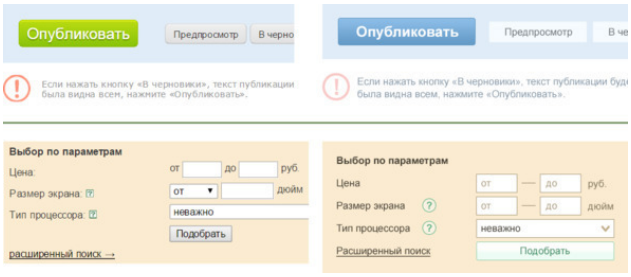


Рисунок 2.3. Различные варианты использования цвета в одном и том же приложении.

Обратим внимание на то, что ни один компонент элемента, будь то цвет текста, цвет обводки значков, цвет обводки кнопки, не оставлен без внимания – это создает цельность и гармоничность. Оттенок кнопки совершенно необязательно должен быть зеленым, важно, что если это зеленый, то он не случаен.

Существует несколько ограничений, которые нужно всегда держать в голове:

– Никогда не использовать черный и серый цвет текста на цветном фоне. Это создаёт ощущение грязи. На белом фоне, кстати, тоже далеко не во всех случаях можно использовать чистый черный цвет.

– Не использовать глухой серый для фонов. Если на макете есть хотя бы один цветной элемент, всегда следует окрашивать серые элементы в его оттенок.

– Не использовать правый крайний угол палитры для основных цветов.

Ниже отмечена приемлемая зона выбора четкого цвета, но не бледных оттенков (Рисунок 2.4):

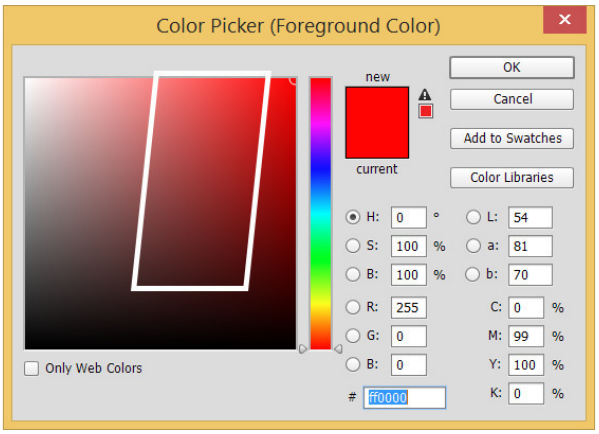


Рисунок 2.4. Зона выбора оптимальных цветов для основных элементов.

Кстати, веб-цвета тоже настоятельно не рекомендуется использовать, особенно учитывая современное развитие браузеров. Веб-цвета – это всего лишь оптимизация значений, они на удивление грубы и плохо соотносятся между собой. Во время разработки палитры интерфейса стоит использовать различные генераторы палитр и цветовых схем. Например это Kuler от компании Adobe или Paletton (Рисунок 2.5.). Разумеется, все вышеперечисленное следует воспринимать с умом, а не в лоб. Для того, чтобы грамотно подобрать практически любой цвет, как правило, приходится подвигать ползунок тона, т.к. чистые относительные цвета обычно недостаточно/чрезмерно контрастны для своих задач.

Рисунок 2.5. Генератор цветовых схем.

Человеческое зрение очень хорошо умеет адаптироваться к условиям различного контраста, например, его статический контраст равен примерно 100:1, а динамический может достигать 1.000.000:1, поэтому не следует бояться его понижения внутри элементов. При этом восприятие контрастности повышается с уменьшением циклов (грубо говоря, разноцветных элементов). В отличие от реального мира, компьютерный интерфейс имеет весьма примитивную структуру контрастов, небольшое количество цветов и типов элементов. Хороший пример грамотно выстроенного контраста – текущая версия социальной сети Facebook (Рисунок 2.6.):

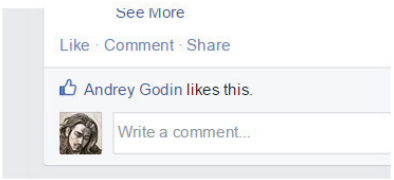


Рисунок 2.6. Цвет и контраст в интерфейсе Facebook.

Обратим внимание на то, что легкие контрасты между областями тем не менее хорошо отделяют их друг от друга. Визуально сохраняется даже наследственность (область комментариев явно принадлежит общей области поста), и это заслуга не только обводки.

Хорошим показателем того, что палитра в интерфейсе удалась, если он создает «на вкус» ощущение единого освещения. Это достигается путем использования естественных контрастов, насыщенных оттенком теней. Например, не рекомендуется использовать холодное на теплом в качестве акцента. Это противоестественно, в природе такого не бывает. Даже подобранные по цветовым отношениям синий в красном смотрится хуже, чем красный в синем (Рисунок 2.7):

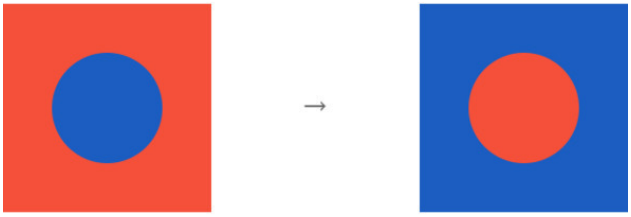


Рисунок 2.7. Использование естественных контрастов

Также следует давать пользователю ясный фокус на одном первостепенном сценарии и делать остальные сценариии явно второстепенными. Отсюда следует хрестоматийное call-to-action. Однако следует понимать, что в 90% случаев call-to-action является кнопкой и предваряется какой-то информативной частью. Это означает что она должна быть легко находима, но не бросаться в глаза первой. Это достигается, например, при помощи яркого, активного цвета в элементе небольшого размера (Рисунки 2.8; 2.9; 2.10)

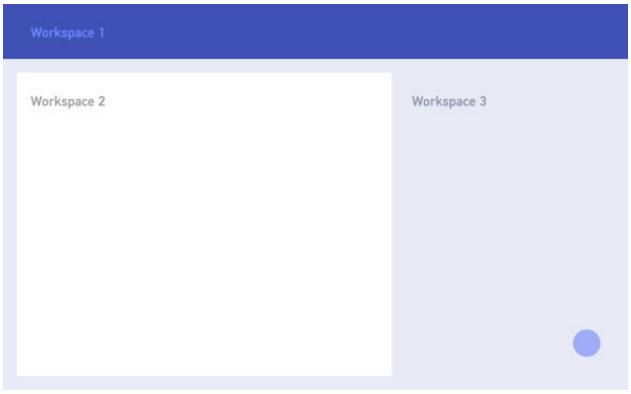


Рисунок 2.8.Call-to-action как часть сценария workspace 2.

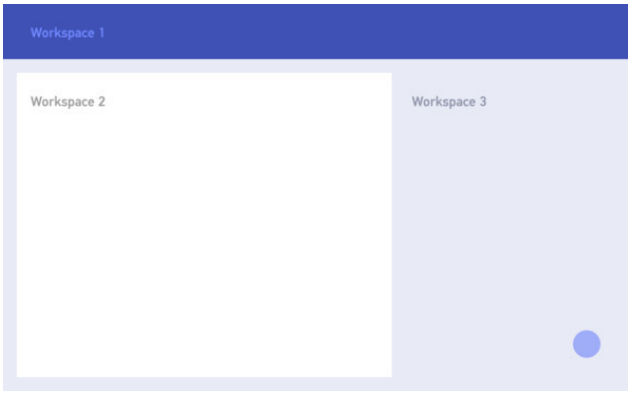


Рисунок 2.9.Call-to-action как часть сценария workspace 3.

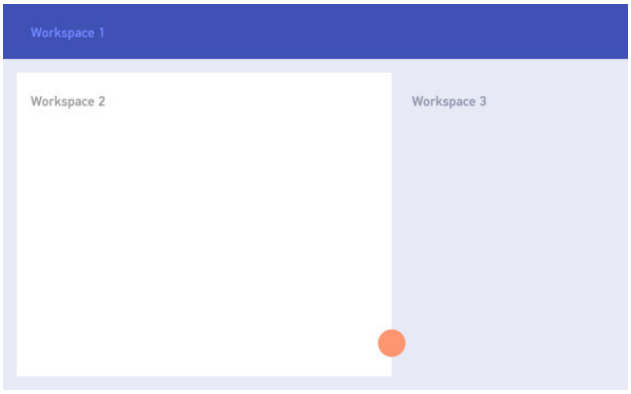


Рисунок 2.10.Call-to-action как отдельный сценарий.

3.Мобильный интерфейс для одной руки

В своей книге Designing Mobile Interfaces (2011) дизайнер Стивен Хубер ввел понятие The Thumb Zone («зона большого пальца») – область экрана, наиболее удобная при использовании телефона одной рукой. С года издания книги средний размер смартфона заметно увеличился, и «мертвая зона» – область, которую сложно достать пальцем одной руки, – также стала больше (Рисунок 3.1):

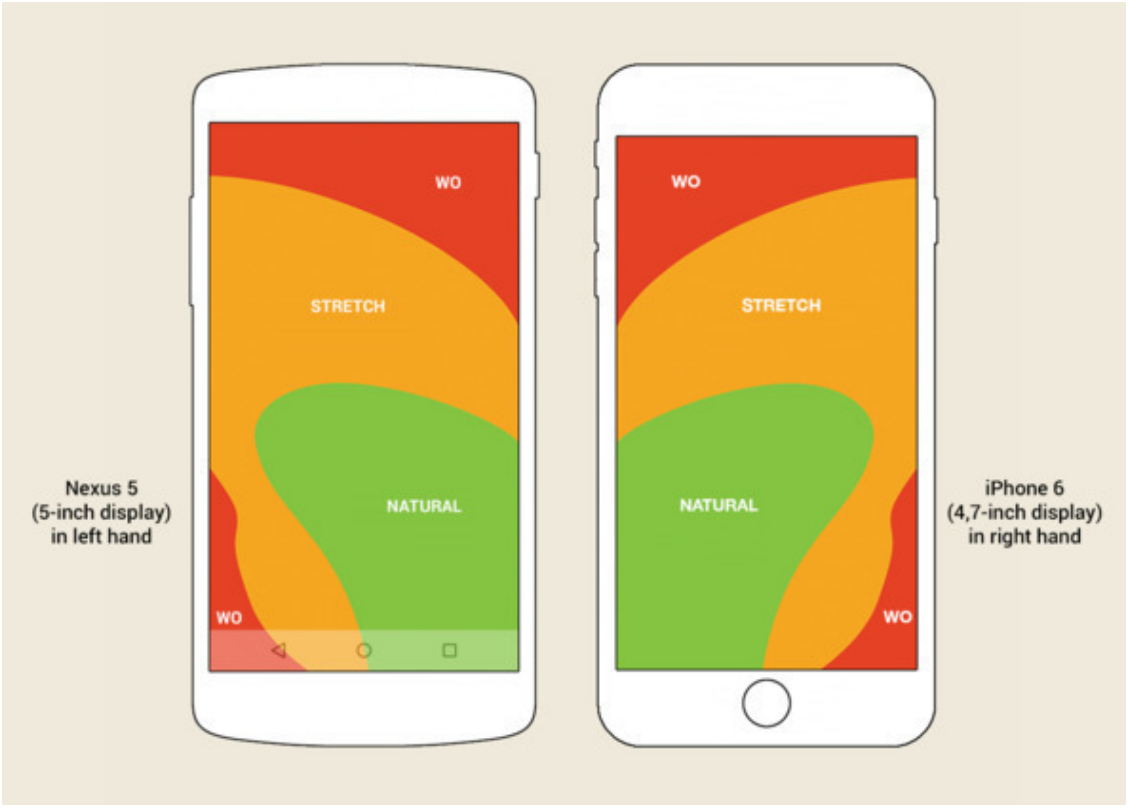


Рисунок 3.1. «Зона большого пальца» для 5—4,7-дюймового экрана.

В «мертвую зону», отмеченную красным, попадают панели инструментов приложений, которые как в Android (App Bar / Primary Toolbar), так и в iOS (Navigation Bar) находятся в верхней части экрана.

Настоящее решение данной проблемы должно не помогать дотянуться до нужной кнопки, а избавить пользователя от необходимости дотягиваться.

В эпицентре «мертвой зоны» iOS стандартно расположена кнопка Back, на телефонах Android в данную область попадает кнопка вызова бокового меню (Navigation Drawer). Однако в большинстве приложений нет необходимости к ней тянуться – достаточно сделать свайп от левого края телефона (Рисунок 3.2):

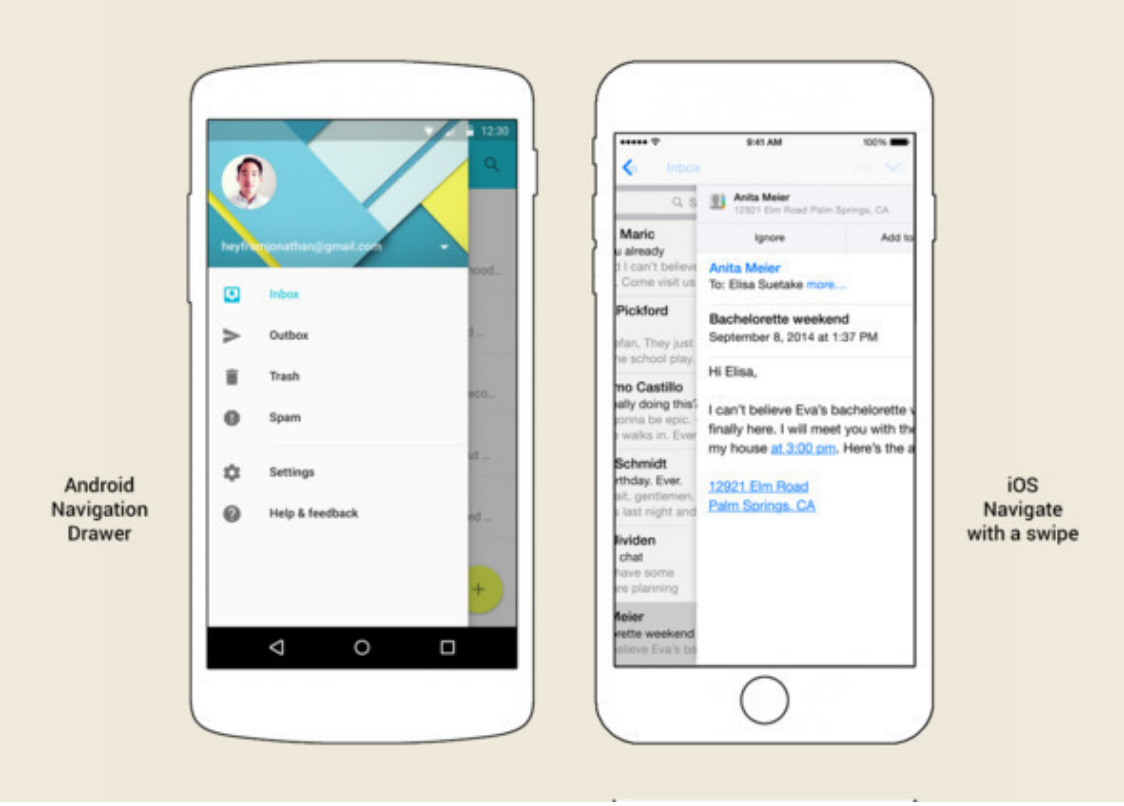


Рисунок 3.2. Боковое (главное) меню (Navigation Drawer) в Android и функция Navigare with a swipe («Навигация смахиванием») в iOS.

Но не бывает правил без исключений. С правой стороны панели инструментов могут находиться кнопки «Готово» или «Отправить», которые ведут к необратимому действию. Такое простое движение как swipe не должно приводить к выполнению необратимых действий.

Использование swipe от левого и правого края телефона для активации функций верхней панели инструментов легко применимо как для iOS, так и для Android. Более того, подобный функционал в отношении левой стороны уже частично реализован на обеих платформах. Такое интуитивно-понятное поведение имеет все шансы стать стандартом мобильных интерфейсов.

Гайдлайны для различных мобильных платформ

Разрабатывая интерфейс, следует учитывать, на какой платформе будет выпускаться данное приложение: каждая платформа имеет свой отличительный стиль (Рисунки 4.1, 4.2, 4.3) которого и следует придерживаться и своем контенте. Можно заметить, что каждая платформа использует особенный шрифт, имеет различные размеры главных элементов интерфейса, по-своему группирует объекты на рабочей области:

Рисунок 4.1 Интерфейс IOS 8

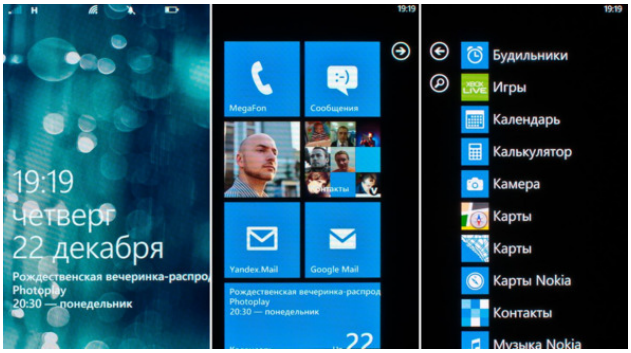


Рисунок 4.2 Интерфейс Windows phone

Рисунок 4.3 Интерфейс Android

Гайдлайны – это попытка описать набор правил для создания интерфейса приложений, то есть того самого опыта, который пользователи получают взаимодействуя с ОС и ее приложениями. Они выполняют двойную задачу. С одной стороны, использование гайдлайнов при проектировке приложения позволяет сократить время на проектирование, за счет использования готовых решений. С другой позволяет сократить время, которое тратит пользователь на обучение работы с вашим приложением. Но, следует понимать и то, что любой набор готовых решений не совершенен. Гайдлайны это рекомендации, их можно нарушать.

В ходе разработки интерфейсов дизайнеру, рано или поздно придется столкнуться с официальной документацией- ведь именно оттуда будет изыматься информация о структуре элементов. Каждая платформа имеет собственные стандарты на иконки, меню, и бекграунды. Как дизайнер, вы должны не усложнять работу и себе, и программистам: лучше сразу сделать все по стандартам, дабы избежать переделывания в будущем.

Функции устройства

Apple предоставляет возможность использовать функции телефона, такие как: акселерометр, работа с файловой, работа с почтой и др.

Существует сразу готовые решения, которые реализованы в различных фреймворках, таких как: core motion, message и т. д.

Из-за консервативной системы Apple – функционал, предоставляемый приложениям, ограничен. Нет возможности использовать системные возможности, такие как: работа с питанием, работа с файловой системой (приложение может только читать файлы, а изменять может только в своей «песочнице») и т. д.

Акселерометр

Акселерометр, с технической точки зрения, представляет из себя устройство, способное измерять ускорение предмета, которое оно приобретает при смещении относительно своего нулевого положения. Акселерометр применяется как для измерения ускорения в сторону, в которую произошло смещение, так и для измерения ускорения, вызванного силой тяжести Земли. Не работает в вакууме.

Конец ознакомительного фрагмента.